

# xPC Target™

## I/O Reference

**R2012b**

**MATLAB®**  
& **SIMULINK®**

## How to Contact MathWorks



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



[suggest@mathworks.com](mailto:suggest@mathworks.com) Product enhancement suggestions  
[bugs@mathworks.com](mailto:bugs@mathworks.com) Bug reports  
[doc@mathworks.com](mailto:doc@mathworks.com) Documentation error reports  
[service@mathworks.com](mailto:service@mathworks.com) Order status, license renewals, passcodes  
[info@mathworks.com](mailto:info@mathworks.com) Sales, pricing, and general information



508-647-7000 (Phone)



508-647-7001 (Fax)



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

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

*xPC Target™ I/O Reference*

© COPYRIGHT 2000–2012 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 and Simulink are registered trademarks of The MathWorks, Inc. See [www.mathworks.com/trademarks](http://www.mathworks.com/trademarks) for a list of additional trademarks. Other product or brand names may be trademarks or registered trademarks of their respective holders.

### Patents

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+)
September 2006	Online only	Revised for Version 3.1 (Release 2006b)
March 2007	Online only	Revised for Version 3.2 (Release 2007a)
September 2007	Online only	Revised for Version 3.3 (Release 2007b)
March 2008	Online only	Revised for Version 3.4 (Release 2008a)
October 2008	Online only	Revised for Version 4.0 (Release 2008b)
March 2009	Online only	Revised for Version 4.1 (Release 2009a)
September 2009	Online only	Revised for Version 4.2 (Release 2009b)
March 2010	Online only	Revised for Version 4.3 (Release 2010a)
September 2010	Online only	Revised for Version 4.4 (Release 2010b)
April 2011	Online only	Revised for Version 5.0 (Release 2011a)
September 2011	Online only	Revised for Version 5.1 (Release 2011b)
March 2012	Online only	Revised for Version 5.2 (Release 2012a)
September 2012	Online only	Revised for Version 5.3 (Release 2012b)



## xPC Target I/O Library

**1**

<b>I/O Driver Blocks</b> .....	<b>1-2</b>
Introduction .....	1-2
xPC Target Turnkey I/O Modules .....	1-2
Third-Party Driver Blocks .....	1-3
I/O Driver Block Library .....	1-3
Memory-Mapped Devices .....	1-4
ISA Bus I/O Devices .....	1-5
PCI Bus I/O Devices .....	1-5
xPC Target I/O Driver Structures .....	1-6
xPC Target Support and SimState .....	1-8
PWM and FM Driver Block Notes .....	1-10
Driver Block Documentation .....	1-11
<b>Adding I/O Blocks with the xPC Target Library</b> .....	<b>1-12</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>1-18</b>

## Blocks — By Category

**2**

<b>A/D Converter</b> .....	<b>2-4</b>
<b>A/D Frame Converter</b> .....	<b>2-11</b>
<b>ARINC 429</b> .....	<b>2-12</b>

<b>Asynchronous Event</b> .....	<b>2-13</b>
<b>Audio</b> .....	<b>2-14</b>
<b>Controller Area Network (CAN) Protocols</b> .....	<b>2-15</b>
CAN (DOB) .....	<b>2-15</b>
CAN (FIFO) .....	<b>2-16</b>
<b>Counter</b> .....	<b>2-18</b>
<b>D/A Converter</b> .....	<b>2-20</b>
<b>Data Marshalling</b> .....	<b>2-28</b>
<b>Digital Input</b> .....	<b>2-29</b>
<b>Digital Output</b> .....	<b>2-39</b>
<b>Digital Signal Processing</b> .....	<b>2-49</b>
<b>Ethernet Protocols</b> .....	<b>2-50</b>
Ethernet .....	<b>2-50</b>
Ethernet (Network Buffers) .....	<b>2-50</b>
Ethernet (EtherCAT) .....	<b>2-51</b>
<b>Field Programmable Gate Arrays (FPGA)</b> .....	<b>2-52</b>
<b>General Purpose Interface Bus (GPIB)</b> .....	<b>2-53</b>
<b>Incremental Encoder</b> .....	<b>2-54</b>
<b>I/O Pack (IP) Carrier</b> .....	<b>2-56</b>
<b>J1939 (CAN)</b> .....	<b>2-57</b>
<b>Light Emitting Diodes (LED)</b> .....	<b>2-58</b>

<b>Linear Variable Differential Transformer (LVDT) . . . .</b>	<b>2-59</b>
<b>MIL-STD-1553 . . . . .</b>	<b>2-60</b>
<b>Miscellaneous . . . . .</b>	<b>2-62</b>
<b>Parallel Port . . . . .</b>	<b>2-63</b>
<b>Pulse Width Measurement . . . . .</b>	<b>2-64</b>
<b>Real-Time Performance . . . . .</b>	<b>2-67</b>
<b>RS-232 . . . . .</b>	<b>2-68</b>
<b>Shared Memory . . . . .</b>	<b>2-69</b>
<b>Signal Conditioning . . . . .</b>	<b>2-70</b>
<b>Synchro Resolver . . . . .</b>	<b>2-71</b>
<b>Thermocouple . . . . .</b>	<b>2-72</b>
<b>User Datagram Protocol (UDP) Protocols . . . . .</b>	<b>2-73</b>
UDP . . . . .	2-73
UDP (Real Time) . . . . .	2-73
UDP (Internet Protocol) . . . . .	2-73
<b>Utility . . . . .</b>	<b>2-75</b>
<b>Video . . . . .</b>	<b>2-76</b>
<b>Visualization . . . . .</b>	<b>2-77</b>
<b>XCP (CAN) . . . . .</b>	<b>2-78</b>
<b>Watchdog Timer . . . . .</b>	<b>2-79</b>

## Boards and Blocks — By Manufacturer

### 3

<b>ACCES I/O</b> .....	<b>3-4</b>
<b>ADDI-DATA</b> .....	<b>3-5</b>
<b>Adlink</b> .....	<b>3-6</b>
<b>Advantech</b> .....	<b>3-7</b>
<b>Analogic</b> .....	<b>3-10</b>
<b>BitFlow</b> .....	<b>3-11</b>
<b>BittWare</b> .....	<b>3-12</b>
<b>BVM</b> .....	<b>3-13</b>
<b>Commtech</b> .....	<b>3-14</b>
<b>Contec</b> .....	<b>3-15</b>
<b>Curtiss-Wright Electronic Systems</b> .....	<b>3-17</b>
<b>Data Translation</b> .....	<b>3-18</b>
<b>Diamond Systems</b> .....	<b>3-21</b>
<b>GE Intelligent Platforms</b> .....	<b>3-24</b>
ARINC 429 Support .....	<b>3-24</b>
GE Fanuc Embedded Systems Shared Memory Support ..	<b>3-24</b>
SBS Technologies Support .....	<b>3-25</b>
MIL-STD-1553 Support .....	<b>3-27</b>
<b>General Standards</b> .....	<b>3-29</b>

<b>HUMUSOFT</b> .....	<b>3-30</b>
<b>Intel</b> .....	<b>3-31</b>
<b>Keithley</b> .....	<b>3-32</b>
<b>Measurement Computing</b> .....	<b>3-33</b>
<b>MPL</b> .....	<b>3-45</b>
<b>National Instruments</b> .....	<b>3-46</b>
<b>North Atlantic Industries</b> .....	<b>3-63</b>
<b>Philips</b> .....	<b>3-64</b>
<b>Quanser</b> .....	<b>3-66</b>
<b>Qatech</b> .....	<b>3-67</b>
<b>Real Time Devices</b> .....	<b>3-68</b>
<b>Sensoray</b> .....	<b>3-70</b>
<b>Softing</b> .....	<b>3-71</b>
<b>Speedgoat</b> .....	<b>3-73</b>
<b>Texas Instruments</b> .....	<b>3-75</b>
<b>United Electronic Industries (UEI)</b> .....	<b>3-76</b>
<b>Versallogic</b> .....	<b>3-82</b>

## Serial Communications Support

### 4

<b>Serial Drivers</b> .....	4-2
RS-232 I/O Communication .....	4-2
Hardware Connections for RS-232 .....	4-4
Host and Target Computer Communication .....	4-4
<b>RS-232/422/485 Drivers (Composite)</b> .....	4-6
Introduction .....	4-6
Adding RS-232 Blocks .....	4-7
Building and Running the Target Application (Composite) .....	4-13
RS-232/422/485 Simulink Block Reference .....	4-14
<b>Boards and Blocks — Alphabetical List</b> .....	4-22

## Serial Communications Obsolete Drivers Support

### 5

<b>Obsolete RS-232 Drivers</b> .....	5-2
Introduction .....	5-2
xPC Target RS-232 Drivers (Obsolete) .....	5-2
<b>Obsolete Boards and Blocks — Alphabetical List</b> .....	5-35

## GPIB I/O Support

### 6

<b>GPIB Drivers</b> .....	6-2
Introduction .....	6-2
Hardware Connections for GPIB .....	6-2
Simulink Blocks for GPIB .....	6-3

MATLAB Message Structures for GPIB .....	6-3
<b>Using GPIB Drivers</b> .....	<b>6-5</b>
Introduction .....	6-5
Adding GPIB Driver Blocks .....	6-5
Creating GPIB Message Structures .....	6-10
<b>GPIB MATLAB Structure Reference</b> .....	<b>6-13</b>
Introduction .....	6-13
GPIB Initialization and Termination Message Structures .....	6-13
GPIB Send/Receive Message Structure .....	6-14
Shortcuts and Features for Messages .....	6-17
Supported Data Types for Message Fields .....	6-19
<b>Boards and Blocks — Alphabetical List</b> .....	<b>6-20</b>

## CAN I/O Support

# 7

<b>CAN Basics</b> .....	<b>7-2</b>
xPC Target CAN Library .....	7-2
Data Types .....	7-4
Sending and Receiving Remote Frames .....	7-5
Supported Softing Boards .....	7-5
<b>Model Execution Driven by CAN Messages</b> .....	<b>7-79</b>
CAN-AC2 (ISA) .....	7-79
CAN-AC2-PCI .....	7-80
CAN-AC2-104 (PC/104) .....	7-81
<b>Initialization and Termination CAN Messages</b> .....	<b>7-83</b>
Defining Initialization and Termination CAN Messages ..	7-83
A/D Converter with CANopen Interface .....	7-84
<b>CAN-AC2 and CANopen Devices</b> .....	<b>7-86</b>

<b>CAN Data Frames</b> .....	7-87
<b>Timeouts When Receiving CAN Messages</b> .....	7-88
<b>Blocks — Alphabetical List</b> .....	7-91

## CAN I/O Support for FIFO

# 8

<b>CAN FIFO Basics</b> .....	8-2
FIFO Mode .....	8-2
FIFO Mode Drivers for CAN Boards from Softing .....	8-5
<b>Acceptance Filters</b> .....	8-47
<b>CANdb DBC Format Databases</b> .....	8-49
<b>CAN FIFO Loopback Tests</b> .....	8-50
<b>Blocks — Alphabetical List</b> .....	8-53

## Model-Based Ethernet Communications Support

# 9

<b>Model-Based Ethernet Communications</b> .....	9-2
What Is Model-Based Ethernet Communications? .....	9-2
PCI Bus and Slot Numbers .....	9-3
MAC Addresses .....	9-3
Network Buffer Pointers .....	9-4
Filter Type and Filter Address Blocks .....	9-4
Execution Priority .....	9-4
xPC Target Ethernet Block Library .....	9-4

Using xPC Target for Ethernet Communications .....	9-6
Blocks — Alphabetical List .....	9-7

## Network Buffer Library for Model-Based Ethernet Communications Support

# 10

Network Buffer Blocks .....	10-2
Blocks — Alphabetical List .....	10-3

## Model-Based EtherCAT Communications Support

# 11

<b>Model-Based EtherCAT Communications</b> .....	11-2
EtherCAT Protocol .....	11-2
Before You Start .....	11-2
Basic Workflow .....	11-3
EtherCAT Chip Sets .....	11-6
EtherCAT Data Types .....	11-7
EtherCAT Error Values .....	11-8
 Blocks — Alphabetical List .....	 11-11

## Real-Time UDP Communication Support

# 12

Using Real-Time UDP Communications .....	12-2
Before You Start .....	12-2

Blocks — Alphabetical List .....	12-3
----------------------------------	------

## IP Library for Real-Time UDP Communication Support

### 13

Internet Protocol Blocks .....	13-2
Blocks — Alphabetical List .....	13-3

## UDP I/O Support

### 14

User Datagram Protocol (UDP) .....	14-2
What Is UDP? .....	14-2
Why UDP? .....	14-4
Note on UDP Communication .....	14-4
Two-Way Data Exchange Using UDP .....	14-6
UDP Communication Setup .....	14-12
UDP and Variable-Size Signals .....	14-14
Boards and Blocks — Alphabetical List .....	14-16

## Parallel Ports

### 15

Using Parallel Ports .....	15-2
Introduction .....	15-2

Using the Parallel Port as an Interrupt Source .....	15-3
Using Add-On Parallel Port Boards .....	15-4

<b>Boards and Blocks — Alphabetical List .....</b>	<b>15-6</b>
--	-------------

## SAE J1939

# 16

<b>SAE J1939 Blocks .....</b>	<b>16-2</b>
<b>Blocks — Alphabetical List .....</b>	<b>16-3</b>

## Shared Memory Support

# 17

<b>GE Fanuc Embedded Systems Shared Memory .....</b>	<b>17-2</b>
Before You Start .....	17-2
Shared Memory Structure Reference .....	17-5
<b>GE Intelligent Platforms Shared Memory .....</b>	<b>17-13</b>
Broadcast Memory .....	17-13
Shared Memory Structure Reference .....	17-18
<b>Curtiss-Wright Electronic Systems Shared Memory ...</b>	<b>17-23</b>
Before You Start .....	17-23
Systran Shared Memory Structure Reference .....	17-26
<b>Boards and Blocks — Alphabetical List .....</b>	<b>17-37</b>

## Video Image Processing

# 18

<b>Processing Video Images Using xPC Target</b> .....	18-2
xPC Target Video Basics .....	18-2
Before You Start .....	18-2
Basic Workflow .....	18-3
<b>Acquiring Images from Camera Link Cameras</b> .....	18-5
Introduction .....	18-5
Before You Start .....	18-5
Camera Triggering .....	18-7
Serial Camera Configuration .....	18-12
Offline Camera Configuration .....	18-13
BitFlow Neon-CLB Example .....	18-14
<b>Boards and Blocks — Alphabetical List</b> .....	18-15

## XCP Over CAN Support

# 19

<b>XCP Over CAN</b> .....	19-2
Introduction .....	19-2
Using the XCP Blocks .....	19-2
<b>Blocks — Alphabetical List</b> .....	19-4

## ACCES I/O

# 20

<b>Boards and Blocks — Alphabetical List</b> .....	20-2
--	------

**21**

Boards and Blocks — Alphabetical List ..... 21-2

**22**

Boards and Blocks — Alphabetical List ..... 22-2

**23**

Boards and Blocks — Alphabetical List ..... 23-3

**24**

Boards and Blocks — Alphabetical List ..... 24-2

**25**

Running Models with BittWare Blocks ..... 25-2  
    Model Notes ..... 25-2  
    Frame Size, Sample Rate, and Sample Time Parameter  
        Notes ..... 25-4  
  
Boards and Blocks — Alphabetical List ..... 25-6

**BVM**

**26**

---

Boards and Blocks — Alphabetical List ..... 26-2

**Contec**

**27**

---

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

**Curtiss-Wright Electronic Systems**

**28**

---

Board and Blocks — Alphabetical List ..... 28-2

**Data Translation**

**29**

---

Boards and Blocks — Alphabetical List ..... 29-4

**Diamond**

**30**

---

Boards and Blocks — Alphabetical List ..... 30-4

**31**

**ARINC 429 Support** ..... 31-2

**MIL-STD-1553 Support** ..... 31-3

    Before You Start ..... 31-3

    Remote Terminal Operation ..... 31-5

    Bus Controller Operation ..... 31-7

    Remote Terminal and Bus Controller Operation ..... 31-10

    Bus Monitor Operation ..... 31-12

    Boards and Blocks ..... 31-14

**SBS Technologies Support** ..... 31-17

**Boards and Blocks — Alphabetical List** ..... 31-19

General Standards

**32**

**PMC-ADADIO Functionality** ..... 32-2

    PMC-ADADIO Basics ..... 32-2

    A/D Blocks ..... 32-3

    Create Enable Signal Blocks ..... 32-6

    D/A Blocks ..... 32-9

    Interleaving Analog Input and Output Blocks ..... 32-11

    Using Multiple PMC-ADADIO Boards ..... 32-13

**Overview of Audio Applications** ..... 32-17

**Boards and Blocks — Alphabetical List** ..... 32-18

**33**

---

Boards and Blocks .....	33-2
-------------------------	------

**Keithley**

**34**

---

Boards and Blocks — Alphabetical List .....	34-2
---	------

**MathWorks**

**35**

---

xPC TargetBox I/O Options .....	35-2
---------------------------------	------

Blocks — Alphabetical List .....	35-3
----------------------------------	------

**Measurement Computing**

**36**

---

Boards and Blocks — Alphabetical List .....	36-6
---	------

**MPL**

**37**

---

Boards and Blocks — Alphabetical List .....	37-2
---	------

**National Instruments**

---

**38**

Boards and Blocks — Alphabetical List ..... 38-6

**North Atlantic Industries, Inc.**

---

**39**

Boards and Blocks — Alphabetical List ..... 39-3

**Quanser**

---

**40**

Boards and Blocks — Alphabetical list ..... 40-2

**Real Time Devices**

---

**41**

Boards and Blocks — Alphabetical List ..... 41-3

**Sensoray**

---

**42**

Boards and Blocks — Alphabetical List ..... 42-2

**43**

**Blocks — Alphabetical List** ..... 43-4

**Texas Instruments**

**44**

**Boards and Blocks — Alphabetical List** ..... 44-2

**United Electronic Industries (UEI)**

**45**

**Grouping the UEI Boards** ..... 45-5  
    Introduction ..... 45-5  
    Changing the Board Associated with the Block ..... 45-6  
    Getting Information on a Specific Board ..... 45-6

**Analog Input Frame Driver Blocks** ..... 45-7  
    Introduction ..... 45-7  
    Notes on Master and Slave Boards ..... 45-7  
    Interrupt Numbers ..... 45-8  
    IRQ Source Block ..... 45-10  
    Example Models ..... 45-11

**Boards and Blocks — Alphabetical List** ..... 45-13

**Versallogic**

**46**

**Boards and Blocks — Alphabetical List** ..... 46-2

## Miscellaneous Blocks

### 47

<b>Asynchronous Event Support</b> .....	47-2
Adding an Asynchronous Event .....	47-2
Asynchronous Interrupt Examples .....	47-4
<b>Blocks — Alphabetical List</b> .....	47-6

## Utility Blocks

### 48

<b>Blocks — Alphabetical List</b> .....	48-2
---	------

## Obsolete Drivers

### 49

<b>xPC Target Library of Obsolete Drivers</b> .....	49-2
<b>Obsolete Blocks and Boards — Alphabetical List</b> .....	49-4

## Serial Communications Support with Internal Drivers

### A

<b>RS-232/422/485 Internal Blocks</b> .....	A-2
<b>Blocks — Alphabetical List</b> .....	A-3



# xPC Target I/O Library

---

- “I/O Driver Blocks” on page 1-2
- “Adding I/O Blocks with the xPC Target Library” on page 1-12
- “Adding I/O Blocks with the Simulink Library Browser” on page 1-15
- “Defining I/O Block Parameters” on page 1-18

## I/O Driver Blocks

In this section...
“Introduction” on page 1-2
“xPC Target Turnkey I/O Modules” on page 1-2
“Third-Party Driver Blocks” on page 1-3
“I/O Driver Block Library” on page 1-3
“Memory-Mapped Devices” on page 1-4
“ISA Bus I/O Devices” on page 1-5
“PCI Bus I/O Devices” on page 1-5
“xPC Target I/O Driver Structures” on page 1-6
“xPC Target Support and SimState” on page 1-8
“PWM and FM Driver Block Notes” on page 1-10
“Driver Block Documentation” on page 1-11

### Introduction

The xPC Target™ environment is a solution for prototyping, testing, and deploying real-time systems using standard PC hardware. In support of this, the software allows you to add I/O blocks to your model. The blocks of the xPC Target library provides 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.

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.

### xPC Target Turnkey I/O Modules

xPC Target Turnkey is available with a number of I/O modules. See the **I/O Modules** tab of xPC Target Turnkey for a list of these I/O modules.

## Third-Party Driver Blocks

In addition to the blocks contained in the xPC Target library, you can also use third-party driver blocks in your xPC Target model. The description of these blocks is beyond the scope of the xPC Target documentation. See the provider of the third-party driver blocks for information on those boards and driver blocks.

## 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 is included with the xPC Target software.

Note, if your model contains I/O blocks, take hardware latency values into account for the model sample time. Use the xPC Target Interactive Guide tool to find latency values for the supported boards:

[http://www.mathworks.com/support/product/XP/productnews/-interactive\\_guide/xPC\\_Target\\_Interactive\\_Guide.html](http://www.mathworks.com/support/product/XP/productnews/-interactive_guide/xPC_Target_Interactive_Guide.html)

The xPC Target system 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.

This library also contains the following blocks:

- xPC Target Driver Examples — When you double-click this block, the **Demos** tab in the MATLAB Help Navigator opens, displaying the xPC Target examples and example 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.

---

**Note** The xPC Target documentation describes only the xPC Target blocks. It does not describe the actual board. Refer to the board manufacturer documentation for information about the boards.

---

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

When you double-click one of the 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. The xPC Target software 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 the xPC Target software. 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. Select a memory range supported by both the device and the xPC Target software.

## ISA Bus I/O Devices

There are two types of ISA boards:

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

The xPC Target software 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 computer, 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 computer monitor shows parameters for any PCI boards installed on the target computer. 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, you must provide a more textual description to indicate what the driver has to do at runtime. *Textual* in this context refers to a programming-language-like syntax and style.

The xPC Target software 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(2).field	Message(3).field
Message(1).field	Message(2).field	Message(3).field
Message(1).field	Message(2).field	Message(3).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 upper- and lowercase letters as defined. 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 a variable in the MATLAB workspace and pass the variable name to the driver. For example, to initialize an external A/D module and acquire a value during each sample interval, create a script 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 script containing the definition of the message structure has to be executed before the model is opened.

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

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

- When you move or copy the model file to a new directory, you also need to move or copy the script 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 topics:

- “RS-232/422/485 Simulink Block Reference” on page 4-14
- “GPIB MATLAB Structure Reference” on page 6-13
- “Initialization and Termination CAN Messages” on page 7-83
- 

## **xPC Target Support and SimState**

You can save complete model simulation states while simulating, on a host computer, a Simulink model that contains some xPC Target blocks. The software does not support this behavior when executing such a model on the target computer.

For this operation, set the **Save complete SimState in final state** check box in the **Data Import/Export** pane of the Configuration Parameters dialog box. If your model contains the following blocks, you cannot save complete model simulation states while simulating on the host computer.

- ASCII Encode
- ASCII Decode
- Async Buffer Read
- Async Buffer Write
- Baseboard Serial
- Baseboard Serial F
- Bit Packing (Utilities library)
- Bit Unpacking (Utilities library)
- Byte Packing (Utilities library)
- Byte Unpacking (Utilities library)

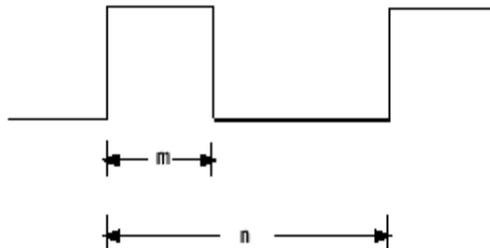
- Commtech Fastcom 422/2-PCI
- Commtech Fastcom 422/2-PCI F
- Commtech Fastcom 422/2-PCI-335
- Commtech Fastcom 422/2-PCI-335 F
- Commtech Fastcom 422/4-PCI-335
- Commtech Fastcom 422/4-PCI-335 F
- Condor 1553 BC List
- Create Ethernet Packet (Ethernet library)
- Diamond Systems Emerald-MM Serial
- Diamond Systems Emerald-MM Serial F
- Diamond Systems Emerald-MM8 Serial
- Diamond Systems Emerald-MM8 Serial F
- FIFO bin read
- FIFO ASCII read
- FIFO write
- Quatech DSCP-200/300
- Quatech DSCP-200/300 F
- Quatech ESC-100
- Quatech ESC-100 F
- Quatech QSC-100
- Quatech QSC-100 F
- Quatech QSC-200/300
- Quatech QSC-200/300 F
- UDP Receive
- UDP Send

To prevent these messages, clear the **Save complete SimState in final state** check box in the **Data Import/Export** node of the Configuration Parameters dialog box.

## 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.

## Driver Block Documentation

The typical xPC Target block documentation briefly describes the supported board, then describes the parameters for each of the blocks that support the board. Included in the documentation for each board is a board characteristics table. Board characteristics tables can include the following information:

Characteristic	Specifies...
Board name	Name of the board supported by the blocks. For example, National Instruments® PCI-6221.
Manufacturer	Manufacturer of the board. For example, National Instruments.
Bus type	Bus that is used by the board. For example, PCI or ISA.
Access method	Whether the board is memory mapped or I/O mapped.
Multiple block instance support	If you can use multiple blocks for the same function on the same board. For example, different blocks for different channels of an A/D device.
Multiple board support	If you can use multiple boards of the same type in one target application.

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

The xPC Target product 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` 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.

- 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.

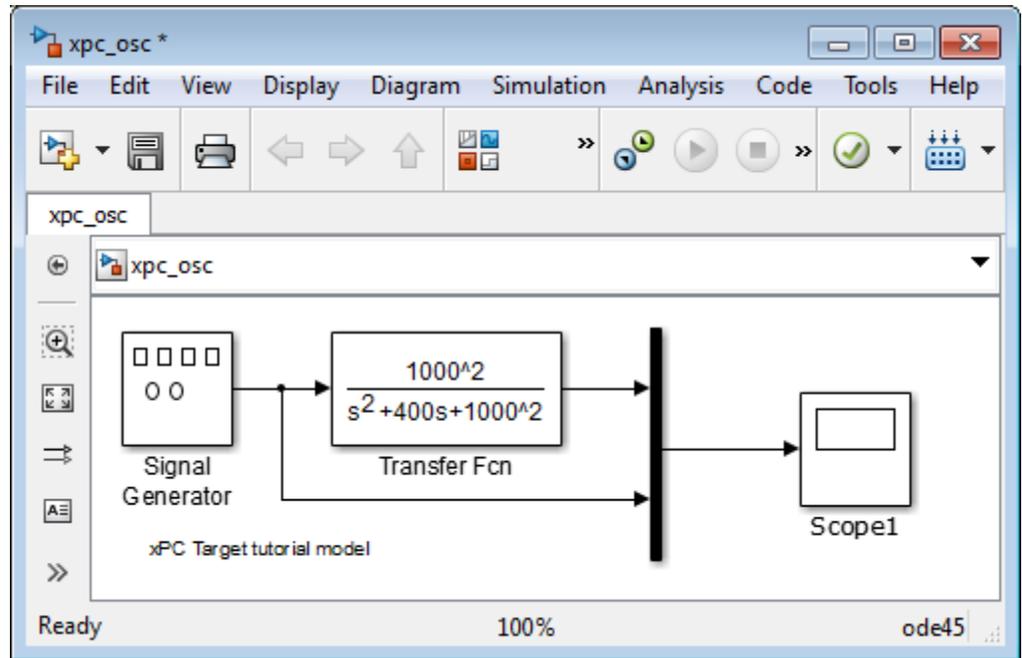
- 3** Open a manufacturer group. For example, to open the A/D driver blocks from Measurement Computing, double-click the group marked Measurement Computing.

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`.

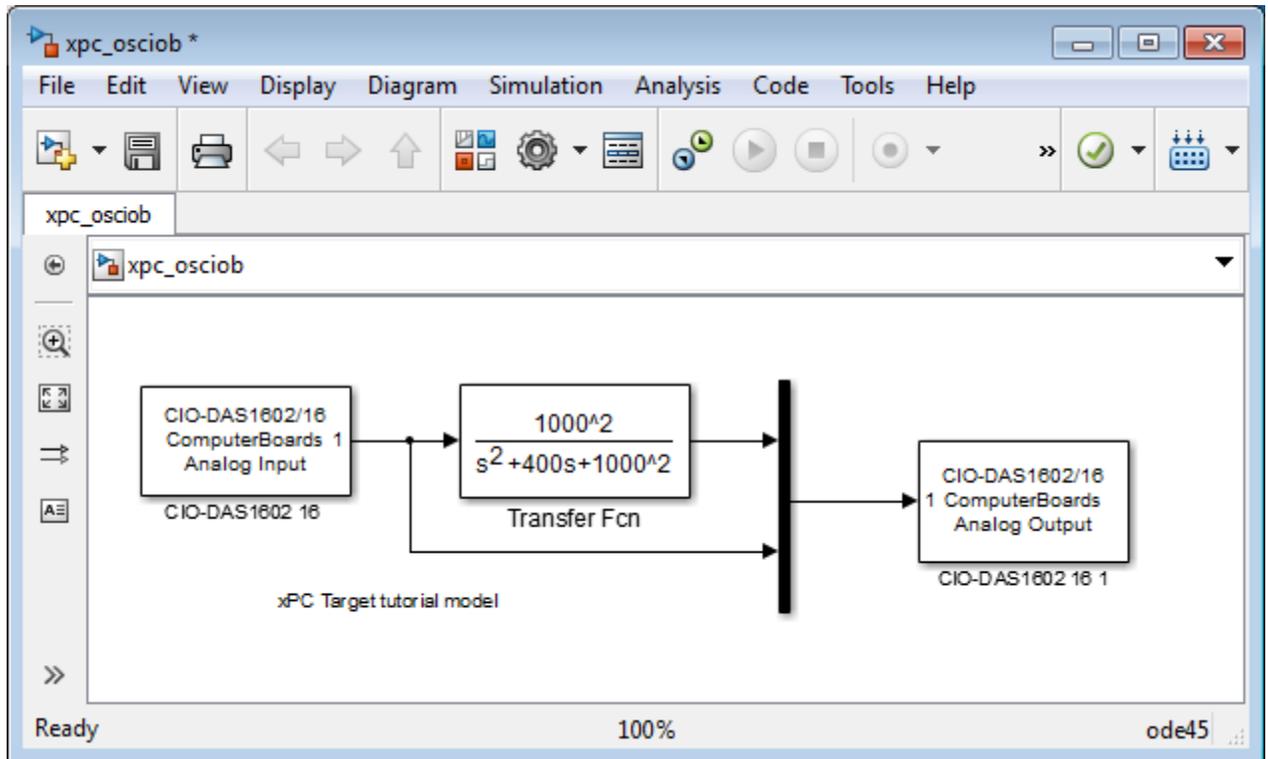


- 5 From the block library, click and drag the name of an A/D board to the Simulink block diagram. Likewise, click and drag the name of a D/A board to your model.

The Simulink software 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 example model xpc\_osc looks like this:



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

Your next task is to define the I/O block parameters. See “Defining I/O Block Parameters” on page 1-18.

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

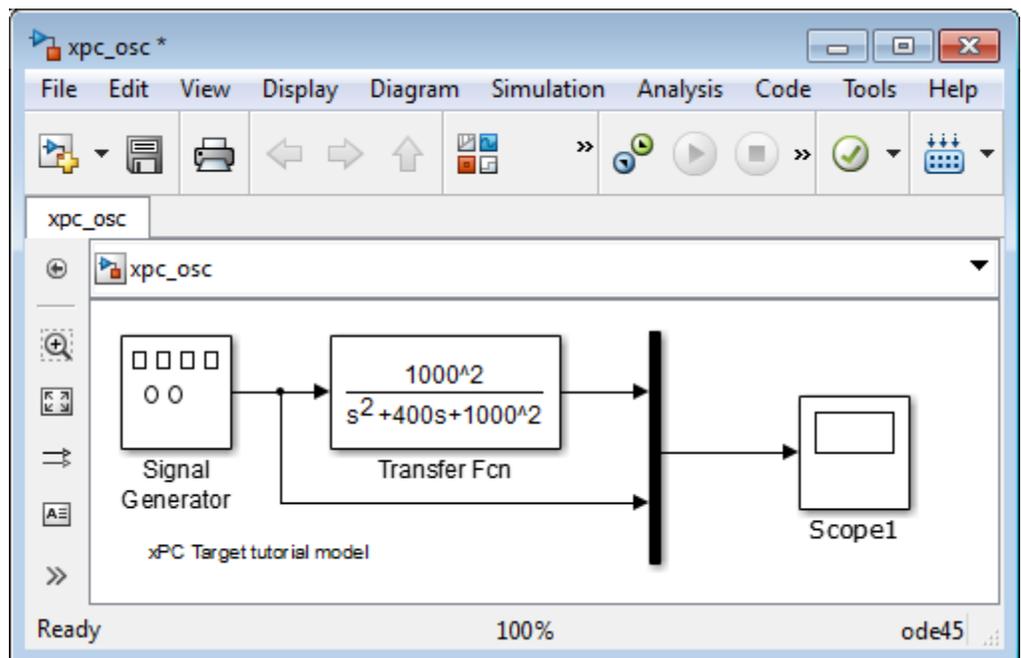
The xPC Target product includes a Simulink block library for I/O drivers. The highest hierarchical level in the library is grouped by I/O function. The second level is grouped by board manufacturer. The manufacturer groups within this second level contain the driver blocks for specific boards.

This procedure uses the Simulink model `xpc_osc` 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`.



- 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 1-12.

**3** Double-click **xPC Target**.

A list of I/O block function groups 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.

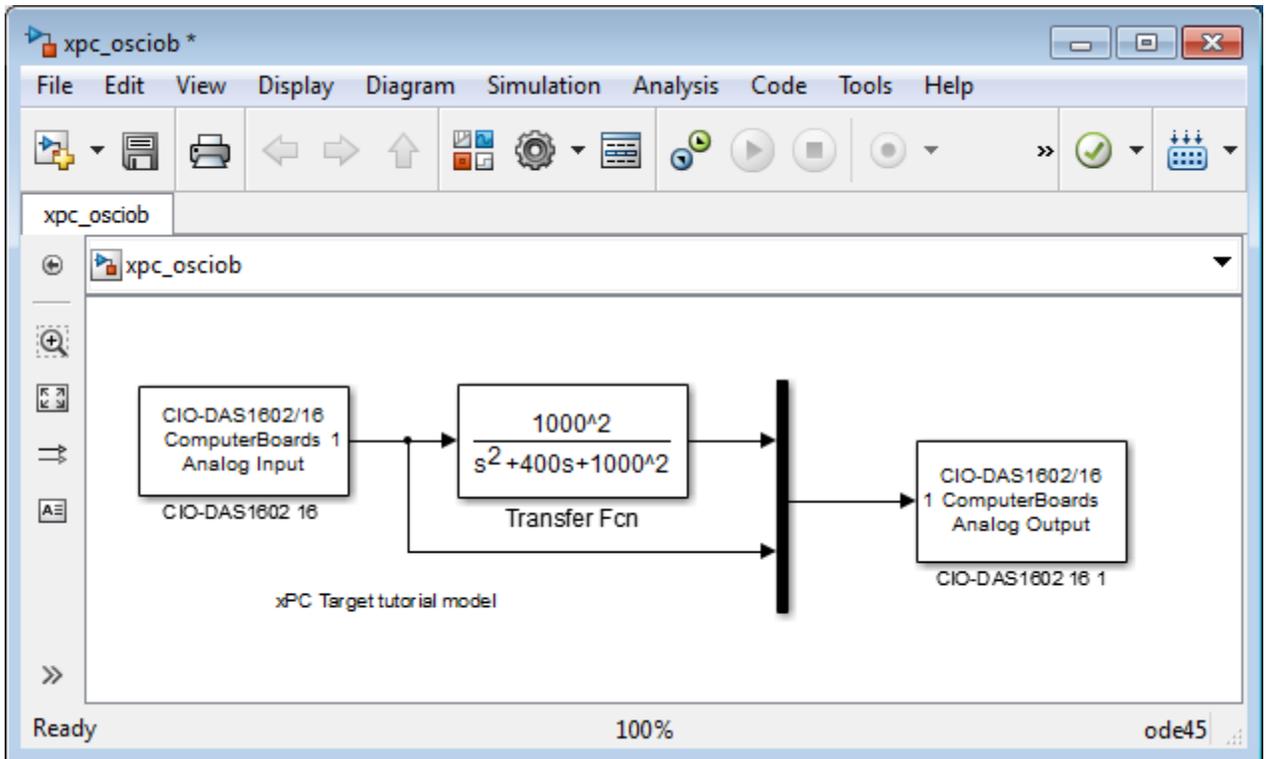
A list with the A/D driver blocks for Measurement Computing opens.

**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.

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

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

The model `xpc_osc` looks like this:



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

Your next task is to define the I/O block parameters. See “Defining I/O Block Parameters” on page 1-18

## 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.

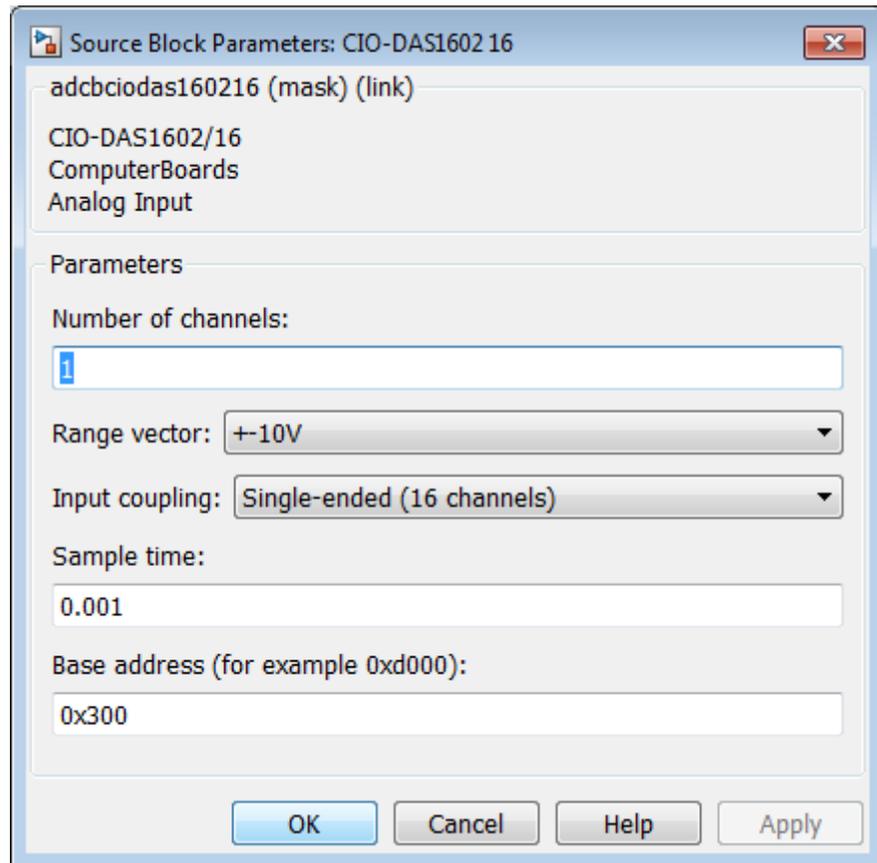
This procedure uses the Simulink model `xpc_osc` as an example, and assumes you have added an analog input and an analog output block to your model. To add an I/O block, see either “Adding I/O Blocks with the xPC Target Library” on page 1-12 or “Adding I/O Blocks with the Simulink Library Browser” on 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 **Solver** pane of the **Simulation > Model Configuration Parameters** dialog box. Enter the base address for this ISA-bus board.

The Block Parameters dialog box looks like this:

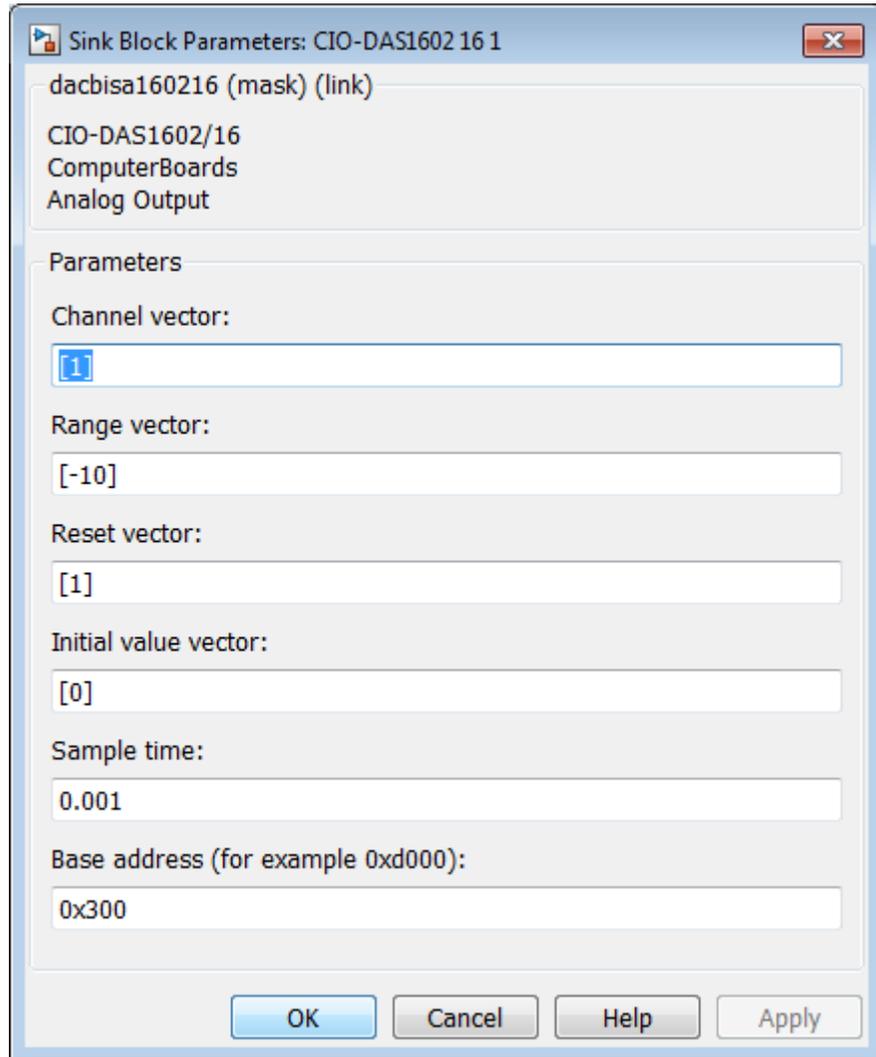


- 3 In the Simulink window, double-click the output block labeled **Analog Output**.

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 **Solver** pane of the **Simulation > Model Configuration Parameters** dialog box. Enter the base address for this ISA-bus board.

The Block Parameters dialog box looks like this:



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 execute the target application in real time. Start with “Boot Target Hardware”.



# Blocks — By Category

---

A/D Converter (p. 2-4)	Support for analog-digital converters.
A/D Frame Converter (p. 2-11)	Support for analog-digital frame converters.
ARINC 429 (p. 2-12)	Support for the ARINC 429 avionics standard.
Asynchronous Event (p. 2-13)	Support for asynchronous event blocks.
Audio (p. 2-14)	Support for audio waveform processing.
Controller Area Network (CAN) Protocols (p. 2-15)	Support for Controller Area Network-based protocols.
Counter (p. 2-18)	Support for digital counter blocks.
D/A Converter (p. 2-20)	Support for digital-analog converters.
Data Marshalling (p. 2-28)	Support for data marshalling.
Digital Input (p. 2-29)	Support for digital inputs.
Digital Output (p. 2-39)	Support for digital outputs.
Digital Signal Processing (p. 2-49)	Support for digital signal processing
Ethernet Protocols (p. 2-50)	Support for Ethernet-based protocols.
Field Programmable Gate Arrays (FPGA) (p. 2-52)	Support for FPGAs.

General Purpose Interface Bus (GPIB) (p. 2-53)	Support for the GPIB standard.
Incremental Encoder (p. 2-54)	Support for incremental digital encoder.
I/O Pack (IP) Carrier (p. 2-56)	Support for physically plugging an interface module into a carrier board.
J1939 (CAN) (p. 2-57)	Support for the J1939 network standard running over CAN.
Light Emitting Diodes (LED) (p. 2-58)	Support for LED blocks.
Linear Variable Differential Transformer (LVDT) (p. 2-59)	Support for linear displacement sensor blocks.
MIL-STD-1553 (p. 2-60)	Support for the MIL-STD-1553 bus with the MIL-STD-1553 sublibrary
Miscellaneous (p. 2-62)	Support for miscellaneous blocks.
Parallel Port (p. 2-63)	Support for parallel port blocks.
Pulse Width Measurement (p. 2-64)	Support for pulse width measurement blocks.
Real-Time Performance (p. 2-67)	Support for real-time performance blocks.
RS-232 (p. 2-68)	Support for RS232 serial standard blocks.
Shared Memory (p. 2-69)	Support for shared memory blocks.
Signal Conditioning (p. 2-70)	Support for signal conditioning blocks.
Synchro Resolver (p. 2-71)	Support for angular position/velocity sensor blocks.
Thermocouple (p. 2-72)	Support for thermocouple blocks.
User Datagram Protocol (UDP) Protocols (p. 2-73)	Support for protocols based on the User Datagram Protocol (UDP)
Utility (p. 2-75)	Support for utility blocks.

---

Video (p. 2-76)	Support for video blocks.
Visualization (p. 2-77)	Support for scope visualization blocks.
XCP (CAN) (p. 2-78)	Support for the XCP network standard running over CAN.
Watchdog Timer (p. 2-79)	Support for watchdog timer blocks.

## A/D Converter

Advantech PCL-1800 Analog Input (A/D)	PCL-1800 Analog Input block
Advantech PCL-711B Analog Input (A/D)	PCL-711B Analog Input block
Advantech PCL-812 Analog Input (A/D)	PCL-812 Analog Input block
Advantech PCL-812PG Analog Input (A/D)	PCI-812PG Analog Input block
Advantech PCL-818 Analog Input (A/D)	PCL-818 Analog Input (A/D)
Advantech PCL-818H Analog Input (A/D)	PCL-818H Analog Input block
Advantech PCL-818HD Analog Input (A/D)	PCL-818HD Analog Input block
Advantech PCL-818HG Analog Input (A/D)	PCL-818HG Analog Input block
Advantech PCL-818L Analog Input (A/D)	PCL-818L Analog Input block
Analogic AIM12 Analog Input (A/D)	Analogic® AIM12 Analog Input block
Analogic AIM16 Analog Input (A/D)	AIM16 Analog Input block
Contec AD12-16(PCI) Analog Input (A/D)	AD12-16(PCI) Analog Input (A/D) block
Contec AD12-16(PCI)E Analog Input (A/D)	AD12-16(PCI)E Analog Input block
Contec AD12-16U(PCI)E Analog Input (A/D)	AD12-16U(PCI)E Analog Input block
Contec AD12-64(PCI) Analog Input (A/D)	AD12-64(PCI) Analog Input block
Contec AD16-16(PCI)E Analog Input (A/D)	AD16-16(PCI)E Analog Input block

Contec ADI12-16(PCI) Analog Input (A/D)	ADI12-16(PCI) Analog Input block
Data Translation DT2821 Analog Input (A/D)	DT2821 Analog Input block
Data Translation DT2821-F-16SE Analog Input (A/D)	DT2821-F-16SE Analog Input block
Data Translation DT2821-F-8DI Analog Input (A/D)	DT2821-F-8DI Analog Input block
Data Translation DT2821-G-16SE Analog Input (A/D)	DT2821-G-16SE Analog Input block
Data Translation DT2821-G-8DI Analog Input (A/D)	DT2821-G-8DI Analog Input block
Data Translation DT2823 Analog Input (A/D)	DT2823 Analog Input block
Data Translation DT2824-PGH Analog Input (A/D)	DT2824-PGH Analog Input block
Data Translation DT2824-PGL Analog Input (A/D)	DT2824-PGL Analog Input block
Data Translation DT2825 Analog Input (A/D)	DT2825 Analog Input block
Data Translation DT2827 Analog Input (A/D)	DT2827 Analog Input block
Data Translation DT2828 Analog Input (A/D)	DT2828 Analog Input block
Diamond MM Analog Input (A/D)	MM Analog Input block
Diamond MM-16-AT Analog Input (A/D)	MM-16-AT Analog Input block
Diamond MM-32-AT Analog Input (A/D)	MM-32-AT Analog Input block
Diamond Prometheus/Athena/Athena II Analog Input (A/D)	Prometheus/Athena Analog Input block

General Standards Create Enable Signal	Create Enable Signal block
General Standards PMC-ADADIO Analog Input (A/D) Read	PMC-ADADIO Analog Input Read block
General Standards PMC-ADADIO Analog Input (A/D) Start	PMC-ADADIO Analog Input Start block
Humusoft AD 512 Analog Input (A/D)	HUMUSOFT® AD 512 Analog Input block
Humusoft AD622 Analog Input (A/D)	HUMUSOFT AD622 Analog Input block
Humusoft MF624 Analog Input (A/D)	HUMUSOFT MF624 Analog Input block
Keithley DAS-1800HR Analog Input (A/D)	DAS-1800HR Analog Input block
Keithley KPCI-1801HC Analog Input (A/D)	KPCI-1801HC Analog Input block
Keithley KPCI-1802HC Analog Input (A/D)	KPCI-1802HC Analog Input block
Measurement Computing CIO-DAS16/330 Analog Input (A/D)	CIO-DAS16/330 Analog Input (A/D) block
Measurement Computing CIO-DAS16/JR (/12) Analog Input (A/D) with EXP	CIO-DAS16/JR (/12) Analog Input with EXP Signal Conditioning Board block
Measurement Computing CIO-DAS16/JR Analog Input (A/D)	CIO-DAS16/JR Analog Input (A/D) block
Measurement Computing CIO-DAS1601/12 Analog Input (A/D)	CIO-DAS1601/12 Analog Input block
Measurement Computing CIO-DAS1602/12 Analog Input (A/D)	CIO-DAS1602/12 Analog Input block

Measurement Computing CIO-DAS1602/16 Analog Input (A/D)	CIO-DAS1602/16 Analog Input block
Measurement Computing CIO-DAS16JR/16 Analog Input (A/D)	CIO-DAS16JR/16 Analog Input block
Measurement Computing PC104-DAS16JR/12 Analog Input (A/D)	PC104-DAS16JR/12 Analog Input block
Measurement Computing PC104-DAS16JR/16 Analog Input (A/D)	PC104-DAS16JR/16 Analog Input block
Measurement Computing PCI-DAS1002 Analog Input (A/D)	PCI-DAS1002 Analog Input block
Measurement Computing PCI-DAS1200 Analog Input (A/D)	PCI-DAS1200 Analog Input block
Measurement Computing PCI-DAS1200/JR Analog Input (A/D)	PCI-DAS1200/JR Analog Input block
Measurement Computing PCI-DAS1602/12 Analog Input (A/D)	PCI-DAS1602/12 Analog Input block
Measurement Computing PCI-DAS1602/16 Analog Input (A/D)	PCI-DAS1602/16 Analog Input block
Measurement Computing PCIM-DAS1602/16 Analog Input (A/D)	PCIM-DAS1602/16 Analog Input block
National Instruments PCI-6011E Analog Input (A/D) (formerly PCI-MIO-16XE-50)	PCI-6011E Analog Input block

National Instruments PCI-6023E Analog Input (A/D)	PCI-6023E Analog Input block
National Instruments PCI-6024E Analog Input (A/D)	PCI-6024E Analog Input block
National Instruments PCI-6025E Analog Input (A/D)	PCI-6025E Analog Input block
National Instruments PCI-6030E Analog Input (A/D) (formerly PCI-MIO-16XE-10)	PCI-6030E Analog Input block
National Instruments PCI-6031E Analog Input (A/D)	PCI-6031E Analog Input block
National Instruments PCI-6040E Analog Input (A/D) (formerly PCI-MIO-16E-4)	PCI-6040E Analog Input block
National Instruments PCI-6070E Analog Input (A/D) (formerly PCI-MIO-16E-1)	PCI-6070E Analog Input block
National Instruments PCI-6071E Analog Input (A/D)	PCI-6071E Analog Input (A/D)
National Instruments PCI-6221 Analog Input	National Instruments PCI-6221 Analog Input block
National Instruments PCI-6221/37 Analog Input	National Instruments PCI-6221/37 Analog Input block
National Instruments PCI-6225 Analog Input	National Instruments PCI-6225 Analog Input block
National Instruments PCI-6229 Analog Input	National Instruments PCI-6229 Analog Input block
National Instruments PCI-6251 Analog Input	National Instruments PCI-6251 Analog Input block
National Instruments PCI-6259 Analog Input	National Instruments PCI-6259 Analog Input block
National Instruments PCI-6280 Analog Input	National Instruments PCI-6280 Analog Input block

National Instruments PCI-6281 Analog Input	National Instruments PCI-6281 Analog Input block
National Instruments PCI-6289 Analog Input	National Instruments PCI-6289 Analog Input block
National Instruments PXI-6040E Analog Input (A/D)	PXI-6040E Analog Input block
National Instruments PXI-6070E Analog Input (A/D)	PXI-6070E Analog Input block
National Instruments PXI-6071E Analog Input (A/D)	PXI-6071E Analog Input block
National Instruments PXI/PCI-6052E Analog Input (A/D)	PXI/PCI-6052E Analog Input block
Quanser Q4 Analog Input	Q4 Analog Input block
Quanser Q8 Analog Input	Q8 Analog Input block
Real Time Devices DM6430 Analog Input (A/D)	DM6430 Analog Input block
Real Time Devices DM7420 Analog Input (A/D)	DM7420 Analog Input block
SBS IP-16ADC Analog Input (A/D)	IP-16ADC Analog Input block
SBS IP-HiADC Analog Input (A/D)	IP-HiADC Analog Input block
Sensoray526 AD	Sensoray526 AD block
Sensoray526 Dual AD	Sensoray526 Dual AD block
UEI PD2-MF 12-Bit Series Analog Input (A/D)	PD2-MF 12-Bit Series Analog Input block
UEI PD2-MF 14-Bit Series Analog Input (A/D)	PD2-MF 14-Bit Series Analog Input block
UEI PD2-MF 16-Bit Series Analog Input (A/D)	PD2-MF 16-Bit Series Analog Input block
UEI PD2-MFS 12-Bit Series Analog Input (A/D)	PD2-MFS 12-Bit Series Analog Input block

UEI PD2-MFS 14-Bit Series Analog Input (A/D)	PD2-MFS 14-Bit Series Analog Input block
UEI PD2-MFS 16-Bit Series Analog Input (A/D)	PD2-MFS 16-Bit Series Analog Input block
UEI PDXI-MF 14-Bit Series Analog Input (A/D)	PDXI-MF 14-Bit Series Analog Input block
UEI PDXI-MF 16-Bit Series Analog Input (A/D)	PDXI-MF 16-Bit Series Analog Input block
UEI PDXI-MFS 12-Bit Series Analog Input (A/D)	PDXI-MFS 12-Bit Series Analog Input block
UEI PDXI-MFS 14-Bit Series Analog Input (A/D)	PDXI-MFS 14-Bit Series Analog Input block
UEI PDXI-MFS 16-Bit Series Analog Input (A/D)	PDXI-MFS 16-Bit Series Analog Input block
Versallogic VSBC-6 Analog Input (A/D)	VSBC-6 Analog Input block

## A/D Frame Converter

Diamond MM-32-AT Frame Analog Input (A/D)	MM-32-AT Frame Analog Input block
UEI PD2-MF 12-Bit Series Frame Analog Input	PD2-MF 12-Bit Series Frame Analog Input block
UEI PD2-MF 14-Bit Series Frame Analog Input	PD2-MF 14-Bit Series Frame Analog Input block
UEI PD2-MF 16-Bit Series Frame Analog Input	PD2-MF 16-Bit Series Frame Analog Input block
UEI PD2-MFS 12-Bit Series Frame Analog Input	PD2-MFS 12-Bit Series Frame Analog Input block
UEI PD2-MFS 14-Bit Series Frame Analog Input	PD2-MFS 14-Bit Series Frame Analog Input block
UEI PD2-MFS 16-Bit Series Frame Analog Input	PD2-MFS 16-Bit Series Frame Analog Input block
UEI PDXI-MF 12-Bit Series Frame Analog Input	PDXI-MF 12-Bit Series Frame Analog Input block
UEI PDXI-MF 14-Bit Series Frame Analog Input	PDXI-MF 14-Bit Series Frame Analog Input block
UEI PDXI-MF 16-Bit Series Frame Analog Input	PDXI-MF 16-Bit Series Frame Analog Input block
UEI PDXI-MFS 12-Bit Series Frame Analog Input	PDXI-MFS 12-Bit Series Frame Analog Input block
UEI PDXI-MFS 14-Bit Series Frame Analog Input	PDXI-MFS 14-Bit Series Frame Analog Input block
UEI PDXI-MFS 16-Bit Series Frame Analog Input	PDXI-MFS 16-Bit Series Frame Analog Input block

## **ARINC 429**

Condor CEI-520/520A

Condor CEI-x20 Initialize

Condor CEI-x20 Receive

Condor CEI-x20 Send

Condor Decode ARINC 429 Words  
from Receive

Condor Encode ARINC 429 Words  
for Send

Support for ARINC 429  
CEI-520/5201A board.

Condor CEI-x20 Initialize block

Condor CEI-x20 Receive block

Condor CEI-x20 Send block

Decode ARINC 429 Words from  
Receive block

Encode ARINC 429 Words for Send  
block

## **Asynchronous Event**

Async Buffer Write and Read

Async IRQ Source

Async Rate Transition

Async Buffer Write and Read blocks

Async IRQ Source block

Async Rate Transition block

## **Audio**

BittWare Audio-PMC+ Analog Input	Audio-PMC+ Analog Input block
BittWare Audio-PMC+ Analog Output	Audio-PMC+ Analog Output block
General Standards 16AO16 Analog Output	General Standards 16AO16 Analog Output
General Standards 24DSI12 Analog Input	General Standards 24DSI12 Analog Input

## Controller Area Network (CAN) Protocols

### In this section...

“CAN (DOB)” on page 2-15

“CAN (FIFO)” on page 2-16

### CAN (DOB)

	Softing® CAN-AC2-PCI with SJA1000 Setup block
CAN Bit-Packing	CAN Bit-Packing block
CAN Bit-Unpacking	CAN Bit-Unpacking block
CAN Timeout Detection	CAN Timeout Detection block
Softing CAN-AC2-104 with SJA1000 Receive	Softing CAN-AC2-104 with SJA1000 Receive block
Softing CAN-AC2-104 with SJA1000 Send	Softing CAN-AC2-104 with SJA1000 Send block
Softing CAN-AC2-104 with SJA1000 Setup	Softing CAN-AC2-104 with SJA1000 Setup block
Softing CAN-AC2-ISA with Intel 82527 Receive	Softing CAN-AC2-ISA with Intel® 82527 Receive block
Softing CAN-AC2-ISA with Intel 82527 Send	Softing CAN-AC2-ISA with Intel 82527 Send block
Softing CAN-AC2-ISA with Intel 82527 Setup	Softing CAN-AC2-ISA with Intel 82527 Setup block
Softing CAN-AC2-ISA with Philips PCA 82C200 Receive	Softing CAN-AC2-ISA with Philips PCA 82C200 Receive block
Softing CAN-AC2-ISA with Philips PCA 82C200 Send	Softing CAN-AC2-ISA with Philips PCA 82C200 Send block
Softing CAN-AC2-ISA with Philips PCA 82C200 Setup	Softing CAN-AC2-ISA with Philips PCA 82C200 Setup block

Softing CAN-AC2-PCI with SJA1000  
Receive

Softing CAN-AC2-PCI with SJA1000  
Receive block

Softing CAN-AC2-PCI with SJA1000  
Send

Softing CAN-AC2-PCI with SJA1000  
Send block

## **CAN (FIFO)**

Softing CAN-AC2-104 with Philips  
SJA1000 FIFO Read

Softing CAN-AC2-104 with Philips  
SJA1000 FIFO Read block

Softing CAN-AC2-104 with Philips  
SJA1000 FIFO Read Filter

Softing CAN-AC2-104 with Philips  
SJA1000 FIFO Read Filter Block

Softing CAN-AC2-104 with Philips  
SJA1000 FIFO Read RCV Level

Softing CAN-AC2-104 with Philips  
SJA1000 FIFO Read RCV Level  
block

Softing CAN-AC2-104 with Philips  
SJA1000 FIFO Read XMT Level

Softing CAN-AC2-104 with Philips  
SJA1000 FIFO Read XMT Level  
block

Softing CAN-AC2-104 with Philips  
SJA1000 FIFO Reset RCV

Softing CAN-AC2-104 with Philips  
SJA1000 FIFO Reset RCV block

Softing CAN-AC2-104 with Philips  
SJA1000 FIFO Reset XMT

Softing CAN-AC2-104 with Philips  
SJA1000 FIFO Reset XMT block

Softing CAN-AC2-104 with Philips  
SJA1000 FIFO Setup

Softing CAN-AC2-104 with Philips  
SJA1000 FIFO Setup block

Softing CAN-AC2-104 with Philips  
SJA1000 FIFO Write

Softing CAN-AC2-104 with Philips  
SJA1000 FIFO Write block

Softing CAN-AC2-PCI with Philips  
SJA1000 FIFO Read

Softing CAN-AC2-PCI with Philips  
SJA1000 FIFO Read block

Softing CAN-AC2-PCI with Philips  
SJA1000 FIFO Read Filter

Softing CAN-AC2-PCI with Philips  
SJA1000 FIFO Read Filter block

Softing CAN-AC2-PCI with Philips  
SJA1000 FIFO Read RCV Level

Softing CAN-AC2-PCI with Philips  
SJA1000 FIFO Read RCV Level  
block

Softing CAN-AC2-PCI with Philips  
SJA1000 FIFO Read XMT Level

Softing CAN-AC2-PCI with Philips  
SJA1000 FIFO Read XMT Level  
block

Softing CAN-AC2-PCI with Philips  
SJA1000 FIFO Reset RCV

Softing CAN-AC2-PCI with Philips  
SJA1000 FIFO Reset RCV block

Softing CAN-AC2-PCI with Philips  
SJA1000 FIFO Reset XMT

Softing CAN-AC2-PCI with Philips  
SJA1000 FIFO Reset XMT block

Softing CAN-AC2-PCI with Philips  
SJA1000 FIFO Setup

Softing CAN-AC2-PCI with Philips  
SJA1000 FIFO Setup block

Softing CAN-AC2-PCI with Philips  
SJA1000 FIFO Write

Softing CAN-AC2-PCI with Philips  
SJA1000 FIFO Write Driver block

## Counter

Humusoft MF624 Counter Input	HUMUSOFT MF624 Counter Input block
MPL PATI Timebase Setup	MPL PATI Timebase Setup block
National Instruments PCI-6011E Pulse Generation (formerly PCI-MIO-16XE-50)	PCI-6011E Pulse Generation block
National Instruments PCI-6023E Pulse Generation	PCI-6023E Pulse Generation
National Instruments PCI-6024E Pulse Generation	PCI-6024E Pulse Generation
National Instruments PCI-6025E Pulse Generation	PCI-6025E Pulse Generation block
National Instruments PCI-6030E Pulse Generation (formerly PCI-MIO-16XE-10)	PCI-6030E Pulse Generation block
National Instruments PCI-6031E Pulse Generation	PCI-6031E Pulse Generation
National Instruments PCI-6040E Pulse Generation (formerly PCI-MIO-16E-4)	PCI-6040E Pulse Generation block
National Instruments PCI-6070E Pulse Generation (formerly PCI-MIO-16E-1)	PCI-6070E Pulse Generation
National Instruments PCI-6071E Pulse Generation	PCI-6071E Pulse Generation block
National Instruments PCI-6601 Armed Pulse Generation	PCI-6601 Armed Pulse Generation block
National Instruments PCI-6601 Event Counter	PCI-6601 Event Counter block
National Instruments PCI-6601 Pulse Generation	PCI-6601 Pulse Generation block

---

National Instruments PCI/PXI-6602 Armed Pulse Generation	PCI/PXI-6602 Armed Pulse Generation block
National Instruments PCI/PXI-6602 Event Counter	PCI-6602 Event Counter block
National Instruments PCI/PXI-6602 Pulse Generation	PCI/PXI-6602 Pulse Generation block
National Instruments PCI/PXI-6711 Pulse Generation	PCI/PXI-6711 Pulse Generation block
National Instruments PCI/PXI-6713 Pulse Generation	PCI/PXI-6713 Pulse Generation block
National Instruments PCI/PXI-6731 Pulse Generation	PCI/PXI-6731 Pulse Generation block
National Instruments PCI/PXI-6733 Pulse Generation	PCI/PXI-6733 Pulse Generation block
National Instruments PXI-6040E Pulse Generation	PXI-6040E Pulse Generation block
National Instruments PXI-6070E Pulse Generation	PXI-6070E Pulse Generation block
National Instruments PXI-6071E Pulse Generation	PXI-6071E Pulse Generation block
National Instruments PXI/PCI-6052E Pulse Generation	PXI/PCI-6052E Pulse Generation
Quanser Q4 Counter	Q4 Counter block
Quanser Q8 Counter	Q8 Counter block

## D/A Converter

Adlink PCI-6208A Analog Output	Adlink PCI-6208A Analog Output block
Advantech PCL-1800 Analog Output (D/A)	PCL-1800 Analog Output block
Advantech PCL-711B Analog Output (D/A)	PCL-711B Analog Output block
Advantech PCL-726 Analog Output (D/A)	PCL-726 Analog Output block
Advantech PCL-727 Analog Output (D/A)	PCL-727 Analog Output block
Advantech PCL-728 Analog Output (D/A)	PCL-728 Analog Output block
Advantech PCL-812 Analog Output (D/A)	PCL-812 Analog Output block
Advantech PCL-812PG Analog Output (D/A)	PCL-812PG Analog Output block
Advantech PCL-818 Analog Output (D/A)	PCL-818 Analog Output block
Advantech PCL-818H Analog Output (D/A)	PCL-818H Analog Output block
Advantech PCL-818HD Analog Output (D/A)	PCL-818HD Analog Output block
Advantech PCL-818HG Analog Output (D/A)	PCL-818HG Analog Output block
Advantech PCL-818L Analog Output (D/A)	PCL-818L Analog Output block
Contec AD12-16(PCI)E Analog Output (D/A)	AD12-16(PCI)E Analog Output block
Contec AD12-16U(PCI)E Analog Output (D/A)	AD12-16U(PCI)E Analog Output block

Contec AD16-16(PCI)E Analog Output (D/A)	AD16-16(PCI)E Analog Output block
Contec DA12-16(PCI) Analog Output (D/A)	DA12-16(PCI) Analog Output block
Contec DA12-4(PCI) Analog Output (D/A)	DA12-4(PCI) Analog Output block
Data Translation DT2821 Analog Output (D/A)	DT2821 Analog Output block
Data Translation DT2821-F-16SE Analog Output (D/A)	DT2821-F-16SE Analog Output block
Data Translation DT2821-F-8DI Analog Output (D/A)	DT2821-F-8DI Analog Output block
Data Translation DT2821-G-16SE Analog Output (D/A)	DT2821-G-16SE Analog Output block
Data Translation DT2821-G-8DI Analog Output (D/A)	DT2821-G-8DI Analog Output block
Data Translation DT2823 Analog Output (D/A)	DT2823 Analog Output block
Data Translation DT2825 Analog Output (D/A)	DT2825 Analog Output block
Data Translation DT2827 Analog Output (D/A)	DT2827 Analog Output block
Data Translation DT2828 Analog Output (D/A)	DT2828 Analog Output block
Diamond MM Analog Output (D/A)	MM Analog Output block
Diamond MM-16-AT Analog Output (D/A)	MM-16-AT Analog Output block
Diamond MM-32-AT Analog Output (D/A)	MM-32-AT Analog Output block
Diamond Prometheus/Athena/Athena II Analog Output (D/A)	Prometheus/Athena Analog Output block

Diamond Ruby-MM Analog Output (D/A)	Ruby-MM Analog Output block
Diamond Ruby-MM-1612 Analog Output (D/A)	Ruby-MM-1612 Analog Output block
Diamond Ruby-MM-416 Analog Output (D/A)	Ruby-MM-416 Analog Output block
General Standards Create Enable Signal	Create Enable Signal block
General Standards PMC-16AO-12 Analog Output	PMC-16AO-12 Analog Output block
General Standards PMC-ADADIO Analog Output (D/A) Update	PMC-ADADIO Analog Output Update block
General Standards PMC-ADADIO Analog Output Write Block	PMC-ADADIO Analog Output Write block
Humusoft AD 512 Analog Output (D/A)	HUMUSOFT AD 512 Analog Output block
Humusoft MF624 Analog Output (D/A)	HUMUSOFT MF624 Analog Output block
Keithley KPCI-1801HC Analog Output (D/A)	KPCI-1801HC Analog Output block
Keithley KPCI-1802HC Analog Output (D/A)	KPCI-1802HC Analog Output block
Measurement Computing CIO-DAC08 (/12) Analog Output (D/A)	CIO-DAC08 (/12) Analog Output (D/A) block
Measurement Computing CIO-DAC08/16 Analog Output (D/A)	CIO-DAC08/16 Analog Output (D/A) block
Measurement Computing CIO-DAC16 Analog Output (D/A)	CIO-DAC16 Analog Output (D/A) block
Measurement Computing CIO-DAC16/16 Analog Output (D/A)	CIO-DAC16/16 Analog Output (D/A) block

Measurement Computing CIO-DAS1601/12 Analog Output (D/A)	CIO-DAS1601/12 Analog Output block
Measurement Computing CIO-DAS1602/12 Analog Output (D/A)	CIO-DAS1602/12 Analog Output block
Measurement Computing CIO-DAS1602/16 Analog Output (D/A)	CIO-DAS1602/16 Analog Output block
Measurement Computing CIO-DDA06 (/12) Analog Output (D/A)	CIO-DDA06 (/12) Analog Output block
Measurement Computing CIO-DDA06/16 Analog Output (D/A)	CIO-DDA06/16 Analog Output block
Measurement Computing PC104-DAC06 (/12) Analog Output (D/A)	PC104-DAC06 (/12) Analog Output block
Measurement Computing PCI-DAC6703 Analog Output (D/A)	PCI-DAC6703 Analog Output block
Measurement Computing PCI-DAS1002 Analog Output (D/A)	PCI-DAS1002 Analog Output block
Measurement Computing PCI-DAS1200 Analog Output (D/A)	PCI-DAS1200 Analog Output block
Measurement Computing PCI-DAS1602/12 Analog Output (D/A)	PCI-DAS1602/12 Analog Output block
Measurement Computing PCI-DAS1602/16 Analog Output (D/A)	PCI-DAS1602/16 Analog Output block

Measurement Computing PCI-DDA02/12 Analog Output (D/A)	PCI-DDA02/12 Analog Output block
Measurement Computing PCI-DDA02/16 Analog Output (D/A)	PCI-DDA02/16 Analog Output block
Measurement Computing PCI-DDA04/12 Analog Output (D/A)	PCI-DDA04/12 Analog Output block
Measurement Computing PCI-DDA04/16 Analog Output (D/A)	PCI-DDA04/16 Analog Output block
Measurement Computing PCI-DDA08/12 Analog Output (D/A)	PCI-DDA08/12 Analog Output block
Measurement Computing PCI-DDA08/16 Analog Output (D/A)	PCI-DDA08/16 Analog Output block
Measurement Computing PCIM-DAS1602/16 Analog Output (D/A)	PCIM-DAS1602/16 Analog Output block
Measurement Computing PCIM-DDA06/16 Analog Output (D/A)	PCIM-DDA06/16 Analog Output block
National Instruments AT-AO-10 Analog Output (D/A)	AT-AO-10 Analog Output block
National Instruments AT-AO-6 Analog Output (D/A)	AT-AO-6 Analog Output block
National Instruments PCI-6011E Analog Output (D/A) (formerly PCI-MIO-16XE-50)	PCI-6011E Analog Output block
National Instruments PCI-6024E Analog Output (D/A)	PCI-6024E Analog Output block

National Instruments PCI-6025E Analog Output (D/A)	PCI-6025E Analog Output block
National Instruments PCI-6030E Analog Output (D/A) (formerly PCI-MIO-16XE-10)	PCI-6030E Analog Output block
National Instruments PCI-6031E Analog Output (D/A)	PCI-6031E Analog Output block
National Instruments PCI-6040E Analog Output (D/A) (formerly PCI-MIO-16E-4)	PCI-6040E Analog Output block
National Instruments PCI-6070E Analog Output (D/A) (formerly PCI-MIO-16E-1)	PCI-6070E Analog Output block
National Instruments PCI-6071E Analog Output (D/A)	PCI-6071E Analog Output block
National Instruments PCI-6221 Analog Output	National Instruments PCI-6221 Analog Output block
National Instruments PCI-6221/37 Analog Output	National Instruments PCI-6221/37 Analog Output block
National Instruments PCI-6225 Analog Output	National Instruments PCI-6225 Analog Output block
National Instruments PCI-6229 Analog Output	National Instruments PCI-6229 Analog Output block
National Instruments PCI-6251 Analog Output	National Instruments PCI-6251 Analog Output block
National Instruments PCI-6259 Analog Output	National Instruments PCI-6259 Analog Output block
National Instruments PCI-6281 Analog Output	National Instruments PCI-6281 Analog Output block
National Instruments PCI-6289 Analog Output	National Instruments PCI-6289 Analog Output block
National Instruments PCI-6703 Analog Output (D/A)	PCI-6703 Analog Output block

National Instruments PCI-6704 Analog Output (D/A)	PCI-6704 Analog Output block
National Instruments PCI/PXI-6711 Analog Output (D/A)	PCI/PXI-6711 Analog Output block
National Instruments PCI/PXI-6713 Analog Output (D/A)	PCI/PXI-6713 Analog Output block
National Instruments PCI/PXI-6731 Analog Output (D/A)	PCI/PXI-6731 Analog Output block
National Instruments PCI/PXI-6733 Analog Output (D/A)	PCI/PXI-6733 Analog Output block
National Instruments PXI-6040E Analog Output (D/A)	PXI-6040E Analog Output block
National Instruments PXI-6070E Analog Output (D/A)	PXI-6070E Analog Output block
National Instruments PXI-6071E Analog Output (D/A)	PXI-6071E Analog Output block
National Instruments PXI-6704 Analog Output (D/A)	PXI-6704 Analog Output block
National Instruments PXI/PCI-6052E Analog Output (D/A)	PXI/PCI-6052E Analog Output block
Quanser Q4 Analog Output	Q4 Analog Output block
Quanser Q8 Analog Output	Q8 Analog Output block
Real Time Devices DM6430 Analog Output (D/A)	DM6430 Analog Output block
Real Time Devices DM6604 Analog Output (D/A)	DM6604 Analog Output block
SBS IP-16DAC Analog Output (D/A)	IP-16DAC Analog Output block
SBS IP-DAC Analog Output (D/A)	IP-DAC Analog Output block
Sensoray526 DA	Sensoray526 DA block
Sensoray526 Dual DA	Sensoray526 Dual DA block

UEI PD2-AO Analog Output (D/A)	PD2-AO Analog Output block
UEI PD2-MF 12-Bit Series Analog Output (D/A)	PD2-MF 12-Bit Series Analog Output block
UEI PD2-MF 14-Bit Series Analog Output (D/A)	PD2-MF 14-Bit Series Analog Output block
UEI PD2-MF 16-Bit Series Analog Output (D/A)	PD2-MF 16-Bit Series Analog Output block
UEI PD2-MFS 12-Bit Series Analog Output (D/A)	PD2-MFS 12-Bit Series Analog Output block
UEI PD2-MFS 14-Bit Series Analog Output (D/A)	PD2-MFS 14-Bit Series Analog Output block
UEI PD2-MFS 16-Bit Series Analog Output (D/A)	PD2-MFS 16-Bit Series Analog Output block
UEI PDXI-AO Analog Output (D/A)	PDXI-AO Analog Output block
UEI PDXI-MF 12-Bit Series Analog Output (D/A)	PDXI-MF 12-Bit Series Analog Output block
UEI PDXI-MF 14-Bit Series Analog Output (D/A)	PDXI-MF 14-Bit Series Analog Output block
UEI PDXI-MF 16-Bit Series Analog Output (D/A)	PDXI-MF 16-Bit Series Analog Output block
UEI PDXI-MFS 12-Bit Series Analog Output (D/A)	PDXI-MFS 12-Bit Series Analog Output block
UEI PDXI-MFS 14-Bit Series Analog Output (D/A)	PDXI-MFS 14-Bit Series Analog Output block
UEI PDXI-MFS 16-Bit Series Analog Output (D/A)	PDXI-MFS 16-Bit Series Analog Output block

## **Data Marshalling**

Bit Packing

Bit Unpacking

Byte Packing

Byte Unpacking

Digital I/O Bit-Packing

Digital I/O Bit-Unpacking

Shared Memory Pack

Shared Memory Unpack

Construct data frames

Deconstruct data frames

Construct data frames

Deconstruct data frames

Construct data frames

Extract data frames

Shared memory pack

Shared memory unpack

## Digital Input

Adlink PCI-6208A Digital Input	Adlink PCI-6208A Digital Input
Advantech PCL-1800 Digital Input	PCL-1800 Digital Input block
Advantech PCL-711B Digital Input	PCL-711B Digital Input
Advantech PCL-726 Digital Input	PCL-726 Digital Input block
Advantech PCL-727 Digital Input	PCL-727 Digital Input block
Advantech PCL-812 Digital Input	PCL-812 Digital Input block
Advantech PCL-812PG Digital Input	PCL-812PG Digital Input block
Advantech PCL-818 Digital Input	PCL-818 Digital Input
Advantech PCL-818H Digital Input	PCL-818H Digital Input block
Advantech PCL-818HD Digital Input	PCL-818HD Digital Input block
Advantech PCL-818HG Digital Input	PCL-818HG Digital Input block
Advantech PCL-818L Digital Input	PCL-818L Digital Input block
Analogic AIM12 Digital Input	AIM12 Digital Input block
Analogic AIM16 Digital Input	AIM16 Digital Input block
BVM PMCDIO64 Digital Input	PMCDIO64 Digital Input block
Contec AD12-16(PCI) Digital Input	AD12-16(PCI) Digital Input block
Contec AD12-64(PCI) Digital Input	AD12-64(PCI) Digital Input block
Contec PI-64L(PCD)H Digital Input	PI-64L(PCD)H Digital Input block
Contec PIO-32/32F(PCI) Digital Input	PIO-32/32F(PCI) Digital Input block
Contec PIO-32/32L(PCI)H Digital Input	PIO-32/32L(PCI)H Digital Input block
Contec PIO-32/32T(PCI) Digital Input	PIO-32/32T(PCI) Digital Input block
Data Translation DT2821 Digital Input	DT2821 Digital Input block

Data Translation DT2821-F-16SE Digital Input	DT2821-F-16SE Digital Input block
Data Translation DT2821-F-8DI Digital Input	DT2821-F-8DI Digital Input block
Data Translation DT2821-G-16SE Digital Input	DT2821-G-16SE Digital Input block
Data Translation DT2821-G-8DI Digital Input	DT2821-G-8DI Digital Input block
Data Translation DT2823 Digital Input	DT2823 Digital Input block
Data Translation DT2824-PGH Digital Input	DT2824-PGH Digital Input block
Data Translation DT2824-PGL Digital Input	DT2824-PGL Digital Input block
Data Translation DT2825 Digital Input	DT2825 Digital Input block
Data Translation DT2827 Digital Input	DT2827 Digital Input block
Data Translation DT2828 Digital Input	DT2828 Digital Input block
Diamond Garnet-MM Digital Input	Garnet-MM Digital Input block
Diamond MM Digital Input	MM Digital Input block
Diamond MM-16-AT Digital Input	MM-16-AT Digital Input block
Diamond MM-32-AT Digital Input	MM-32-AT Digital Input block
Diamond Onyx-MM Digital Input	Onyx-MM Digital Input block
Diamond Onyx-MM-DIO Digital Input	Onyx-MM-DIO Digital Input block
Diamond Prometheus/Athena/Athena II Digital Input	Prometheus/Athena Digital Input block
Diamond Quartz-MM-10 Digital Input	Quartz-MM-10 Digital Input block

Diamond Quartz-MM-5 Digital Input	Quartz-MM-5 Digital Input block
Diamond Ruby-MM Digital Input	Ruby-MM Digital Input block
Diamond Ruby-MM-1612 Digital Input	Ruby-MM-1612 Digital Input block
Diamond Ruby-MM-416 Digital Input	Ruby-MM-416 Digital Input block
General Standards PMC-ADADIO Digital Input	PMC-ADADIO Digital Input block
Humusoft AD 512 Digital Input	HUMUSOFT AD 512 Digital Input block
Humusoft AD622 Digital Input	HUMUSOFT AD622 Digital Input block
Humusoft MF624 Digital Input	HUMUSOFT MF624 Digital Input block
Keithley DAS-1800HR Digital Input	DAS-1800HR Digital Input block
Keithley KPCI-1801HC Digital Input	KPCI-1801HC Digital Input block
Keithley KPCI-1802HC Digital Input	KPCI-1802HC Digital Input block
Measurement Computing CIO-DAS 1602/16 Digital Input	CIO-DAS 1602/16 Digital Input block
Measurement Computing CIO-DAS1601/12 Digital Input	CIO-DAS1601/12 Digital Input
Measurement Computing CIO-DAS1602/12 Digital Input	CIO-DAS1602/12 Digital Input block
Measurement Computing CIO-DDA06 (/12) Digital Input	CIO-DDA06 (/12) Digital Input
Measurement Computing CIO-DDA06/16 Digital Input	CIO-DDA06/16 Digital Input block
Measurement Computing CIO-DIO192 Digital Input	CIO-DIO192 Digital Input block

Measurement Computing CIO-DIO24 Digital Input	CIO-DIO24 Digital Input block
Measurement Computing CIO-DIO24H Digital Input	CIO-DIO24H Digital Input
Measurement Computing CIO-DIO48 Digital Input	CIO-DIO48 Digital Input
Measurement Computing CIO-DIO48H Digital Input	CIO-DIO48H Digital Input block
Measurement Computing CIO-DIO96 Digital Input	CIO-DIO96 Digital Input block
Measurement Computing CIO-PDISO16 Digital Input	CIO-PDISO16 Digital Input block
Measurement Computing PC104-DAS16JR/12 Digital Input	PC104-DAS16JR/12 Digital Input block
Measurement Computing PC104-DAS16JR/16 Digital Input	PC104-DAS16JR/16 Digital Input block
Measurement Computing PC104-DIO48 Digital Input	PC104-DIO48 Digital Input block
Measurement Computing PCI-DAC6703 Digital Input	PCI-DAC6703 Digital Input block
Measurement Computing PCI-DAS 1602/12 Digital Input	PCI-DAS 1602/12 Digital Input
Measurement Computing PCI-DAS 1602/16 Digital Input	PCI-DAS 1602/16 Digital Input block
Measurement Computing PCI-DAS1002 Digital Input	PCI-DAS1002 Digital Input block
Measurement Computing PCI-DAS1200 Digital Input	PCI-DAS1200 Digital Input block
Measurement Computing PCI-DAS1200/JR Digital Input	PCI-DAS1200/JR Digital Input block
Measurement Computing PCI-DDA02/12 Digital Input	PCI-DDA02/12 Digital Input

Measurement Computing PCI-DDA02/16 Digital Input	PCI-DDA02/16 Digital Input
Measurement Computing PCI-DDA04/12 Digital Input	PCI-DDA04/12 Digital Input
Measurement Computing PCI-DDA04/16 Digital Input	PCI-DDA04/16 Digital Input block
Measurement Computing PCI-DDA08/12 Digital Input	PCI-DDA08/12 Digital Input block
Measurement Computing PCI-DDA08/16 Digital Input	PCI-DDA08/16 Digital Input
Measurement Computing PCI-DIO24 Digital Input	PCI-DIO24 Digital Input
Measurement Computing PCI-DIO24H Digital Input	PCI-DIO24H Digital Input
Measurement Computing PCI-DIO48H Digital Input	PCI-DIO48H Digital Input
Measurement Computing PCI-DIO96 Digital Input	PCI-DIO96 Digital Input block
Measurement Computing PCI-DIO96H Digital Input	PCI-DIO96H Digital Input
Measurement Computing PCI-DUAL-AC5 Digital Input	PCI-DUAL-AC5 Digital Input
Measurement Computing PCI-PDISO16 Digital Input	PCI-PDISO16 Digital Input block
Measurement Computing PCI-PDISO8 Digital Input	PCI-PDISO8 Digital Input block
Measurement Computing PCIM-DAS 1602/16 Digital Input	PCIM-DAS 1602/16 Digital Input block
Measurement Computing PCIM-DDA06/16 Digital Input	PCIM-DDA06/16 Digital Input
MPL PATI Digital Input	MPL PATI Digital Input block

National Instruments PC-DIO-24 Digital Input	PC-DIO-24 Digital Input block
National Instruments PC-TIO-10 Digital Input	PC-TIO-10 Digital Input block
National Instruments PCI-6011E Digital Input (formerly PCI-MIO-16XE-50)	PCI-6011E Digital Input block
National Instruments PCI-6023E Digital Input	PCI-6023E Digital Input
National Instruments PCI-6024E Digital Input	PCI-6024E Digital Input block
National Instruments PCI-6025E and PCI-6025E 8255 Digital Input	PCI-6025E and PCI-6025E 8255 Digital Input blocks
National Instruments PCI-6030E Digital Input (formerly PCI-MIO-16XE-10)	PCI-6030E Digital Input block
National Instruments PCI-6031E Digital Input	PCI-6031E Digital Input block
National Instruments PCI-6040E Digital Input (formerly PCI-MIO-16E-4)	PCI-6040E Digital Input block
National Instruments PCI-6070E Digital Input (formerly PCI-MIO-16E-1)	PCI-6070E Digital Input
National Instruments PCI-6071E Digital Input	PCI-6071E Digital Input block
National Instruments PCI-6221 Digital Input	National Instruments PCI-6221 Digital Input block
National Instruments PCI-6221 PFI Digital Input	National Instruments PCI-6221 PFI Digital Input block
National Instruments PCI-6221/37 Digital Input	National Instruments PCI-6221/37 Digital Input block

National Instruments PCI-6221/37 PFI Digital Input	National Instruments PCI-6221/37 PFI Digital Input block
National Instruments PCI-6225 Digital Input	National Instruments PCI-6225 Digital Input block
National Instruments PCI-6225 PFI Digital Input	National Instruments PCI-6225 PFI Digital Input block
National Instruments PCI-6229 Digital Input	National Instruments PCI-6229 Digital Input block
National Instruments PCI-6229 PFI Digital Input	National Instruments PCI-6229 PFI Digital Input block
National Instruments PCI-6251 Digital Input	National Instruments PCI-6251 Digital Input block
National Instruments PCI-6251 PFI Digital Input	National Instruments PCI-6251 PFI Digital Input block
National Instruments PCI-6259 Digital Input	National Instruments PCI-6259 Digital Input block
National Instruments PCI-6259 PFI Digital Input	National Instruments PCI-6259 PFI Digital Input block
National Instruments PCI-6280 Digital Input	National Instruments PCI-6280 Digital Input block
National Instruments PCI-6280 PFI Digital Input	National Instruments PCI-6280 PFI Digital Input block
National Instruments PCI-6281 Digital Input	National Instruments PCI-6281 Digital Input block
National Instruments PCI-6281 PFI Digital Input	National Instruments PCI-6281 PFI Digital Input block
National Instruments PCI-6289 Digital Input	National Instruments PCI-6289 Digital Input block
National Instruments PCI-6289 PFI Digital Input	National Instruments PCI-6289 PFI Digital Input block
National Instruments PCI-6503 Digital Input	National Instruments PCI-6503 Digital Input block

National Instruments PCI-6527 Digital Input	National Instruments PCI-6527 Digital Input block
National Instruments PCI-6528 Digital Input	PCI-6528 Digital Input block
National Instruments PCI-6601 Digital Input	PCI-6601 Digital Input block
National Instruments PCI-DIO-96 Digital Input	PCI-DIO-96 Digital Input block
National Instruments PCI/PXI-6602 Digital Input	PCI/PXI-6602 Digital Input block
National Instruments PCI/PXI-6711 Digital Input	PCI/PXI-6711 Digital Input block
National Instruments PCI/PXI-6713 Digital Input	PCI/PXI-6713 Digital Input
National Instruments PCI/PXI-6731 Digital Input	PCI/PXI-6731 Digital Input block
National Instruments PCI/PXI-6733 Digital Input	PCI/PXI-6733 Digital Input block
National Instruments PXI-6040E Digital Input	PXI-6040E Digital Input block
National Instruments PXI-6070E Digital Input	PXI-6070E Digital Input block
National Instruments PXI-6071E Digital Input	PXI-6071E Digital Input block
National Instruments PXI-6508 Digital Input	PXI-6508 Digital Input block
National Instruments PXI-6527 Digital Input	PXI-6527 Digital Input block
National Instruments PXI/PCI-6052E Digital Input	PXI/PCI-6052E Digital Input block
Quanser Q4 Digital Input	Q4 Digital Input block
Quanser Q8 Digital Input	Q8 Digital Input block

Real Time Devices DM6420 Analog Input (A/D)	DM6420 Analog Input block
Real Time Devices DM6420 Digital Input	DM6420 Digital Input block
Real Time Devices DM6430 Digital Input	DM6430 Digital Input block
Real Time Devices DM6604 Digital Input	DM6604 Digital Input block
Real Time Devices DM6804 Digital Input	DM6804 Digital Input block
Real Time Devices DM6814 Digital Input	DM6814 Digital Input block
Real Time Devices DM7420 Digital Input	DM7420 Digital Input block
SBS IP-Digital 24 Digital Input	IP-Digital 24 Digital Input block
SBS IP-UD-IHV-16I80 Digital Input	SBS IP-UD-IHV-16I80 Digital Input block
SBS IP-UD-IHV-8I160 Digital Input	SBS IP-UD-IHV-8I160 Digital Input block
SBS IP-Unidig-E-48 Digital Input	IP-Unidig-E-48 Digital Input block
Sensoray526 DI	Sensoray526 DI block
UEI PD2-AO Digital Input	PD2-AO Digital Input block
UEI PD2-DIO-128 Digital Input	PD2-DIO-128 Digital Input block
UEI PD2-DIO-128i Digital Input	PD2-DIO-128i Digital Input block
UEI PD2-DIO-64 Digital Input	PD2-DIO-64 Digital Input block
UEI PD2-MF 12-Bit Series Digital Input	PD2-MF 12-Bit Series Digital Input block
UEI PD2-MF 14-Bit Series Digital Input	PD2-MF 14-Bit Series Digital Input block
UEI PD2-MF 16-Bit Series Digital Input	PD2-MF 16-Bit Series Digital Input block

UEI PD2-MFS 12-Bit Series Digital Input	PD2-MFS 12-Bit Series Digital Input block
UEI PD2-MFS 14-Bit Series Digital Input	PD2-MFS 14-Bit Series Digital Input block
UEI PD2-MFS 16-Bit Series Digital Input	PD2-MFS 16-Bit Series Digital Input block
UEI PDL-DIO-64 Digital Input	PDL-DIO-64 Digital Input block
UEI PDXI-AO Digital Input	PDXI-AO Digital Input block
UEI PDXI-DIO-64 Digital Input	PDXI-DIO-64 Digital Input block
UEI PDXI-MF 12-Bit Series Analog Input (A/D)	PDXI-MF 12-Bit Series Analog Input block
UEI PDXI-MF 12-Bit Series Digital Input	PDXI-MF 12-Bit Series Digital Input block
UEI PDXI-MF 14-Bit Series Digital Input	PDXI-MF 14-Bit Series Digital Input block
UEI PDXI-MF 16-Bit Series Digital Input	PDXI-MF 16-Bit Series Digital Input block
UEI PDXI-MFS 12-Bit Series Digital Input	PDXI-MFS 12-Bit Series Digital Input block
UEI PDXI-MFS 14-Bit Series Digital Input	PDXI-MFS 14-Bit Series Digital Input block
UEI PDXI-MFS 16-Bit Series Digital Input	PDXI-MFS 16-Bit Series Digital Input block
Versalogic VSBC-6 Digital Input	VSBC-6 Digital Input block

## Digital Output

	CIO-DIO24 Digital Output block
	CIO-DDA06 (/12) Digital Output block
Adlink PCI-6208A Digital Output	Adlink PCI-6208A Digital Output
Advantech PCL-1800 Digital Output	PCL-1800 Digital Output
Advantech PCL-711B Digital Output	PCL-711B Digital Output block
Advantech PCL-726 Digital Output	PCL-726 Digital Input block
Advantech PCL-727 Digital Output	PCL-727 Digital Input block
Advantech PCL-812 Digital Output	PCL-812 Digital Output block
Advantech PCL-812PG Digital Output	PCL-812PG Digital Input block
Advantech PCL-818 Digital Output	PCL-818 Digital Output
Advantech PCL-818H Digital Output	PCL-818H Digital Output block
Advantech PCL-818HD Digital Output	PCL-818HD Digital Input block
Advantech PCL-818HG Digital Output	PCL-818HG Digital Output block
Advantech PCL-818L Digital Output	PCL-818L Digital Output block
Analogic AIM12 Digital Output	AIM12 Digital Output
Analogic AIM16 Digital Output	AIM16 Digital Output
BVM PMCDIO64 Digital Output	PMCDIO64 Digital Output block
Contec AD12-16(PCI) Digital Output	AD12-16(PCI) Digital Output block
Contec AD12-64(PCI) Digital Output	AD12-64(PCI) Digital Output block
Contec PIO-32/32F(PCI) Digital Output	PIO-32/32F(PCI) Digital Output block
Contec PIO-32/32L(PCI)H Digital Output	PIO-32/32L(PCI)H Digital Output block

Contec PIO-32/32T(PCI) Digital Output	PIO-32/32T(PCI) Digital Output block
Contec PO-64L(PCI)H Digital Output	PO-64L(PCI)H Digital Output block
Data Translation DT2821 Digital Output	DT2821 Digital Output block
Data Translation DT2821-F-16SE Digital Output	DT2821-F-16SE Digital Output block
Data Translation DT2821-F-8DI Digital Output	DT2821-F-8DI Digital Output block
Data Translation DT2821-G-16SE Digital Output	DT2821-G-16SE Digital Output block
Data Translation DT2821-G-8DI Digital Output	DT2821-G-8DI Digital Output block
Data Translation DT2823 Digital Output	DT2823 Digital Output block
Data Translation DT2824-PGH Digital Output	DT2824-PGH Digital Output block
Data Translation DT2824-PGL Digital Output	DT2824-PGL Digital Output block
Data Translation DT2825 Digital Output	DT2825 Digital Output block
Data Translation DT2827 Digital Output	DT2827 Digital Output block
Data Translation DT2828 Digital Output	DT2828 Digital Output block
Diamond Garnet-MM Digital Output	Garnet-MM Digital Output block
Diamond MM Digital Output	MM Digital Output block
Diamond MM-16-AT Digital Output	MM-16-AT Digital Output block
Diamond MM-32-AT Digital Output	MM-32-AT Digital Output block
Diamond Onyx-MM Digital Output	Onyx-MM Digital Output block

Diamond Onyx-MM-DIO Digital Output	Onyx-MM-DIO Digital Output block
Diamond Prometheus/Athena/Athena II Digital Output	Prometheus/Athena Digital Output block
Diamond Quartz-MM-10 Digital Output	Quartz-MM-10 Digital Output block
Diamond Quartz-MM-5 Digital Output	Quartz-MM-5 Digital Output block
Diamond Ruby-MM Digital Output	Ruby-MM Digital Output block
Diamond Ruby-MM-1612 Digital Output	Ruby-MM-1612 Digital Output block
Diamond Ruby-MM-416 Digital Output	Ruby-MM-416 Digital Output block
General Standards PMC-ADADIO Digital Output	PMC-ADADIO Digital Output block
Humusoft AD 512 Digital Output	HUMUSOFT AD 512 Digital Output block
Humusoft AD622 Analog Output (D/A)	HUMUSOFT AD622 Analog Output block
Humusoft AD622 Digital Output	HUMUSOFT AD622 Digital Output block
Humusoft MF624 Digital Output	HUMUSOFT MF624 Digital Output block
Keithley DAS-1800HR Digital Output	DAS-1800HR Digital Output block
Keithley KPCI-1801HC Digital Output	KPCI-1801HC Digital Output block
Keithley KPCI-1802HC Digital Output	KPCI-1802HC Digital Output block
Measurement Computing CIO DAS1602/16 Digital Output	CIO DAS1602/16 Digital Output block

Measurement Computing CIO-DAS1601/12 Digital Output	CIO-DAS1601/12 Digital Output block
Measurement Computing CIO-DAS1602/12 Digital Output	CIO-DAS1602/12 Digital Output block
Measurement Computing CIO-DDA06/16 Digital Output	CIO-DDA06/16 Digital Output block
Measurement Computing CIO-DIO192 Digital Output	CIO-DIO192 Digital Output block
Measurement Computing CIO-DIO24H Digital Output	CIO-DIO24H Digital Output
Measurement Computing CIO-DIO48 Digital Output	CIO-DIO48 Digital Output block
Measurement Computing CIO-DIO48H Digital Output	CIO-DIO48H Digital Output block
Measurement Computing CIO-DIO96 Digital Output	CIO-DIO96 Digital Output block
Measurement Computing CIO-DO24DD Digital Output	CIO-DO24DD Digital Output block
Measurement Computing CIO-PDISO16 Digital Output	CIO-PDISO16 Digital Output block
Measurement Computing PC104-DAS16JR/12 Digital Output	PC104-DAS16JR/12 Digital Output block
Measurement Computing PC104-DAS16JR/16 Digital Output	PC104-DAS16JR/16 Digital Output
Measurement Computing PC104-DIO48 Digital Output	PC104-DIO48 Digital Output block
Measurement Computing PCI-DAC6703 Digital Output	PCI-DAC6703 Digital Output block
Measurement Computing PCI-DAS1002 Digital Output	PCI-DAS1002 Digital Output block
Measurement Computing PCI-DAS1200 Digital Output	PCI-DAS1200 Digital Output block

Measurement Computing PCI-DAS1200/JR Digital Output	PCI-DAS1200/JR Digital Output block
Measurement Computing PCI-DAS1602/12 Digital Output	PCI-DAS1602/12 Digital Output
Measurement Computing PCI-DAS1602/16 Digital Output	PCI-DAS1602/16 Digital Output block
Measurement Computing PCI-DDA02/12 Digital Output	PCI-DDA02/12 Digital Output
Measurement Computing PCI-DDA02/16 Digital Output	PCI-DDA02/16 Digital Output
Measurement Computing PCI-DDA04/12 Digital Output	PCI-DDA04/12 Digital Output
Measurement Computing PCI-DDA04/16 Digital Output	PCI-DDA04/16 Digital Output
Measurement Computing PCI-DDA08/12 Digital Output	PCI-DDA08/12 Digital Output block
Measurement Computing PCI-DDA08/16 Digital Output	PCI-DDA08/16 Digital Output
Measurement Computing PCI-DIO24 Digital Output	PCI-DIO24 Digital Output
Measurement Computing PCI-DIO24H Digital Output	PCI-DIO24H Digital Output
Measurement Computing PCI-DIO48H Digital Output	PCI-DIO48H Digital Output
Measurement Computing PCI-DIO96 Digital Output	PCI-DIO96 Digital Output block
Measurement Computing PCI-DIO96H Digital Output	PCI-DIO96H Digital Output
Measurement Computing PCI-DUAL-AC5 Digital Output	PCI-DUAL-AC5 Digital Output
Measurement Computing PCI-PDISO16 Digital Output	PCI-PDISO16 Digital Output block

Measurement Computing PCI-PDISO8 Digital Output	PCI-PDISO8 Digital Output
Measurement Computing PCIM-DAS1602/16 Digital Output	PCIM-DAS1602/16 Digital Output
Measurement Computing PCIM-DDA06/16 Digital Output	PCIM-DDA06/16 Digital Output
MPL PATI Digital Output	MPL PATI Digital Output block
National Instruments PC-DIO-24 Digital Output	PC-DIO-24 Digital Output
National Instruments PC-TIO-10 Digital Output	PC-TIO-10 Digital Output block
National Instruments PCI-6011E Digital Output (formerly PCI-MIO-16XE-50)	PCI-6011E Digital Output block
National Instruments PCI-6023E Digital Output	PCI-6023E Digital Output
National Instruments PCI-6024E Digital Output	PCI-6024E Digital Output
National Instruments PCI-6025E Digital Output	PCI-6025E Digital Output block
National Instruments PCI-6030E Digital Output (formerly PCI-MIO-16XE-10)	PCI-6030E Digital Output block
National Instruments PCI-6031E Digital Output	PCI-6031E Digital Output block
National Instruments PCI-6040E Digital Output (formerly PCI-MIO-16E-4)	PCI-6040E Digital Output block
National Instruments PCI-6070E Digital Output (formerly PCI-MIO-16E-1)	PCI-6070E Digital Output
National Instruments PCI-6071E Digital Output	PCI-6071E Digital Output block

National Instruments PCI-6221 Digital Output	National Instruments PCI-6221 Digital Output block
National Instruments PCI-6221 PFI Digital Output	National Instruments PCI-6221 PFI Digital Output block
National Instruments PCI-6221/37 Digital Output	National Instruments PCI-6221/37 Digital Output block
National Instruments PCI-6221/37 PFI Digital Output	National Instruments PCI-6221/37 PFI Digital Output block
National Instruments PCI-6225 Digital Output	National Instruments PCI-6225 Digital Output block
National Instruments PCI-6225 PFI Digital Output	National Instruments PCI-6225 PFI Digital Output block
National Instruments PCI-6229 Digital Output	National Instruments PCI-6229 Digital Output block
National Instruments PCI-6229 PFI Digital Output	National Instruments PCI-6229 PFI Digital Output block
National Instruments PCI-6251 Digital Output	National Instruments PCI-6251 Digital Output block
National Instruments PCI-6251 PFI Digital Output	National Instruments PCI-6251 PFI Digital Output block
National Instruments PCI-6259 Digital Output	National Instruments PCI-6259 Digital Output block
National Instruments PCI-6259 PFI Digital Output	National Instruments PCI-6259 PFI Digital Output block
National Instruments PCI-6280 Digital Output	National Instruments PCI-6280 Digital Output block
National Instruments PCI-6280 PFI Digital Output	National Instruments PCI-6280 PFI Digital Output block
National Instruments PCI-6281 Digital Output	National Instruments PCI-6281 Digital Output block
National Instruments PCI-6281 PFI Digital Output	National Instruments PCI-6281 PFI Digital Output block

National Instruments PCI-6289 Digital Output	National Instruments PCI-6289 Digital Output block
National Instruments PCI-6289 PFI Digital Output	National Instruments PCI-6289 PFI Digital Output block
National Instruments PCI-6503 Digital Output	National Instruments PCI-6503 Digital Output block
National Instruments PCI-6527 Digital Output	PCI-6527 Digital Output block
National Instruments PCI-6528 Digital Output	PCI-6528 Digital Output block
National Instruments PCI-6601 Digital Output	PCI-6601 Digital Output block
National Instruments PCI-DIO-96 Digital Output	PCI-DIO-96 Digital Output block
National Instruments PCI/PXI-6602 Digital Output	PCI/PXI-6602 Digital Output block
National Instruments PCI/PXI-6711 Digital Output	PCI/PXI-6711 Digital Output
National Instruments PCI/PXI-6713 Digital Output	PCI/PXI-6713 Digital Output block
National Instruments PCI/PXI-6731 Digital Output	PCI/PXI-6731 Digital Output
National Instruments PCI/PXI-6733 Digital Output	PCI/PXI-6733 Digital Output
National Instruments PXI-6040E Digital Output	PXI-6040E Digital Output block
National Instruments PXI-6070E Digital Output	PXI-6070E Digital Output block
National Instruments PXI-6071E Digital Output	PXI-6071E Digital Output block
National Instruments PXI-6508 Digital Output	PXI-6508 Digital Output block

National Instruments PXI-6527 Digital Output	PXI-6527 Digital Output block
National Instruments PXI/PCI-6052E Digital Output	PXI/PCI-6052E Digital Output block
Quanser Q4 Digital Output	Q4 Digital Output block
Quanser Q8 Digital Output	Q8 Digital Output block
Real Time Devices DM6420 Analog Output (D/A)	DM6420 Analog Output block
Real Time Devices DM6420 Digital Output	DM6420 Digital Output block
Real Time Devices DM6430 Digital Output	DM6430 Digital Output
Real Time Devices DM6604 Digital Output	DM6604 Digital Output block
Real Time Devices DM6804 Digital Output	DM6804 Digital Output block
Real Time Devices DM6814 Digital Output	DM6814 Digital Output block
Real Time Devices DM7420 Digital Output	DM7420 Digital Output block
SBS IP-Digital 24 Digital Output	IP-Digital 24 Digital Output block
SBS IP-UD-IHV-16I80 Digital Output	SBS IP-UD-IHV-16I80 Digital Output block
SBS IP-UD-IHV-8I160 Digital Output	SBS IP-UD-IHV-8I160 Digital Output block
SBS IP-Unidig-E-48 Digital Output	IP-Unidig-E-48 Digital Output block
Sensoray526 DO	Sensoray526 DO block
UEI PD2-AO Digital Output	PD2-AO Digital Output block
UEI PD2-DIO-128 Digital Output	PD2-DIO-128 Digital Output block
UEI PD2-DIO-128i Digital Output	PD2-DIO-128i Digital Output block
UEI PD2-DIO-64 Digital Output	PD2-DIO-64 Digital Output block

UEI PD2-MF 12-Bit Series Digital Output	PD2-MF 12-Bit Series Digital Output block
UEI PD2-MF 14-Bit Series Digital Output	PD2-MF 14-Bit Series Digital Output block
UEI PD2-MF 16-Bit Series Digital Output	PD2-MF 16-Bit Series Digital Output block
UEI PD2-MFS 12-Bit Series Digital Output	PD2-MFS 12-Bit Series Digital Output block
UEI PD2-MFS 14-Bit Series Digital Output	PD2-MFS 14-Bit Series Digital Output block
UEI PD2-MFS 16-Bit Series Digital Output	PD2-MFS 16-Bit Series Digital Output block
UEI PDL-DIO-64 Digital Output	PDL-DIO-64 Digital Output block
UEI PDXI-AO Digital Output	PDXI-AO Digital Output block
UEI PDXI-DIO-64 Digital Output	PDXI-DIO-64 Digital Output block
UEI PDXI-MF 12-Bit Series Digital Output	PDXI-MF 12-Bit Series Digital Output block
UEI PDXI-MF 14-Bit Series Digital Output	PDXI-MF 14-Bit Series Digital Output block
UEI PDXI-MF 16-Bit Series Digital Output	PDXI-MF 16-Bit Series Digital Output block
UEI PDXI-MFS 12-Bit Series Digital Output	PDXI-MFS 12-Bit Series Digital Output block
UEI PDXI-MFS 14-Bit Series Digital Output	PDXI-MFS 14-Bit Series Digital Output block
UEI PDXI-MFS 16-Bit Series Digital Output	PDXI-MFS 16-Bit Series Digital Output block
Versallogic VSBC-6 Digital Output	VSBC-6 Digital Output block

## Digital Signal Processing

TI DM642 Init

TI DM642 RX

TI DM642 TX

TI DM642 initialization block

TI DM642 receive block

TI DM642 transmit block

## Ethernet Protocols

In this section...
“Ethernet” on page 2-50
“Ethernet (Network Buffers)” on page 2-50
“Ethernet (EtherCAT)” on page 2-51

### Ethernet

Buffer Mngmt	Buffer Mngmt block
Create Ethernet Packet	Create Ethernet Packet block
Ethernet Init	Ethernet Init block
Ethernet Rx	Ethernet Rx block
Ethernet Tx	Ethernet Tx block
Extract Ethernet Packet	Extract Ethernet Packet block
Filter Address	Filter Address block
Filter Type	Filter Type block
Header Extract	Header Extract block

### Ethernet (Network Buffers)

Buffer	Network Buffer block
Buffer Mngmt	Network Buffer Mngmt block
Chain Size	Network Buffer Chain Size block
Compose	Network Buffer Compose block
Extract	Network Buffer Extract block
Link	Network Buffer Link block
Merge	Network Buffer Merge block

Split  
Unlink

Network Buffer Split block  
Network Buffer Unlink block

## **Ethernet (EtherCAT)**

EtherCAT COE Read  
EtherCAT COE Write  
EtherCAT Get State  
EtherCAT Init  
EtherCAT Rx Frames  
EtherCAT Rx Var  
EtherCAT Set State  
EtherCAT Tx Var  
EtherCAT Update

EtherCAT COE Read block  
EtherCAT COE Write block  
EtherCAT Get State block  
EtherCAT Init block  
EtherCAT Rx Frames block  
EtherCAT Rx Var block  
EtherCAT Set State block  
EtherCAT Tx Var block  
EtherCAT Update block

## **Field Programmable Gate Arrays (FPGA)**

Speedgoat IO301 PCI Read	Speedgoat IO301 PCI Read block
Speedgoat IO301 PCI Setup	Speedgoat IO301 PCI Setup block
Speedgoat IO301 PCI Write	Speedgoat IO301 PCI Write block
Speedgoat IO302 PCI Read	Speedgoat IO302 PCI Read block
Speedgoat IO302 PCI Setup	Speedgoat IO302 PCI Setup block
Speedgoat IO302 PCI Write	Speedgoat IO302 PCI Write block
Speedgoat IO303 PCI Read	Speedgoat IO303 PCI Read block
Speedgoat IO303 PCI Setup	
Speedgoat IO303 PCI Write	Speedgoat IO303 PCI Write block
Speedgoat IO311 PCI Read	Speedgoat IO311 PCI Read block
Speedgoat IO311 PCI Setup	Speedgoat IO311 PCI Setup block
Speedgoat IO311 PCI Write	Speedgoat IO311 PCI Write block
Speedgoat IO312 PCI Read	Speedgoat IO312 PCI Read block
Speedgoat IO312 PCI Setup	Speedgoat IO312 PCI Setup block
Speedgoat IO312 PCI Write	Speedgoat IO312 PCI Write block
Speedgoat IO313 PCI Read	Speedgoat IO313 PCI Read block
Speedgoat IO313 PCI Setup	Speedgoat IO313 PCI Setup block
Speedgoat IO313 PCI Write	Speedgoat IO313 PCI Write block
Speedgoat IO314 PCI Read	Speedgoat IO314 PCI Read block
Speedgoat IO314 PCI Setup	Speedgoat IO314 PCI Setup block
Speedgoat IO314 PCI Write	Speedgoat IO314 PCI Write block
Speedgoat IO325 A/D Calibration	
Speedgoat IO325 PCI Read	Speedgoat IO325 PCI Read block
Speedgoat IO325 PCI Setup	Speedgoat IO325 PCI Setup block
Speedgoat IO325 PCI Write	Speedgoat IO325 PCI Write block

## General Purpose Interface Bus (GPIB)

GPIB-232CT-A	Support for GPIB-232CT-A controller
GPIB-232CT-A Send/Receive Block	GPIB-232CT-A Send/Receive Block
GPIB-232CT-A Setup Block	GPIB-232CT-A Setup block

## Incremental Encoder

ADDI-DATA APCI-1710 Incremental Encoder	APCI-1710 Incremental Encoder
ADDI-DATA PA-1700 Incremental Encoder	PA-1700 Incremental Encoder block
Contec CNT24-4D(PCI) Incremental Encoder	CNT24-4D(PCI) Incremental Encoder block
Contec CNT32-8M(PCI) Incremental Encoder	CNT32-8M(PCI) Incremental Encoder block
Humusoft MF624 Encoder Input	HUMUSOFT MF624 Encoder Input block
Measurement Computing CIO-QUAD02 Incremental Encoder	CIO-QUAD02 Incremental Encoder block
Measurement Computing CIO-QUAD04 Incremental Encoder	CIO-QUAD04 Incremental Encoder block
Measurement Computing PCI-QUAD04 Incremental Encoder	PCI-QUAD04 Incremental Encoder
MPL PATI Incremental Encoder	MPL PATI Incremental Encoder block
National Instruments PCI-6221 Incremental Encoder	National Instruments PCI-6221 Incremental Encoder block
National Instruments PCI-6221/37 Incremental Encoder	National Instruments PCI-6221/37 Incremental Encoder block
National Instruments PCI-6225 Incremental Encoder	National Instruments PCI-6225 Incremental Encoder block
National Instruments PCI-6229 Incremental Encoder	National Instruments PCI-6229 Incremental Encoder block
National Instruments PCI-6251 Incremental Encoder	National Instruments PCI-6251 Incremental Encoder block
National Instruments PCI-6259 Incremental Encoder	National Instruments PCI-6259 Incremental Encoder block

---

National Instruments PCI-6280 Incremental Encoder	National Instruments PCI-6280 Incremental Encoder block
National Instruments PCI-6281 Incremental Encoder	National Instruments PCI-6281 Incremental Encoder block
National Instruments PCI-6289 Incremental Encoder	National Instruments PCI-6289 Incremental Encoder block
National Instruments PCI-6601 Incremental Encoder	PCI-6601 Incremental Encoder block
National Instruments PCI/PXI-6602 Incremental Encoder	PCI/PXI-6602 Incremental Encoder block
Quanser Q4 Incremental Encoder	Q4 Incremental Encoder block
Quanser Q8 Incremental Encoder	Q8 Incremental Encoder block
Real Time Devices DM6814 Incremental Encoder	DM6814 Incremental Encoder block
Sensoray526 Encoder Input	Sensoray526 Encoder Input block

## **I/O Pack (IP) Carrier**

SBS Flex-104A

Flex-104A block

SBS PCI-40A

PCI-40A block

SBS PCI-40B

PCI-40B block

## J1939 (CAN)

J1939 Controller Application	J1939 Controller Application
J1939 Database (CANdb) Setup	J1939 Database Setup
J1939 Message Trigger	J1939 Message Triggering
J1939 Protocol Stack	J1939 Protocol Stack instance
J1939 Receive Message	J1939 Receive Message
J1939 Transmit Message	J1939 Transmit

## **Light Emitting Diodes (LED)**

LED

Panel LEDs

## **Linear Variable Differential Transformer (LVDT)**

NAII 73LD3 LVDT/RVDT Converter	73LD3 LVDT/RVDT Converter block
NAII 76CL1 D/L	NAII 76CL1 D/L block
NAII 76LD1 L/D	NAII 76LD1 L/D block

## **MIL-STD-1553**

Condor 1553 Create BC Message List	Condor Create Bus Controller (BC) Message List block
Condor 1553 Decode BC Message	Condor 1553 Decode Bus Controller (BC) Message block
Condor 1553 Decode BC Status	Condor 1553 Decode Bus Controller (BC) Status block
Condor 1553 Encode BC Message	Condor 1553 Encode Bus Controller (BC) Message block
Condor 1553 Select BM Message	Condor Select 1553 Bus Monitor (BC) Message block
Condor PCI-1553 Bus Controller Send	Condor PCI-1553 Bus Controller (BC) Send block
Condor PCI-1553 Bus Monitor	Condor PCI-1553 Bus Monitor block
Condor PCI-1553 Initialize	Condor PCI-1553 Initialize block
Condor PCI-1553 RT Initialize	Condor PCI-1553 RT Initialize block
Condor PCI-1553 RT Receive	Condor PCI-1553 Remote Terminal (RT) Receive
Condor PCI-1553 RT Send	Condor PCI-1553 Remote Terminal (RT) Send block
Condor PCI/QPCI-1553	Support for MIL-STD-1553 PCI and QPCI boards.
Condor Q104-1553	Support for MIL-STD-1553 PC104 boards.
Condor Q104-1553 Bus Controller Send	Condor Q104-1553 Bus Controller (BC) Send block
Condor Q104-1553 Bus Monitor	Condor Q104-1553 Bus Monitor block
Condor Q104-1553 Initialize	Condor Q104-1553 Initialize block
Condor Q104-1553 RT Initialize	Condor Q104-1553 RT Initialize block

---

Condor Q104-1553 RT Receive	Condor Q104-1553 Remote Terminal (RT) Receive block
Condor Q104-1553 RT Send	Condor Q104-1553 Remote Terminal Send (RT) block
Condor QPCI-1553 Bus Controller Send	Condor QPCI-1553 Bus Controller (BC) Send block
Condor QPCI-1553 Bus Monitor	Condor QPCI-1553 Bus Monitor block
Condor QPCI-1553 Initialize	Condor QPCI-1553 Initialize block
Condor QPCI-1553 RT Initialize	Condor QPCI-1553 RT Initialize block
Condor QPCI-1553 RT Receive	Condor QPCI-1553 Remote Terminal (RT) Receive block
Condor QPCI-1553 RT Send	Condor QPCI-1553 Remote Terminal Send (RT) block

## Miscellaneous

Async Buffer Write and Read	Async Buffer Write and Read blocks
Async IRQ Source	Async IRQ Source block
Async Rate Transition	Async Rate Transition block
From File	From File
From xPC Target	Data from target computer
I/O Port Read	I/O Port Read
I/O Port Write	I/O Port Write
Pentium Time Stamp Counter	Pentium time stamp counter
Scope (xPC)	Scope block
Software Reboot	Software Reboot
Time Stamp Delta	Time stamp delta
To xPC Target	Data to target computer
XCP Server	XCP server
xPC Target Get Free Stack Size	xPC Target Get Free Stack Size
xPC Target Get Minimal Free Stack Size	xPC Target Get Minimal Free Stack Size
xPC Target Get Overload Counter	Return current CPU overload count
xPC Target Set Overload Counter	Set current CPU overload count
xPC Target TET	xPC Target TET
xPC Target Time	xPC Target Time

## Parallel Port

Parallel Port Digital Input	Parallel Port Digital Input block
Parallel Port Digital Input Status Bits	Parallel Port Digital Input Status Bits block
Parallel Port Digital Output	Parallel Port Digital Output block
Parallel Port Digital Output Control Bits	Parallel Port Digital Output Control Bits block

## Pulse Width Measurement

Adlink PCI-8133 3-Phase PWM	Adlink PCI-8133 3-Phase pulse width modulator block
Humusoft MF624 PWM Output	HUMUSOFT MF624 PWM Output block
MPL PATI PWM generate	MPL PATI PWM generate block
MPL PATI PWM measure	MPL PATI PWM measure
National Instruments PCI-6011E Pulse Width/Period Measurement (formerly PCI-MIO-16XE-50)	PCI-6011E Pulse Width/Period Measurement
National Instruments PCI-6023E Pulse Width/Period Measurement	PCI-6023E Pulse Width/Period Measurement
National Instruments PCI-6024E Pulse Width/Period Measurement	PCI-6024E Pulse Width/Period Measurement
National Instruments PCI-6025E Pulse Width/Period Measurement	PCI-6025E Pulse Width/Period Measurement block
National Instruments PCI-6030E Pulse Width/Period Measurement (formerly PCI-MIO-16XE-10)	PCI-6030E Pulse Width/Period Measurement
National Instruments PCI-6031E Pulse Width/Period Measurement	PCI-6031E Pulse Width/Period Measurement block
National Instruments PCI-6040E Pulse Width/Period Measurement (formerly PCI-MIO-16E-4)	PCI-6040E Pulse Width/Period Measurement block
National Instruments PCI-6070E Pulse Width/Period Measurement (formerly PCI-MIO-16E-1)	PCI-6070E Pulse Width/Period Measurement
National Instruments PCI-6071E Pulse Width/Period Measurement	PCI-6071E Pulse Width/Period Measurement block
National Instruments PCI-6221 PWM Generate	National Instruments PCI-6221 PWM Generate block

National Instruments PCI-6221 PWM Measure	National Instruments PCI-6221 PWM Measure block
National Instruments PCI-6221/37 PWM Generate	National Instruments PCI-6221/37 PWM Generate block
National Instruments PCI-6221/37 PWM Measure	National Instruments PCI-6221/37 PWM Measure block
National Instruments PCI-6225 PWM Generate	National Instruments PCI-6225 PWM Generate block
National Instruments PCI-6225 PWM Measure	National Instruments PCI-6225 PWM Measure block
National Instruments PCI-6229 PWM Generate	National Instruments PCI-6229 PWM Generate block
National Instruments PCI-6229 PWM Measure	National Instruments PCI-6229 PWM Measure block
National Instruments PCI-6251 PWM Generate	National Instruments PCI-6251 PWM Generate block
National Instruments PCI-6251 PWM Measure	National Instruments PCI-6251 PWM Measure block
National Instruments PCI-6259 PWM Generate	National Instruments PCI-6259 PWM Generate block
National Instruments PCI-6259 PWM Measure	National Instruments PCI-6259 PWM Measure block
National Instruments PCI-6280 PWM Generate	National Instruments PCI-6280 PWM Generate block
National Instruments PCI-6280 PWM Measure	National Instruments PCI-6280 PWM Measure block
National Instruments PCI-6281 PWM Generate	National Instruments PCI-6281 PWM Generate block
National Instruments PCI-6281 PWM Measure	National Instruments PCI-6281 PWM Measure block
National Instruments PCI-6289 PWM Generate	National Instruments PCI-6289 PWM Generate block

National Instruments PCI-6289 PWM Measure	National Instruments PCI-6289 PWM Measure block
National Instruments PCI-6601 Pulse Width/Period Measurement	PCI-6601 Pulse Width/Period Measurement block
National Instruments PCI/PXI-6602 Pulse Width/Period Measurement	PCI/PXI-6602 Pulse Width/Period Measurement block
National Instruments PCI/PXI-6711 Pulse Width/Period Measurement	PCI/PXI-6711 Pulse Width/Period Measurement block
National Instruments PCI/PXI-6713 Pulse Width/Period Measurement	PCI/PXI-6713 Pulse Width/Period Measurement block
National Instruments PCI/PXI-6731 Pulse Width/Period Measurement	PCI/PXI-6731 Pulse Width/Period Measurement block
National Instruments PCI/PXI-6733 Pulse Width/Period Measurement	PCI/PXI-6733 Pulse Width/Period Measurement block
National Instruments PXI-6040E Pulse Width/Period Measurement	PXI-6040E Pulse Width/Period Measurement block
National Instruments PXI-6070E Pulse Width/Period Measurement	PXI-6070E Pulse Width/Period Measurement block
National Instruments PXI-6071E Pulse Width/Period Measurement	PXI-6071E Pulse Width/Period Measurement block
National Instruments PXI/PCI-6052E Pulse Width/Period Measurement	PXI/PCI-6052E Pulse Width/Period Measurement
Real Time Devices DM6816 PWM	DM6816 PWM block

## Real-Time Performance

Extended Counter	Sample extended counter
Pentium Time Stamp Counter	Pentium time stamp counter
Rollover Counter	Sample rollover counter
Time Stamp Delta	Time stamp delta
xPC Target Get Overload Counter	Return current CPU overload count
xPC Target Set Overload Counter	Set current CPU overload count
xPC Target TET	xPC Target TET
xPC Target Time	xPC Target Time

## **RS-232**

ASCII Decode V2	ASCII Decode V2
ASCII Encode/Decode (Composite)	ASCII Encode/Decode
FIFO Read Binary (Composite)	FIFO Read Binary
FIFO Read HDRS (Composite)	FIFO Read HDRS block
FIFO Read/Write (Composite)	FIFO Read/Write
Modem Control (Composite)	Modem Control block
Modem Status (Composite)	Modem Status block
RS-232 Binary Pack	RS-232 Binary Pack block
RS-232 Binary Receive	RS-232 Binary Receive block
RS-232 Binary Send	RS-232 Binary Send block
RS-232 Binary Unpack Block	RS-232 Binary Unpack Block
RS-232 Mainboard Receive (Asynchronous)	RS-232 Mainboard Receive block (Asynchronous)
RS-232 Mainboard Send (Asynchronous)	RS-232 Mainboard Send block (Asynchronous)
RS-232 Mainboard Send/Receive (Synchronous)	RS-232 Mainboard Send/Receive Block (Synchronous)
RS-232 Mainboard Setup	RS-232 Mainboard Setup block
RS-232/RS-422/RS-485 Send/Receive (Composite)	RS-232/RS-422/RS-485 Send/Receive block
RS232 State (Composite)	RS232 State block

## Shared Memory

GE Fanuc 5565 broadcast	GE Fanuc PCI-5565 broadcast interrupt block
GE Fanuc 5565 init	GE Fanuc PCI-5565 initialization block
GE Fanuc 5565 read	GE Fanuc PCI-5565 read block
GE Fanuc 5565 write	GE Fanuc PCI-5565 write block
SBS25x0 init	SBS25x0 init block
SBS25x0 read	SBS25x0 read block
SBS25x0 write	SBS25x0 write block
Systran SC150 init	Systran®SC150 init block
Systran SC150 init	SystranSC150 init block
Systran SC150 read	Systran SC150 read block
Systran SC150 read	Systran SC150 read block
Systran SC150 rearm	Systran SC150 rearm block
Systran SC150 rearm	Systran SC150 rearm block
Systran SC150 write	Systran SC150 write block
Systran SC150 write	Systran SC150 write block

## **Signal Conditioning**

Measurement Computing  
CIO-DIO24 Signal Conditioning

CIO-DIO24 Signal Conditioning  
block

Measurement Computing  
PCI-DIO24 Signal Conditioning

PCI-DIO24 Signal Conditioning

## Synchro Resolver

NAII 73SD3 Synchro/Resolver

NAII 76CS1 D/S

NAII 76CS1 S/D

NAII PC-12SD (PC-77SD1)  
Synchro/Resolver

SBS IP-Synchro/Resolver

NAII 73SD3 Synchro/Resolver block

NAII 76CS1 D/S block

NAII 76CS1 S/D block

PC-12SD (PC-77SD1)  
Synchro/Resolver block

IP-Synchro/Resolver block

## **Thermocouple**

Measurement Computing  
PCI-DAS-TC Thermocouple

PCI-DAS-TC Thermocouple

## User Datagram Protocol (UDP) Protocols

### In this section...

“UDP” on page 2-73

“UDP (Real Time)” on page 2-73

“UDP (Internet Protocol)” on page 2-73

### UDP

Byte Reversal/Change Endianess	Byte Reversal/Change Endianess block
UDP Pack	UDP Pack block
UDP Receive	UDP Receive block
UDP Send	UDP Send block
UDP Unpack	UDP Unpack block

### UDP (Real Time)

Network Configuration	Real-time UDP Network Configuration block
Receive	Real-time UDP Receive block
Send	Real-time UDP Send block

### UDP (Internet Protocol)

Network Configuration	Real-time UDP Network Configuration block
UDP Consume	Real-time UDP Consume block
UDP Produce	Real-time UDP Produce block

UDP Rx

Real-time UDP Rx block

UDP Tx

Real-time UDP Tx block

## Utility

Bit Packing	Construct data frames
Bit Unpacking	Deconstruct data frames
Byte Packing	Construct data frames
Byte Unpacking	Deconstruct data frames
Digital I/O Bit-Packing	Construct data frames
Digital I/O Bit-Unpacking	Extract data frames
Extended Counter	Sample extended counter
Rollover Counter	Sample rollover counter
Shared Memory Pack	Shared memory pack
Shared Memory Unpack	Shared memory unpack

## **Video**

From USB Video Device

Image Receive

Image Transmit

JPEG Compression

JPEG Decompression

NEON BitFlow Image Input

USB Video Device List

From USB Video Device block

Image Receive block

Image Transmit block

JPEG Compression block

JPEG Decompression block

Support for NEON BitFlow Image  
Input block

USB Video Device List block

## Visualization

Scope (xPC)

Scope block

## **XCP (CAN)**

XCP DAQSTIM

XCP DAQSTIM block

XCP Protocol Stack

XCP Protocol Stack block

## Watchdog Timer

ACCES I/O WDG-CSM Watchdog  
Timer

WDG-CSM Watchdog Timer block

Versallogic VSBC-6 Watch Dog  
Watchdog Timer

VSBC-6 Watch Dog block

Watchdog timer



# Boards and Blocks — By Manufacturer

---

ACCES I/O (p. 3-4)	Boards and blocks manufactured by ACCES I/O.
ADDI-DATA (p. 3-5)	Boards and blocks manufactured by ADDI-DATA.
Adlink (p. 3-6)	Boards and blocks manufactured by Adlink.
Advantech (p. 3-7)	Boards and blocks manufactured by Advantech.
Analogic (p. 3-10)	Boards and blocks manufactured by Analogic.
BitFlow (p. 3-11)	Boards and blocks manufactured by BitFlow.
BittWare (p. 3-12)	Boards and blocks manufactured by BittWare.
BVM (p. 3-13)	Boards and blocks manufactured by BVM.
Commtech (p. 3-14)	Boards and blocks manufactured by Commtech.
Contec (p. 3-15)	Boards and blocks manufactured by Contec.
Curtiss-Wright Electronic Systems (p. 3-17)	Boards and blocks manufactured by Curtiss-Wright Electronic Systems (formerly Systran).

Data Translation (p. 3-18)	Boards and blocks manufactured by Data Translation.
Diamond Systems (p. 3-21)	Boards and blocks manufactured by Diamond Systems.
GE Intelligent Platforms (p. 3-24)	Boards and blocks manufactured by GE® Intelligent Platforms, formerly by Condor Engineering®, SBS Technologies, and VMIC.
General Standards (p. 3-29)	Boards and blocks manufactured by General Standards.
HUMUSOFT (p. 3-30)	Boards and blocks manufactured by HUMUSOFT.
Intel (p. 3-31)	Boards and blocks manufactured by Intel.
Keithley (p. 3-32)	Boards and blocks manufactured by Keithley.
Measurement Computing (p. 3-33)	Boards and blocks manufactured by Measurement Computing (formerly Computer Boards).
MPL (p. 3-45)	Boards and blocks manufactured by MPL.
National Instruments (p. 3-46)	Boards and blocks manufactured by National Instruments.
North Atlantic Industries (p. 3-63)	Boards and blocks manufactured by North Atlantic Industries (formerly Apex).
Philips (p. 3-64)	Boards and blocks manufactured by Philips.
Quanser (p. 3-66)	Boards and blocks manufactured by Quanser.
Quatech (p. 3-67)	Boards and blocks manufactured by Quatech.

---

Real Time Devices (p. 3-68)	Boards and blocks manufactured by Real Time Devices.
Sensoray (p. 3-70)	Boards and blocks manufactured by Sensoray.
Softing (p. 3-71)	Boards and blocks manufactured by Softing.
Speedgoat (p. 3-73)	Boards and blocks manufactured by Speedgoat.
Texas Instruments (p. 3-75)	Boards and blocks manufactured by Texas Instruments.
United Electronic Industries (UEI) (p. 3-76)	Boards and blocks manufactured by United Electronic Industries (UEI).
Versallogic (p. 3-82)	Boards and blocks manufactured by Versallogic.

## **ACCES I/O**

ACCES I/O WDG-CSM

Support for ACCES I/O WDG-CSM  
watchdog timer

ACCES I/O WDG-CSM Watchdog  
Timer

WDG-CSM Watchdog Timer block

## **ADDI-DATA**

ADDI-DATA APCI-1710

Support for APCI-1710 general purpose counting board

ADDI-DATA APCI-1710 Incremental Encoder

APCI-1710 Incremental Encoder

ADDI-DATA PA-1700

Support for the PI1700 counter board

ADDI-DATA PA-1700 Incremental Encoder

PA-1700 Incremental Encoder block

## **Adlink**

Adlink PCI-6208	Support for the Adlink PCI-6208 digital-to-analog converter board
Adlink PCI-6208A Analog Output	Adlink PCI-6208A Analog Output block
Adlink PCI-6208A Digital Input	Adlink PCI-6208A Digital Input
Adlink PCI-6208A Digital Output	Adlink PCI-6208A Digital Output
Adlink PCI-8133	Support for Adlink PCI-8133 encoder counter and PWM output board
Adlink PCI-8133 3-Phase PWM	Adlink PCI-8133 3-Phase pulse width modulator block

## Advantech

Advantech PCL-1800 Analog Output (D/A)	PCL-1800 Analog Output block
Advantech PCL-1800 Digital Input	PCL-1800 Digital Input block
Advantech PCL-1800 Digital Output	PCL-1800 Digital Output block
Advantech PCL-711B Analog Input (A/D)	PCL-711B Analog Input block
Advantech PCL-711B Analog Output (D/A)	PCL-711B Analog Output block
Advantech PCL-711B Digital Input	PCL-711B Digital Input block
Advantech PCL-711B Digital Output	PCL-711B Digital Output block
Advantech PCL-726 Analog Output (D/A)	PCL-726 Analog Output block
Advantech PCL-726 Digital Input	PCL-726 Digital Input block
Advantech PCL-726 Digital Output	PCL-726 Digital Output block
Advantech PCL-727 Analog Output (D/A)	PCL-727 Analog Output block
Advantech PCL-727 Digital Input	PCL-727 Digital Input block
Advantech PCL-727 Digital Output	PCL-727 Digital Output block
Advantech PCL-728 Analog Output (D/A)	PCL-728 Analog Output block
Advantech PCL-812 Analog Input (A/D)	PCL-812 Analog Input block
Advantech PCL-812 Analog Output (D/A)	PCL-812 Analog Output block
Advantech PCL-812 Digital Input	PCL-812 Digital Input block
Advantech PCL-812 Digital Output	PCL-812 Digital Output block
Advantech PCL-812PG Analog Input (A/D)	PCL-812PG Analog Input block

Advantech PCL-812PG Analog Output (D/A)	PCL-812PG Analog Output block
Advantech PCL-812PG Digital Input	PCL-812PG Digital Input block
Advantech PCL-812PG Digital Output	PCL-812PG Digital Input block
Advantech PCL-818 Analog Input (A/D)	PCL-818 Analog Input (A/D)
Advantech PCL-818 Analog Output (D/A)	PCL-818 Analog Output block
Advantech PCL-818 Digital Input	PCL-818 Digital Input
Advantech PCL-818 Digital Output	PCL-818 Digital Output
Advantech PCL-818H Analog Input (A/D)	PCL-818H Analog Input block
Advantech PCL-818H Analog Output (D/A)	PCL-818H Analog Output block
Advantech PCL-818H Digital Input	PCL-818H Digital Input block
Advantech PCL-818H Digital Output	PCL-818H Digital Output block
Advantech PCL-818HD Analog Input (A/D)	PCL-818HD Analog Input block
Advantech PCL-818HD Analog Output (D/A)	PCL-818HD Analog Output block
Advantech PCL-818HD Digital Input	PCL-818HD Digital Input block
Advantech PCL-818HD Digital Output	PCL-818HD Digital Input block
Advantech PCL-818HG Analog Input (A/D)	PCL-818HG Analog Input block
Advantech PCL-818HG Analog Output (D/A)	PCL-818HG Analog Output block
Advantech PCL-818HG Digital Input	PCL-818HG Digital Input block

---

Advantech PCL-818HG Digital Output	PCL-818HG Digital Output block
Advantech PCL-818L Analog Input (A/D)	PCL-818L Analog Input block
Advantech PCL-818L Analog Output (D/A)	PCL-818L Analog Output block
Advantech PCL-818L Digital Input	PCL-818L Digital Input block
Advantech PCL-818L Digital Output	PCL-818L Digital Output block

## **Analogic**

Analogic AIM12 Analog Input (A/D)	Analogic AIM12 Analog Input block
Analogic AIM12 Digital Input	AIM12 Digital Input block
Analogic AIM12 Digital Output	AIM12 Digital Output
Analogic AIM16	Analogic AIM16 board
Analogic AIM16 Analog Input (A/D)	AIM16 Analog Input block
Analogic AIM16 Digital Input	AIM16 Digital Input block
Analogic AIM16 Digital Output	AIM16 Digital Output

# BitFlow

BitFlow Neon-CLB

Support for BitFlow™ Neon-CLB camera interface board

NEON BitFlow Image Input

Support for NEON BitFlow Image Input block

## **BittWare**

BittWare Audio-PMC+

Support for BittWare Audio PMC+ board

BittWare Audio-PMC+ Analog Input

Audio-PMC+ Analog Input block

BittWare Audio-PMC+ Analog Output

Audio-PMC+ Analog Output block

## **BVM**

BVM PMCDIO64

BVM PMCDIO64 Digital Input

BVM PMCDIO64 Digital Output

Support for 64-bit digital I/O

PMCDIO64 Digital Input block

PMCDIO64 Digital Output block

## Commtech

ASCII Decode V2	ASCII Decode V2
ASCII Encode/Decode (Composite)	ASCII Encode/Decode
FIFO Read Binary (Composite)	FIFO Read Binary
FIFO Read HDRS (Composite)	FIFO Read HDRS block
FIFO Read/Write (Composite)	FIFO Read/Write
Modem Control (Composite)	Modem Control block
Modem Status (Composite)	Modem Status block
RS-232/RS-422/RS-485 Send/Receive (Composite)	RS-232/RS-422/RS-485 Send/Receive block
RS232 State (Composite)	RS232 State block

## Contec

Contec AD12-16(PCI)	
Contec AD12-16(PCI) Analog Input (A/D)	AD12-16(PCI) Analog Input (A/D) block
Contec AD12-16(PCI) Digital Input	AD12-16(PCI) Digital Input block
Contec AD12-16(PCI) Digital Output	AD12-16(PCI) Digital Output block
Contec AD12-16(PCI)E	
Contec AD12-16(PCI)E Analog Input (A/D)	AD12-16(PCI)E Analog Input block
Contec AD12-16(PCI)E Analog Output (D/A)	AD12-16(PCI)E Analog Output block
Contec AD12-16U(PCI)E	
Contec AD12-16U(PCI)E Analog Input (A/D)	AD12-16U(PCI)E Analog Input block
Contec AD12-16U(PCI)E Analog Output (D/A)	AD12-16U(PCI)E Analog Output block
Contec AD12-64(PCI)	Contec AD12-64(PCI) block
Contec AD12-64(PCI) Analog Input (A/D)	AD12-64(PCI) Analog Input block
Contec AD12-64(PCI) Digital Input	AD12-64(PCI) Digital Input block
Contec AD12-64(PCI) Digital Output	AD12-64(PCI) Digital Output block
Contec AD16-16(PCI)E	
Contec AD16-16(PCI)E Analog Input (A/D)	AD16-16(PCI)E Analog Input block
Contec AD16-16(PCI)E Analog Output (D/A)	AD16-16(PCI)E Analog Output block
Contec ADI12-16(PCI)	
Contec ADI12-16(PCI) Analog Input (A/D)	ADI12-16(PCI) Analog Input block
Contec CNT24-4D(PCI)	

Contec CNT24-4D(PCI) Incremental Encoder	CNT24-4D(PCI) Incremental Encoder block
Contec CNT32-8M(PCI)	
Contec CNT32-8M(PCI) Incremental Encoder	CNT32-8M(PCI) Incremental Encoder block
Contec DA12-16(PCI)	Contec DA12-16(PCI) block
Contec DA12-16(PCI) Analog Output (D/A)	DA12-16(PCI) Analog Output block
Contec DA12-4(PCI)	
Contec DA12-4(PCI) Analog Output (D/A)	DA12-4(PCI) Analog Output block
Contec PI-64L(PCD)H	
Contec PI-64L(PCD)H Digital Input	PI-64L(PCD)H Digital Input block
Contec PIO-32/32F(PCI)	
Contec PIO-32/32F(PCI) Digital Input	PIO-32/32F(PCI) Digital Input block
Contec PIO-32/32F(PCI) Digital Output	PIO-32/32F(PCI) Digital Output block
Contec PIO-32/32L(PCI)H	
Contec PIO-32/32L(PCI)H Digital Input	PIO-32/32L(PCI)H Digital Input block
Contec PIO-32/32L(PCI)H Digital Output	PIO-32/32L(PCI)H Digital Output block
Contec PIO-32/32T(PCI)	
Contec PIO-32/32T(PCI) Digital Input	PIO-32/32T(PCI) Digital Input block
Contec PIO-32/32T(PCI) Digital Output	PIO-32/32T(PCI) Digital Output block
Contec PO-64L(PCI)H	
Contec PO-64L(PCI)H Digital Output	PO-64L(PCI)H Digital Output block

## Curtiss-Wright Electronic Systems

Systran SC150 init	SystranSC150 init block
Systran SC150 init	SystranSC150 init block
Systran SC150 read	Systran SC150 read block
Systran SC150 read	Systran SC150 read block
Systran SC150 rearm	Systran SC150 rearm block
Systran SC150 rearm	Systran SC150 rearm block
Systran SC150 write	Systran SC150 write block
Systran SC150 write	Systran SC150 write block
Systran SCRAMNet+ SC150 PCI	Support for Systran SCRAMNet+ SC150 PCI real-time reflective memory board
Systran SCRAMNet+ SC150 PCI	Support for Systran SCRAMNet+ SC150 PCI real-time reflective memory board

## Data Translation

Data Translation DT2821 Analog Input (A/D)	DT2821 Analog Input block
Data Translation DT2821 Analog Output (D/A)	DT2821 Analog Output block
Data Translation DT2821 Digital Input	DT2821 Digital Input block
Data Translation DT2821 Digital Output	DT2821 Digital Output block
Data Translation DT2821-F-16SE Analog Input (A/D)	DT2821-F-16SE Analog Input block
Data Translation DT2821-F-16SE Analog Output (D/A)	DT2821-F-16SE Analog Output block
Data Translation DT2821-F-16SE Digital Input	DT2821-F-16SE Digital Input block
Data Translation DT2821-F-16SE Digital Output	DT2821-F-16SE Digital Output block
Data Translation DT2821-F-8DI Analog Input (A/D)	DT2821-F-8DI Analog Input block
Data Translation DT2821-F-8DI Analog Output (D/A)	DT2821-F-8DI Analog Output block
Data Translation DT2821-F-8DI Digital Input	DT2821-F-8DI Digital Input block
Data Translation DT2821-F-8DI Digital Output	DT2821-F-8DI Digital Output block
Data Translation DT2821-G-16SE Analog Input (A/D)	DT2821-G-16SE Analog Input block
Data Translation DT2821-G-16SE Analog Output (D/A)	DT2821-G-16SE Analog Output block
Data Translation DT2821-G-16SE Digital Input	DT2821-G-16SE Digital Input block

Data Translation DT2821-G-16SE Digital Output	DT2821-G-16SE Digital Output block
Data Translation DT2821-G-8DI Analog Input (A/D)	DT2821-G-8DI Analog Input block
Data Translation DT2821-G-8DI Analog Output (D/A)	DT2821-G-8DI Analog Output block
Data Translation DT2821-G-8DI Digital Input	DT2821-G-8DI Digital Input block
Data Translation DT2821-G-8DI Digital Output	DT2821-G-8DI Digital Output block
Data Translation DT2823 Analog Input (A/D)	DT2823 Analog Input block
Data Translation DT2823 Analog Output (D/A)	DT2823 Analog Output block
Data Translation DT2823 Digital Input	DT2823 Digital Input block
Data Translation DT2823 Digital Output	DT2823 Digital Output block
Data Translation DT2824-PGH Analog Input (A/D)	DT2824-PGH Analog Input block
Data Translation DT2824-PGH Digital Input	DT2824-PGH Digital Input block
Data Translation DT2824-PGH Digital Output	DT2824-PGH Digital Output block
Data Translation DT2824-PGL Analog Input (A/D)	DT2824-PGL Analog Input block
Data Translation DT2824-PGL Digital Input	DT2824-PGL Digital Input block
Data Translation DT2824-PGL Digital Output	DT2824-PGL Digital Output block
Data Translation DT2825 Analog Input (A/D)	DT2825 Analog Input block

Data Translation DT2825 Analog Output (D/A)	DT2825 Analog Output block
Data Translation DT2825 Digital Input	DT2825 Digital Input block
Data Translation DT2825 Digital Output	DT2825 Digital Output block
Data Translation DT2827 Analog Input (A/D)	DT2827 Analog Input block
Data Translation DT2827 Analog Output (D/A)	DT2827 Analog Output block
Data Translation DT2827 Digital Input	DT2827 Digital Input block
Data Translation DT2827 Digital Output	DT2827 Digital Output block
Data Translation DT2828 Analog Input (A/D)	DT2828 Analog Input block
Data Translation DT2828 Analog Output (D/A)	DT2828 Analog Output block
Data Translation DT2828 Digital Input	DT2828 Digital Input block
Data Translation DT2828 Digital Output	DT2828 Digital Output block

## Diamond Systems

ASCII Decode V2	ASCII Decode V2
ASCII Encode/Decode (Composite)	ASCII Encode/Decode
Diamond Garnet-MM Digital Input	Garnet-MM Digital Input block
Diamond Garnet-MM Digital Output	Garnet-MM Digital Output block
Diamond MM Analog Input (A/D)	MM Analog Input block
Diamond MM Analog Output (D/A)	MM Analog Output block
Diamond MM Digital Input	MM Digital Input block
Diamond MM Digital Output	MM Digital Output block
Diamond MM-16-AT Analog Input (A/D)	MM-16-AT Analog Input block
Diamond MM-16-AT Analog Output (D/A)	MM-16-AT Analog Output block
Diamond MM-16-AT Digital Input	MM-16-AT Digital Input block
Diamond MM-16-AT Digital Output	MM-16-AT Digital Output block
Diamond MM-32-AT Analog Input (A/D)	MM-32-AT Analog Input block
Diamond MM-32-AT Analog Output (D/A)	MM-32-AT Analog Output block
Diamond MM-32-AT Digital Input	MM-32-AT Digital Input block
Diamond MM-32-AT Digital Output	MM-32-AT Digital Output block
Diamond MM-32-AT Frame Analog Input (A/D)	MM-32-AT Frame Analog Input block
Diamond Onyx-MM Digital Input	Onyx-MM Digital Input block
Diamond Onyx-MM Digital Output	Onyx-MM Digital Output block
Diamond Onyx-MM-DIO Digital Input	Onyx-MM-DIO Digital Input block
Diamond Onyx-MM-DIO Digital Output	Onyx-MM-DIO Digital Output block

Diamond Prometheus/Athena/Athena II Analog Input (A/D)	Prometheus/Athena Analog Input block
Diamond Prometheus/Athena/Athena II Analog Output (D/A)	Prometheus/Athena Analog Output block
Diamond Prometheus/Athena/Athena II Digital Input	Prometheus/Athena Digital Input block
Diamond Prometheus/Athena/Athena II Digital Output	Prometheus/Athena Digital Output block
Diamond Quartz-MM-10 Digital Input	Quartz-MM-10 Digital Input block
Diamond Quartz-MM-10 Digital Output	Quartz-MM-10 Digital Output block
Diamond Quartz-MM-5 Digital Input	Quartz-MM-5 Digital Input block
Diamond Quartz-MM-5 Digital Output	Quartz-MM-5 Digital Output block
Diamond Ruby-MM Analog Output (D/A)	Ruby-MM Analog Output block
Diamond Ruby-MM Digital Input	Ruby-MM Digital Input block
Diamond Ruby-MM Digital Output	Ruby-MM Digital Output block
Diamond Ruby-MM-1612	Diamond Ruby-MM-1612 board
Diamond Ruby-MM-1612 Analog Output (D/A)	Ruby-MM-1612 Analog Output block
Diamond Ruby-MM-1612 Digital Input	Ruby-MM-1612 Digital Input block
Diamond Ruby-MM-1612 Digital Output	Ruby-MM-1612 Digital Output block
Diamond Ruby-MM-416 Analog Output (D/A)	Ruby-MM-416 Analog Output block

---

Diamond Ruby-MM-416 Digital Input	Ruby-MM-416 Digital Input block
Diamond Ruby-MM-416 Digital Output	Ruby-MM-416 Digital Output block
FIFO Read Binary (Composite)	FIFO Read Binary
FIFO Read HDRS (Composite)	FIFO Read HDRS block
FIFO Read/Write (Composite)	FIFO Read/Write
Modem Control (Composite)	Modem Control block
Modem Status (Composite)	Modem Status block
RS-232/RS-422/RS-485 Send/Receive (Composite)	RS-232/RS-422/RS-485 Send/Receive block
RS232 State (Composite)	RS232 State block

## GE Intelligent Platforms

In this section...
“ARINC 429 Support” on page 3-24
“GE Fanuc Embedded Systems Shared Memory Support” on page 3-24
“SBS Technologies Support” on page 3-25
“MIL-STD-1553 Support” on page 3-27

### ARINC 429 Support

Condor CEI-520/520A

Support for ARINC 429  
CEI-520/5201A board.

Condor CEI-x20 Initialize

Condor CEI-x20 Initialize block

Condor CEI-x20 Receive

Condor CEI-x20 Receive block

Condor CEI-x20 Send

Condor CEI-x20 Send block

Condor Decode ARINC 429 Words  
from Receive

Decode ARINC 429 Words from  
Receive block

Condor Encode ARINC 429 Words  
for Send

Encode ARINC 429 Words for Send  
block

### GE Fanuc Embedded Systems Shared Memory Support

GE Fanuc 5565 broadcast

GE Fanuc PCI-5565 broadcast  
interrupt block

GE Fanuc 5565 init

GE Fanuc PCI-5565 initialization  
block

GE Fanuc 5565 read

GE Fanuc PCI-5565 read block

GE Fanuc 5565 write  
GE Fanuc PCI-5565

GE Fanuc PCI-5565 write block  
Support for GE Fanuc PCI-5565  
high-speed fiber optic reflective  
memory boards.

## **SBS Technologies Support**

SBS 2510/2500 Broadcast Memory

Support for the SBS 2510/2500  
Broadcast Memory board

SBS Flex-104A

Flex-104A block

SBS Flex/104A PC/104 IP Carrier  
Board

Support for the SBS Flex/104A  
PC/104 IP Carrier board

SBS IP-16ADC

Support for the SBS IP-16ADC board

SBS IP-16ADC Analog Input (A/D)

IP-16ADC Analog Input block

SBS IP-16DAC

Support for the SBS IP-16DAC I/O  
board

SBS IP-16DAC Analog Output (D/A)

IP-16DAC Analog Output block

SBS IP-DAC

Support for the SBS IP-DAC I/O  
board

SBS IP-DAC Analog Output (D/A)

IP-DAC Analog Output block

SBS IP-Digital 24

Support for the SBS IP-Digital 24  
I/O board

SBS IP-Digital 24 Digital Input

IP-Digital 24 Digital Input block

SBS IP-Digital 24 Digital Output

IP-Digital 24 Digital Output block

SBS IP-HiADC

Support for the SBS IP-HiADC board

SBS IP-HiADC Analog Input (A/D)

IP-HiADC Analog Input block

SBS IP-Synchro

Support for the SBS IP-Synchro  
board

SBS IP-Synchro/Resolver

IP-Synchro/Resolver block

SBS IP-UD-IHV-16I80	Support for the SBS IP-UD-IHV-16I80 board
SBS IP-UD-IHV-16I80 Digital Input	SBS IP-UD-IHV-16I80 Digital Input block
SBS IP-UD-IHV-16I80 Digital Output	SBS IP-UD-IHV-16I80 Digital Output block
SBS IP-UD-IHV-8I160	Support for the SBS IP-UD-IHV-8I160 board
SBS IP-UD-IHV-8I160 Digital Input	SBS IP-UD-IHV-8I160 Digital Input block
SBS IP-UD-IHV-8I160 Digital Output	SBS IP-UD-IHV-8I160 Digital Output block
SBS IP-Unidig-E-48 Digital Input	IP-Unidig-E-48 Digital Input block
SBS IP-Unidig-E-48 Digital Output	IP-Unidig-E-48 Digital Output block
SBS P-Unidig-E-48	Support for the SBS P-Unidig-E-48 board
SBS PCI-40A	PCI-40A block
SBS PCI-40A Carrier Board	Support for the SBS PCI-40A Carrier board
SBS PCI-40B	PCI-40B block
SBS PCI-40B Carrier Board	Support for the SBS PCI-40B Carrier board
SBS25x0 init	SBS25x0 init block
SBS25x0 read	SBS25x0 read block
SBS25x0 write	SBS25x0 write block

## MIL-STD-1553 Support

Condor 1553 Create BC Message List	Condor Create Bus Controller (BC) Message List block
Condor 1553 Decode BC Message	Condor 1553 Decode Bus Controller (BC) Message block
Condor 1553 Decode BC Status	Condor 1553 Decode Bus Controller (BC) Status block
Condor 1553 Encode BC Message	Condor 1553 Encode Bus Controller (BC) Message block
Condor 1553 Select BM Message	Condor Select 1553 Bus Monitor (BC) Message block
Condor PCI-1553 Bus Controller Send	Condor PCI-1553 Bus Controller (BC) Send block
Condor PCI-1553 Bus Monitor	Condor PCI-1553 Bus Monitor block
Condor PCI-1553 Initialize	Condor PCI-1553 Initialize block
Condor PCI-1553 RT Initialize	Condor PCI-1553 RT Initialize block
Condor PCI-1553 RT Receive	Condor PCI-1553 Remote Terminal (RT) Receive
Condor PCI-1553 RT Send	Condor PCI-1553 Remote Terminal (RT) Send block
Condor PCI/QPCI-1553	Support for MIL-STD-1553 PCI and QPCI boards.
Condor Q104-1553	Support for MIL-STD-1553 PC104 boards.
Condor Q104-1553 Bus Controller Send	Condor Q104-1553 Bus Controller (BC) Send block
Condor Q104-1553 Bus Monitor	Condor Q104-1553 Bus Monitor block
Condor Q104-1553 Initialize	Condor Q104-1553 Initialize block
Condor Q104-1553 RT Initialize	Condor Q104-1553 RT Initialize block

Condor Q104-1553 RT Receive	Condor Q104-1553 Remote Terminal (RT) Receive block
Condor Q104-1553 RT Send	Condor Q104-1553 Remote Terminal Send (RT) block
Condor QPCI-1553 Bus Controller Send	Condor QPCI-1553 Bus Controller (BC) Send block
Condor QPCI-1553 Bus Monitor	Condor QPCI-1553 Bus Monitor block
Condor QPCI-1553 Initialize	Condor QPCI-1553 Initialize block
Condor QPCI-1553 RT Initialize	Condor QPCI-1553 RT Initialize block
Condor QPCI-1553 RT Receive	Condor QPCI-1553 Remote Terminal (RT) Receive block
Condor QPCI-1553 RT Send	Condor QPCI-1553 Remote Terminal Send (RT) block

## General Standards

General Standards 16AO16 Analog Output	General Standards 16AO16 Analog Output
General Standards 24DSI12 Analog Input	General Standards 24DSI12 Analog Input
General Standards Create Enable Signal	Create Enable Signal block
General Standards PMC-16AO-12	Support for the GS PMC-16AO-12 high-speed analog board.
General Standards PMC-16AO-12 Analog Output	PMC-16AO-12 Analog Output block
General Standards PMC-24DSI12	Support for GS PMC-24DSI12 high-speed analog input board.
General Standards PMC-ADADIO	Support for PMC-ADADIO multifunction ADC/DAC board
General Standards PMC-ADADIO Analog Input (A/D) Read	PMC-ADADIO Analog Input Read block
General Standards PMC-ADADIO Analog Input (A/D) Start	PMC-ADADIO Analog Input Start block
General Standards PMC-ADADIO Analog Output (D/A) Update	PMC-ADADIO Analog Output Update block
General Standards PMC-ADADIO Analog Output Write Block	PMC-ADADIO Analog Output Write block
General Standards PMC-ADADIO Digital Input	PMC-ADADIO Digital Input block
General Standards PMC-ADADIO Digital Output	PMC-ADADIO Digital Output block
General Standards PMC66-16AO16	Support for GS PMC66-16AO16 high-speed analog output board

## **HUMUSOFT**

Humusoft AD 512 Analog Input (A/D)	HUMUSOFT AD 512 Analog Input block
Humusoft AD 512 Analog Output (D/A)	HUMUSOFT AD 512 Analog Output block
Humusoft AD 512 Digital Input	HUMUSOFT AD 512 Digital Input block
Humusoft AD 512 Digital Output	HUMUSOFT AD 512 Digital Output block
Humusoft AD622 Analog Input (A/D)	HUMUSOFT AD622 Analog Input block
Humusoft AD622 Analog Output (D/A)	HUMUSOFT AD622 Analog Output block
Humusoft AD622 Digital Input	HUMUSOFT AD622 Digital Input block
Humusoft AD622 Digital Output	HUMUSOFT AD622 Digital Output block
Humusoft MF624 Analog Input (A/D)	HUMUSOFT MF624 Analog Input block
Humusoft MF624 Analog Output (D/A)	HUMUSOFT MF624 Analog Output block
Humusoft MF624 Counter Input	HUMUSOFT MF624 Counter Input block
Humusoft MF624 Digital Input	HUMUSOFT MF624 Digital Input block
Humusoft MF624 Digital Output	HUMUSOFT MF624 Digital Output block
Humusoft MF624 Encoder Input	HUMUSOFT MF624 Encoder Input block
Humusoft MF624 PWM Output	HUMUSOFT MF624 PWM Output block

## Intel

Softing CAN-AC2-ISA with Intel  
82527 Receive

Softing CAN-AC2-ISA with Intel  
82527 Send

Softing CAN-AC2-ISA with Intel  
82527 Setup

Softing CAN-AC2-ISA with Intel  
82527 Receive block

Softing CAN-AC2-ISA with Intel  
82527 Send block

Softing CAN-AC2-ISA with Intel  
82527 Setup block

## Keithley

Keithley DAS-1800HR Analog Input (A/D)	DAS-1800HR Analog Input block
Keithley DAS-1800HR Digital Input	DAS-1800HR Digital Input block
Keithley DAS-1800HR Digital Output	DAS-1800HR Digital Output block
Keithley KPCI-1801HC Analog Input (A/D)	KPCI-1801HC Analog Input block
Keithley KPCI-1801HC Analog Output (D/A)	KPCI-1801HC Analog Output block
Keithley KPCI-1801HC Digital Input	KPCI-1801HC Digital Input block
Keithley KPCI-1801HC Digital Output	KPCI-1801HC Digital Output block
Keithley KPCI-1802HC Analog Input (A/D)	KPCI-1802HC Analog Input block
Keithley KPCI-1802HC Analog Output (D/A)	KPCI-1802HC Analog Output block
Keithley KPCI-1802HC Digital Input	KPCI-1802HC Digital Input block
Keithley KPCI-1802HC Digital Output	KPCI-1802HC Digital Output block

## Measurement Computing

Measurement Computing CIO-DAS1602/16 Digital Output	CIO-DIO24 Digital Output block
Measurement Computing CIO-DAC08 (/12)	CIO-DDA06 (/12) Digital Output block
Measurement Computing CIO-DAC08 (/12) Analog Output (D/A)	CIO DAS1602/16 Digital Output block
Measurement Computing CIO-DAC08/16	Support for the MC CIO-DAC08 (/12) board
Measurement Computing CIO-DAC08/16 Analog Output (D/A)	CIO-DAC08 (/12) Analog Output (D/A) block
Measurement Computing CIO-DAC16 (/12)	Support for the MC CIO-DAC08/16 I/O board
Measurement Computing CIO-DAC16 Analog Output (D/A)	CIO-DAC08/16 Analog Output (D/A) block
Measurement Computing CIO-DAC16/16	Support for the MC CIO-DAC16 (/12) I/O board
Measurement Computing CIO-DAC16/16 Analog Output (D/A)	CIO-DAC16 Analog Output (D/A) block
Measurement Computing CIO-DAS 1602/16 Digital Input	Support for the MC CIO-DAC16/16 I/O board
Measurement Computing CIO-DAS16/330	CIO-DAC16/16 Analog Output (D/A) block
Measurement Computing CIO-DAS16/330 Analog Input (A/D)	CIO-DAS 1602/16 Digital Input block
	Support for the MC CIO-DAS16/330 I/O board
	CIO-DAS16/330 Analog Input (A/D) block

Measurement Computing CIO-DAS16/JR (/12)	Support for the MC CIO-DAS16/JR (/12) I/O board
Measurement Computing CIO-DAS16/JR (/12) Analog Input (A/D) with EXP	CIO-DAS16/JR (/12) Analog Input with EXP Signal Conditioning Board block
Measurement Computing CIO-DAS16/JR Analog Input (A/D)	CIO-DAS16/JR Analog Input (A/D) block
Measurement Computing CIO-DAS1601/12	Support for the MC CIO-DAS1601/12 I/O board
Measurement Computing CIO-DAS1601/12 Analog Input (A/D)	CIO-DAS1601/12 Analog Input block
Measurement Computing CIO-DAS1601/12 Analog Output (D/A)	CIO-DAS1601/12 Analog Output block
Measurement Computing CIO-DAS1601/12 Digital Input	CIO-DAS1601/12 Digital Input
Measurement Computing CIO-DAS1601/12 Digital Output	CIO-DAS1601/12 Digital Output block
Measurement Computing CIO-DAS1602/12	Support for the MC CIO-DAS1602/12 I/O board
Measurement Computing CIO-DAS1602/12 Analog Input (A/D)	CIO-DAS1602/12 Analog Input block
Measurement Computing CIO-DAS1602/12 Analog Output (D/A)	CIO-DAS1602/12 Analog Output block
Measurement Computing CIO-DAS1602/12 Digital Input	CIO-DAS1602/12 Digital Input block
Measurement Computing CIO-DAS1602/12 Digital Output	CIO-DAS1602/12 Digital Output block

Measurement Computing CIO-DAS1602/16	Support for the MC CIO-DAS1602/16 I/O board
Measurement Computing CIO-DAS1602/16 Analog Input (A/D)	CIO-DAS1602/16 Analog Input block
Measurement Computing CIO-DAS1602/16 Analog Output (D/A)	CIO-DAS1602/16 Analog Output block
Measurement Computing CIO-DAS16JR/16	Support for the MC CIO-DAS16JR/16 I/O board
Measurement Computing CIO-DAS16JR/16 Analog Input (A/D)	CIO-DAS16JR/16 Analog Input block
Measurement Computing CIO-DDA06 (/12)	Support for the MC CIO-DDA06 (/12) I/O board
Measurement Computing CIO-DDA06 (/12) Analog Output (D/A)	CIO-DDA06 (/12) Analog Output block
Measurement Computing CIO-DDA06 (/12) Digital Input	CIO-DDA06 (/12) Digital Input
Measurement Computing CIO-DDA06/16	Support for the MC CIO-DDA06/16 I/O board
Measurement Computing CIO-DDA06/16 Analog Output (D/A)	CIO-DDA06/16 Analog Output block
Measurement Computing CIO-DDA06/16 Digital Input	CIO-DDA06/16 Digital Input block
Measurement Computing CIO-DDA06/16 Digital Output	CIO-DDA06/16 Digital Output block
Measurement Computing CIO-DIO192	Support for the MC CIO-DIO192 I/O board
Measurement Computing CIO-DIO192 Digital Input	CIO-DIO192 Digital Input block

Measurement Computing CIO-DIO192 Digital Output	CIO-DIO192 Digital Output block
Measurement Computing CIO-DIO24	Support for the MC CIO-DIO24 I/O board
Measurement Computing CIO-DIO24 Digital Input	CIO-DIO24 Digital Input block
Measurement Computing CIO-DIO24 Signal Conditioning	CIO-DIO24 Signal Conditioning block
Measurement Computing CIO-DIO24H	Support for the MC CIO-DIO24H I/O board
Measurement Computing CIO-DIO24H Digital Input	CIO-DIO24H Digital Input
Measurement Computing CIO-DIO24H Digital Output	CIO-DIO24H Digital Output
Measurement Computing CIO-DIO48	Support for the MC CIO-DIO48 I/O board
Measurement Computing CIO-DIO48 Digital Input	CIO-DIO48 Digital Input
Measurement Computing CIO-DIO48 Digital Output	CIO-DIO48 Digital Output block
Measurement Computing CIO-DIO48H	Support for the MC CIO-DIO48H I/O board
Measurement Computing CIO-DIO48H Digital Input	CIO-DIO48H Digital Input block
Measurement Computing CIO-DIO48H Digital Output	CIO-DIO48H Digital Output block
Measurement Computing CIO-DIO96	Support for the MC CIO-DIO96 I/O board
Measurement Computing CIO-DIO96 Digital Input	CIO-DIO96 Digital Input block
Measurement Computing CIO-DIO96 Digital Output	CIO-DIO96 Digital Output block

Measurement Computing CIO-DO24DD	Support for the MC CIO-DO24DD I/O board
Measurement Computing CIO-DO24DD Digital Output	CIO-DO24DD Digital Output block
Measurement Computing CIO-PDISO16	Support for the MC CIO-PDISO16 I/O board
Measurement Computing CIO-PDISO16 Digital Input	CIO-PDISO16 Digital Input block
Measurement Computing CIO-PDISO16 Digital Output	CIO-PDISO16 Digital Output block
Measurement Computing CIO-QUAD02	Support for the MC CIO-QUAD02 counting board
Measurement Computing CIO-QUAD02 Incremental Encoder	CIO-QUAD02 Incremental Encoder block
Measurement Computing CIO-QUAD04	Support for the MC CIO-QUAD04 counting board
Measurement Computing CIO-QUAD04 Incremental Encoder	CIO-QUAD04 Incremental Encoder block
Measurement Computing PC104-DAC06 (/12)	Support for the MC PC104-DAC06 (/12) I/O board
Measurement Computing PC104-DAC06 (/12) Analog Output (D/A)	PC104-DAC06 (/12) Analog Output block
Measurement Computing PC104-DAS16JR/12	Support for the MC PC104-DAS16JR/12 I/O board
Measurement Computing PC104-DAS16JR/12 Analog Input (A/D)	PC104-DAS16JR/12 Analog Input block
Measurement Computing PC104-DAS16JR/12 Digital Input	PC104-DAS16JR/12 Digital Input block
Measurement Computing PC104-DAS16JR/12 Digital Output	PC104-DAS16JR/12 Digital Output block

Measurement Computing PC104-DAS16JR/16	Support for the MC PC104–DAS16JR/16 I/O board
Measurement Computing PC104-DAS16JR/16 Analog Input (A/D)	PC104-DAS16JR/16 Analog Input block
Measurement Computing PC104-DAS16JR/16 Digital Input	PC104-DAS16JR/16 Digital Input block
Measurement Computing PC104-DAS16JR/16 Digital Output	PC104-DAS16JR/16 Digital Output
Measurement Computing PC104-DIO48	Support for the MC PC104–DIO48 I/O board
Measurement Computing PC104-DIO48 Digital Input	PC104-DIO48 Digital Input block
Measurement Computing PC104-DIO48 Digital Output	PC104-DIO48 Digital Output block
Measurement Computing PCI-DAC6703	Support for the MC PCI-DAC6703 I/O board
Measurement Computing PCI-DAC6703 Analog Output (D/A)	PCI-DAC6703 Analog Output block
Measurement Computing PCI-DAC6703 Digital Input	PCI-DAC6703 Digital Input block
Measurement Computing PCI-DAC6703 Digital Output	PCI-DAC6703 Digital Output block
Measurement Computing PCI-DAS 1602/12 Digital Input	PCI-DAS 1602/12 Digital Input
Measurement Computing PCI-DAS 1602/16 Digital Input	PCI-DAS 1602/16 Digital Input block
Measurement Computing PCI-DAS-TC	Support for the MC PCI-DAS-TC I/O board
Measurement Computing PCI-DAS-TC Thermocouple	PCI-DAS-TC Thermocouple

Measurement Computing PCI-DAS1002	Support for the MC PCI-DAS1002 I/O board
Measurement Computing PCI-DAS1002 Analog Input (A/D)	PCI-DAS1002 Analog Input block
Measurement Computing PCI-DAS1002 Analog Output (D/A)	PCI-DAS1002 Analog Output block
Measurement Computing PCI-DAS1002 Digital Input	PCI-DAS1002 Digital Input block
Measurement Computing PCI-DAS1002 Digital Output	PCI-DAS1002 Digital Output block
Measurement Computing PCI-DAS1200	Support for the MC PCI-DAS1200 I/O board
Measurement Computing PCI-DAS1200 Analog Input (A/D)	PCI-DAS1200 Analog Input block
Measurement Computing PCI-DAS1200 Analog Output (D/A)	PCI-DAS1200 Analog Output block
Measurement Computing PCI-DAS1200 Digital Input	PCI-DAS1200 Digital Input block
Measurement Computing PCI-DAS1200 Digital Output	PCI-DAS1200 Digital Output block
Measurement Computing PCI-DAS1200/JR	Support for the MC PCI-DAS1200/JR I/O board
Measurement Computing PCI-DAS1200/JR Analog Input (A/D)	PCI-DAS1200/JR Analog Input block
Measurement Computing PCI-DAS1200/JR Digital Input	PCI-DAS1200/JR Digital Input block
Measurement Computing PCI-DAS1200/JR Digital Output	PCI-DAS1200/JR Digital Output block

Measurement Computing PCI-DAS1602/12	Support for the MC PCI-DAS1602/12 I/O board
Measurement Computing PCI-DAS1602/12 Analog Input (A/D)	PCI-DAS1602/12 Analog Input block
Measurement Computing PCI-DAS1602/12 Analog Output (D/A)	PCI-DAS1602/12 Analog Output block
Measurement Computing PCI-DAS1602/12 Digital Output	PCI-DAS1602/12 Digital Output
Measurement Computing PCI-DAS1602/16	Support for the MC PCI-DAS1602/16 I/O board
Measurement Computing PCI-DAS1602/16 Analog Input (A/D)	PCI-DAS1602/16 Analog Input block
Measurement Computing PCI-DAS1602/16 Analog Output (D/A)	PCI-DAS1602/16 Analog Output block
Measurement Computing PCI-DAS1602/16 Digital Output	PCI-DAS1602/16 Digital Output block
Measurement Computing PCI-DDA02/12	Support for the MC PCI-DDA02/12 I/O board
Measurement Computing PCI-DDA02/12 Analog Output (D/A)	PCI-DDA02/12 Analog Output block
Measurement Computing PCI-DDA02/12 Digital Input	PCI-DDA02/12 Digital Input
Measurement Computing PCI-DDA02/12 Digital Output	PCI-DDA02/12 Digital Output
Measurement Computing PCI-DDA02/16	Support for the MC PCI-DDA02/16 I/O board

Measurement Computing PCI-DDA02/16 Analog Output (D/A)	PCI-DDA02/16 Analog Output block
Measurement Computing PCI-DDA02/16 Digital Input	PCI-DDA02/16 Digital Input
Measurement Computing PCI-DDA02/16 Digital Output	PCI-DDA02/16 Digital Output
Measurement Computing PCI-DDA04/12	Support for the MC PCI-DDA04/12 I/O board
Measurement Computing PCI-DDA04/12 Analog Output (D/A)	PCI-DDA04/12 Analog Output block
Measurement Computing PCI-DDA04/12 Digital Input	PCI-DDA04/12 Digital Input
Measurement Computing PCI-DDA04/12 Digital Output	PCI-DDA04/12 Digital Output
Measurement Computing PCI-DDA04/16	Support for the MC PCI-DDA04/16 I/O board
Measurement Computing PCI-DDA04/16 Analog Output (D/A)	PCI-DDA04/16 Analog Output block
Measurement Computing PCI-DDA04/16 Digital Input	PCI-DDA04/16 Digital Input block
Measurement Computing PCI-DDA04/16 Digital Output	PCI-DDA04/16 Digital Output
Measurement Computing PCI-DDA08/12	Support for the MC PCI-DDA08/12 I/O board
Measurement Computing PCI-DDA08/12 Analog Output (D/A)	PCI-DDA08/12 Analog Output block
Measurement Computing PCI-DDA08/12 Digital Input	PCI-DDA08/12 Digital Input block

Measurement Computing PCI-DDA08/12 Digital Output	PCI-DDA08/12 Digital Output block
Measurement Computing PCI-DDA08/16	Support for the MC PCI-DDA08/16 I/O board
Measurement Computing PCI-DDA08/16 Analog Output (D/A)	PCI-DDA08/16 Analog Output block
Measurement Computing PCI-DDA08/16 Digital Input	PCI-DDA08/16 Digital Input
Measurement Computing PCI-DDA08/16 Digital Output	PCI-DDA08/16 Digital Output
Measurement Computing PCI-DIO24	Support for the MC PCI-DIO24 I/O board
Measurement Computing PCI-DIO24 Digital Input	PCI-DIO24 Digital Input
Measurement Computing PCI-DIO24 Digital Output	PCI-DIO24 Digital Output
Measurement Computing PCI-DIO24 Signal Conditioning	PCI-DIO24 Signal Conditioning
Measurement Computing PCI-DIO24H	Support for the MC PCI-DIO24H I/O board
Measurement Computing PCI-DIO24H Digital Input	PCI-DIO24H Digital Input
Measurement Computing PCI-DIO24H Digital Output	PCI-DIO24H Digital Output
Measurement Computing PCI-DIO48H	Support for the MC PCI-DIO48H I/O board
Measurement Computing PCI-DIO48H Digital Input	PCI-DIO48H Digital Input
Measurement Computing PCI-DIO48H Digital Output	PCI-DIO48H Digital Output
Measurement Computing PCI-DIO96	Support for the MC PCI-DIO96 I/O board

Measurement Computing PCI-DIO96 Digital Input	PCI-DIO96 Digital Input block
Measurement Computing PCI-DIO96 Digital Output	PCI-DIO96 Digital Output block
Measurement Computing PCI-DIO96H	Support for the MC PCI-DIO96H I/O board
Measurement Computing PCI-DIO96H Digital Input	PCI-DIO96H Digital Input
Measurement Computing PCI-DIO96H Digital Output	PCI-DIO96H Digital Output
Measurement Computing PCI-DUAL-AC5	Support for the MC PCI-DUAL-AC5 I/O board
Measurement Computing PCI-DUAL-AC5 Digital Input	PCI-DUAL-AC5 Digital Input
Measurement Computing PCI-DUAL-AC5 Digital Output	PCI-DUAL-AC5 Digital Output
Measurement Computing PCI-PDISO16	Support for the MC PCI-PDISO16 I/O board
Measurement Computing PCI-PDISO16 Digital Input	PCI-PDISO16 Digital Input block
Measurement Computing PCI-PDISO16 Digital Output	PCI-PDISO16 Digital Output block
Measurement Computing PCI-PDISO8	Support for the PCI-PDISO8 I/O board
Measurement Computing PCI-PDISO8 Digital Input	PCI-PDISO8 Digital Input block
Measurement Computing PCI-PDISO8 Digital Output	PCI-PDISO8 Digital Output
Measurement Computing PCI-QUAD04	Support for the MC PCI-QUAD04 counting board
Measurement Computing PCI-QUAD04 Incremental Encoder	PCI-QUAD04 Incremental Encoder

Measurement Computing PCIM-DAS 1602/16 Digital Input	PCIM-DAS 1602/16 Digital Input block
Measurement Computing PCIM-DAS1602/16	Support for the MC PCIM-DAS1602/16 I/O board
Measurement Computing PCIM-DAS1602/16 Analog Input (A/D)	PCIM-DAS1602/16 Analog Input block
Measurement Computing PCIM-DAS1602/16 Analog Output (D/A)	PCIM-DAS1602/16 Analog Output block
Measurement Computing PCIM-DAS1602/16 Digital Output	PCIM-DAS1602/16 Digital Output
Measurement Computing PCIM-DDA06/16	Support for the MC PCIM-DDA06/16 I/O board
Measurement Computing PCIM-DDA06/16 Analog Output (D/A)	PCIM-DDA06/16 Analog Output block
Measurement Computing PCIM-DDA06/16 Digital Input	PCIM-DDA06/16 Digital Input
Measurement Computing PCIM-DDA06/16 Digital Output	PCIM-DDA06/16 Digital Output

## MPL

MPL PATI	Support for MPL PATI board.
MPL PATI Digital Input	MPL PATI Digital Input block
MPL PATI Digital Output	MPL PATI Digital Output block
MPL PATI EEPROM Write	MPL PATI EEPROM Write
MPL PATI Incremental Encoder	MPL PATI Incremental Encoder block
MPL PATI PWM generate	MPL PATI PWM generate block
MPL PATI PWM measure	MPL PATI PWM measure
MPL PATI Timebase Setup	MPL PATI Timebase Setup block

## National Instruments

GPIB-232CT-A	Support for GPIB-232CT-A controller
GPIB-232CT-A Send/Receive Block	GPIB-232CT-A Send/Receive Block
GPIB-232CT-A Setup Block	GPIB-232CT-A Setup block
National Instruments AT-AO-10	Support for the National Instruments AT-AO-10
National Instruments AT-AO-10 Analog Output (D/A)	AT-AO-10 Analog Output block
National Instruments AT-AO-6	Support an I/O board with 6 D/A channels and 12 digital I/O lines
National Instruments AT-AO-6 Analog Output (D/A)	AT-AO-6 Analog Output block
National Instruments PC-DIO-24	Support for National Instruments PC-DIO-24 board
National Instruments PC-DIO-24 Digital Input	PC-DIO-24 Digital Input block
National Instruments PC-DIO-24 Digital Output	PC-DIO-24 Digital Output
National Instruments PC-TIO-10	Support for National Instruments PC-TIO-10 board
National Instruments PC-TIO-10 Digital Input	PC-TIO-10 Digital Input block
National Instruments PC-TIO-10 Digital Output	PC-TIO-10 Digital Output block
National Instruments PCI-6011E (formerly PCI-MIO-16XE-50)	Support for National Instruments PCI-6011E board
National Instruments PCI-6011E Analog Input (A/D) (formerly PCI-MIO-16XE-50)	PCI-6011E Analog Input block

National Instruments PCI-6011E Analog Output (D/A) (formerly PCI-MIO-16XE-50)	PCI-6011E Analog Output block
National Instruments PCI-6011E Digital Input (formerly PCI-MIO-16XE-50)	PCI-6011E Digital Input block
National Instruments PCI-6011E Digital Output (formerly PCI-MIO-16XE-50)	PCI-6011E Digital Output block
National Instruments PCI-6011E Pulse Generation (formerly PCI-MIO-16XE-50)	PCI-6011E Pulse Generation block
National Instruments PCI-6011E Pulse Width/Period Measurement (formerly PCI-MIO-16XE-50)	PCI-6011E Pulse Width/Period Measurement
National Instruments PCI-6023E	Support for National Instruments PCI-6023E board
National Instruments PCI-6023E Analog Input (A/D)	PCI-6023E Analog Input block
National Instruments PCI-6023E Digital Input	PCI-6023E Digital Input
National Instruments PCI-6023E Digital Output	PCI-6023E Digital Output
National Instruments PCI-6023E Pulse Generation	PCI-6023E Pulse Generation
National Instruments PCI-6023E Pulse Width/Period Measurement	PCI-6023E Pulse Width/Period Measurement
National Instruments PCI-6024E	Support for National Instruments PCI-6024E board
National Instruments PCI-6024E Analog Input (A/D)	PCI-6024E Analog Input block
National Instruments PCI-6024E Analog Output (D/A)	PCI-6024E Analog Output block

National Instruments PCI-6024E Digital Input	PCI-6024E Digital Input block
National Instruments PCI-6024E Digital Output	PCI-6024E Digital Output
National Instruments PCI-6024E Pulse Generation	PCI-6024E Pulse Generation
National Instruments PCI-6024E Pulse Width/Period Measurement	PCI-6024E Pulse Width/Period Measurement
National Instruments PCI-6025E	Support for National Instruments PCI-6025E board
National Instruments PCI-6025E Analog Input (A/D)	PCI-6025E Analog Input block
National Instruments PCI-6025E Analog Output (D/A)	PCI-6025E Analog Output block
National Instruments PCI-6025E and PCI-6025E 8255 Digital Input	PCI-6025E and PCI-6025E 8255 Digital Input blocks
National Instruments PCI-6025E Digital Output	PCI-6025E Digital Output block
National Instruments PCI-6025E Pulse Generation	PCI-6025E Pulse Generation block
National Instruments PCI-6025E Pulse Width/Period Measurement	PCI-6025E Pulse Width/Period Measurement block
National Instruments PCI-6030E (formerly PCI-MIO-16XE-10)	Support for National Instruments PCI-6030E board
National Instruments PCI-6030E Analog Input (A/D) (formerly PCI-MIO-16XE-10)	PCI-6030E Analog Input block
National Instruments PCI-6030E Analog Output (D/A) (formerly PCI-MIO-16XE-10)	PCI-6030E Analog Output block
National Instruments PCI-6030E Digital Input (formerly PCI-MIO-16XE-10)	PCI-6030E Digital Input block

National Instruments PCI-6030E Digital Output (formerly PCI-MIO-16XE-10)	PCI-6030E Digital Output block
National Instruments PCI-6030E Pulse Generation (formerly PCI-MIO-16XE-10)	PCI-6030E Pulse Generation block
National Instruments PCI-6030E Pulse Width/Period Measurement (formerly PCI-MIO-16XE-10)	PCI-6030E Pulse Width/Period Measurement
National Instruments PCI-6031E	Support for National Instruments PCI-6031E board
National Instruments PCI-6031E Analog Input (A/D)	PCI-6031E Analog Input block
National Instruments PCI-6031E Analog Output (D/A)	PCI-6031E Analog Output block
National Instruments PCI-6031E Digital Input	PCI-6031E Digital Input block
National Instruments PCI-6031E Digital Output	PCI-6031E Digital Output block
National Instruments PCI-6031E Pulse Generation	PCI-6031E Pulse Generation
National Instruments PCI-6031E Pulse Width/Period Measurement	PCI-6031E Pulse Width/Period Measurement block
National Instruments PCI-6040E (formerly PCI-MIO-16E-4)	Support for National Instruments PCI-6040E board
National Instruments PCI-6040E Analog Input (A/D) (formerly PCI-MIO-16E-4)	PCI-6040E Analog Input block
National Instruments PCI-6040E Analog Output (D/A) (formerly PCI-MIO-16E-4)	PCI-6040E Analog Output block

National Instruments PCI-6040E Digital Input (formerly PCI-MIO-16E-4)	PCI-6040E Digital Input block
National Instruments PCI-6040E Digital Output (formerly PCI-MIO-16E-4)	PCI-6040E Digital Output block
National Instruments PCI-6040E Pulse Generation (formerly PCI-MIO-16E-4)	PCI-6040E Pulse Generation block
National Instruments PCI-6040E Pulse Width/Period Measurement (formerly PCI-MIO-16E-4)	PCI-6040E Pulse Width/Period Measurement block
National Instruments PCI-6070E (formerly PCI-MIO-16E-1)	Support for National Instruments PCI-6070E board
National Instruments PCI-6070E Analog Input (A/D) (formerly PCI-MIO-16E-1)	PCI-6070E Analog Input block
National Instruments PCI-6070E Analog Output (D/A) (formerly PCI-MIO-16E-1)	PCI-6070E Analog Output block
National Instruments PCI-6070E Digital Input (formerly PCI-MIO-16E-1)	PCI-6070E Digital Input
National Instruments PCI-6070E Digital Output (formerly PCI-MIO-16E-1)	PCI-6070E Digital Output
National Instruments PCI-6070E Pulse Generation (formerly PCI-MIO-16E-1)	PCI-6070E Pulse Generation
National Instruments PCI-6070E Pulse Width/Period Measurement (formerly PCI-MIO-16E-1)	PCI-6070E Pulse Width/Period Measurement
National Instruments PCI-6071E	Support for National Instruments PCI-6071E board

National Instruments PCI-6071E Analog Input (A/D)	PCI-6071E Analog Input (A/D)
National Instruments PCI-6071E Analog Output (D/A)	PCI-6071E Analog Output block
National Instruments PCI-6071E Digital Input	PCI-6071E Digital Input block
National Instruments PCI-6071E Digital Output	PCI-6071E Digital Output block
National Instruments PCI-6071E Pulse Generation	PCI-6071E Pulse Generation block
National Instruments PCI-6071E Pulse Width/Period Measurement	PCI-6071E Pulse Width/Period Measurement block
National Instruments PCI-6221	Support for National Instruments PCI-6221 board
National Instruments PCI-6221 Analog Input	National Instruments PCI-6221 Analog Input block
National Instruments PCI-6221 Analog Output	National Instruments PCI-6221 Analog Output block
National Instruments PCI-6221 Digital Input	National Instruments PCI-6221 Digital Input block
National Instruments PCI-6221 Digital Output	National Instruments PCI-6221 Digital Output block
National Instruments PCI-6221 Incremental Encoder	National Instruments PCI-6221 Incremental Encoder block
National Instruments PCI-6221 PFI Digital Input	National Instruments PCI-6221 PFI Digital Input block
National Instruments PCI-6221 PFI Digital Output	National Instruments PCI-6221 PFI Digital Output block
National Instruments PCI-6221 PWM Generate	National Instruments PCI-6221 PWM Generate block
National Instruments PCI-6221 PWM Measure	National Instruments PCI-6221 PWM Measure block

National Instruments PCI-6221/37	Support for National Instruments PCI-6221/37 board
National Instruments PCI-6221/37 Analog Input	National Instruments PCI-6221/37 Analog Input block
National Instruments PCI-6221/37 Analog Output	National Instruments PCI-6221/37 Analog Output block
National Instruments PCI-6221/37 Digital Input	National Instruments PCI-6221/37 Digital Input block
National Instruments PCI-6221/37 Digital Output	National Instruments PCI-6221/37 Digital Output block
National Instruments PCI-6221/37 Incremental Encoder	National Instruments PCI-6221/37 Incremental Encoder block
National Instruments PCI-6221/37 PFI Digital Input	National Instruments PCI-6221/37 PFI Digital Input block
National Instruments PCI-6221/37 PFI Digital Output	National Instruments PCI-6221/37 PFI Digital Output block
National Instruments PCI-6221/37 PWM Generate	National Instruments PCI-6221/37 PWM Generate block
National Instruments PCI-6221/37 PWM Measure	National Instruments PCI-6221/37 PWM Measure block
National Instruments PCI-6225	Support for National Instruments PCI-6225 board
National Instruments PCI-6225 Analog Input	National Instruments PCI-6225 Analog Input block
National Instruments PCI-6225 Analog Output	National Instruments PCI-6225 Analog Output block
National Instruments PCI-6225 Digital Input	National Instruments PCI-6225 Digital Input block
National Instruments PCI-6225 Digital Output	National Instruments PCI-6225 Digital Output block
National Instruments PCI-6225 Incremental Encoder	National Instruments PCI-6225 Incremental Encoder block

National Instruments PCI-6225 PFI Digital Input	National Instruments PCI-6225 PFI Digital Input block
National Instruments PCI-6225 PFI Digital Output	National Instruments PCI-6225 PFI Digital Output block
National Instruments PCI-6225 PWM Generate	National Instruments PCI-6225 PWM Generate block
National Instruments PCI-6225 PWM Measure	National Instruments PCI-6225 PWM Measure block
National Instruments PCI-6229	Support for National Instruments PCI-6229 board
National Instruments PCI-6229 Analog Input	National Instruments PCI-6229 Analog Input block
National Instruments PCI-6229 Analog Output	National Instruments PCI-6229 Analog Output block
National Instruments PCI-6229 Digital Input	National Instruments PCI-6229 Digital Input block
National Instruments PCI-6229 Digital Output	National Instruments PCI-6229 Digital Output block
National Instruments PCI-6229 Incremental Encoder	National Instruments PCI-6229 Incremental Encoder block
National Instruments PCI-6229 PFI Digital Input	National Instruments PCI-6229 PFI Digital Input block
National Instruments PCI-6229 PFI Digital Output	National Instruments PCI-6229 PFI Digital Output block
National Instruments PCI-6229 PWM Generate	National Instruments PCI-6229 PWM Generate block
National Instruments PCI-6229 PWM Measure	National Instruments PCI-6229 PWM Measure block
National Instruments PCI-6251	Support for National Instruments PCI-6251 board
National Instruments PCI-6251 Analog Input	National Instruments PCI-6251 Analog Input block

National Instruments PCI-6251 Analog Output	National Instruments PCI-6251 Analog Output block
National Instruments PCI-6251 Digital Input	National Instruments PCI-6251 Digital Input block
National Instruments PCI-6251 Digital Output	National Instruments PCI-6251 Digital Output block
National Instruments PCI-6251 Incremental Encoder	National Instruments PCI-6251 Incremental Encoder block
National Instruments PCI-6251 PFI Digital Input	National Instruments PCI-6251 PFI Digital Input block
National Instruments PCI-6251 PFI Digital Output	National Instruments PCI-6251 PFI Digital Output block
National Instruments PCI-6251 PWM Generate	National Instruments PCI-6251 PWM Generate block
National Instruments PCI-6251 PWM Measure	National Instruments PCI-6251 PWM Measure block
National Instruments PCI-6259	Support for National Instruments PCI-6259 board
National Instruments PCI-6259 Analog Input	National Instruments PCI-6259 Analog Input block
National Instruments PCI-6259 Analog Output	National Instruments PCI-6259 Analog Output block
National Instruments PCI-6259 Digital Input	National Instruments PCI-6259 Digital Input block
National Instruments PCI-6259 Digital Output	National Instruments PCI-6259 Digital Output block
National Instruments PCI-6259 Incremental Encoder	National Instruments PCI-6259 Incremental Encoder block
National Instruments PCI-6259 PFI Digital Input	National Instruments PCI-6259 PFI Digital Input block
National Instruments PCI-6259 PFI Digital Output	National Instruments PCI-6259 PFI Digital Output block

National Instruments PCI-6259 PWM Generate	National Instruments PCI-6259 PWM Generate block
National Instruments PCI-6259 PWM Measure	National Instruments PCI-6259 PWM Measure block
National Instruments PCI-6280	Support for National Instruments PCI-6280 board
National Instruments PCI-6280 Analog Input	National Instruments PCI-6280 Analog Input block
National Instruments PCI-6280 Digital Input	National Instruments PCI-6280 Digital Input block
National Instruments PCI-6280 Digital Output	National Instruments PCI-6280 Digital Output block
National Instruments PCI-6280 Incremental Encoder	National Instruments PCI-6280 Incremental Encoder block
National Instruments PCI-6280 PFI Digital Input	National Instruments PCI-6280 PFI Digital Input block
National Instruments PCI-6280 PFI Digital Output	National Instruments PCI-6280 PFI Digital Output block
National Instruments PCI-6280 PWM Generate	National Instruments PCI-6280 PWM Generate block
National Instruments PCI-6280 PWM Measure	National Instruments PCI-6280 PWM Measure block
National Instruments PCI-6281	Support for National Instruments PCI-6281 board
National Instruments PCI-6281 Analog Input	National Instruments PCI-6281 Analog Input block
National Instruments PCI-6281 Analog Output	National Instruments PCI-6281 Analog Output block
National Instruments PCI-6281 Digital Input	National Instruments PCI-6281 Digital Input block
National Instruments PCI-6281 Digital Output	National Instruments PCI-6281 Digital Output block

National Instruments PCI-6281 Incremental Encoder	National Instruments PCI-6281 Incremental Encoder block
National Instruments PCI-6281 PFI Digital Input	National Instruments PCI-6281 PFI Digital Input block
National Instruments PCI-6281 PFI Digital Output	National Instruments PCI-6281 PFI Digital Output block
National Instruments PCI-6281 PWM Generate	National Instruments PCI-6281 PWM Generate block
National Instruments PCI-6281 PWM Measure	National Instruments PCI-6281 PWM Measure block
National Instruments PCI-6289	Support for National Instruments PCI-6289 board
National Instruments PCI-6289 Analog Input	National Instruments PCI-6289 Analog Input block
National Instruments PCI-6289 Analog Output	National Instruments PCI-6289 Analog Output block
National Instruments PCI-6289 Digital Input	National Instruments PCI-6289 Digital Input block
National Instruments PCI-6289 Digital Output	National Instruments PCI-6289 Digital Output block
National Instruments PCI-6289 Incremental Encoder	National Instruments PCI-6289 Incremental Encoder block
National Instruments PCI-6289 PFI Digital Input	National Instruments PCI-6289 PFI Digital Input block
National Instruments PCI-6289 PFI Digital Output	National Instruments PCI-6289 PFI Digital Output block
National Instruments PCI-6289 PWM Generate	National Instruments PCI-6289 PWM Generate block
National Instruments PCI-6289 PWM Measure	National Instruments PCI-6289 PWM Measure block
National Instruments PCI-6503	Support for National Instruments PCI-6503 board

National Instruments PCI-6503 Digital Input	National Instruments PCI-6503 Digital Input block
National Instruments PCI-6503 Digital Output	National Instruments PCI-6503 Digital Output block
National Instruments PCI-6527	Support for National Instruments PCI-6527 board
National Instruments PCI-6527 Digital Input	National Instruments PCI-6527 Digital Input block
National Instruments PCI-6527 Digital Output	PCI-6527 Digital Output block
National Instruments PCI-6528	Support for National Instruments PCI-6528 board
National Instruments PCI-6528 Digital Input	PCI-6528 Digital Input block
National Instruments PCI-6528 Digital Output	PCI-6528 Digital Output block
National Instruments PCI-6601	Support for National Instruments PCI-6601 board
National Instruments PCI-6601 Armed Pulse Generation	PCI-6601 Armed Pulse Generation block
National Instruments PCI-6601 Digital Input	PCI-6601 Digital Input block
National Instruments PCI-6601 Digital Output	PCI-6601 Digital Output block
National Instruments PCI-6601 Event Counter	PCI-6601 Event Counter block
National Instruments PCI-6601 Incremental Encoder	PCI-6601 Incremental Encoder block
National Instruments PCI-6601 Pulse Generation	PCI-6601 Pulse Generation block
National Instruments PCI-6601 Pulse Width/Period Measurement	PCI-6601 Pulse Width/Period Measurement block

National Instruments PCI-6703	Support for National Instruments PCI-6703 board
National Instruments PCI-6703 Analog Output (D/A)	PCI-6703 Analog Output block
National Instruments PCI-6704	Support for National Instruments PCI-6704 board
National Instruments PCI-6704 Analog Output (D/A)	PCI-6704 Analog Output block
National Instruments PCI-DIO-96	Support for National Instruments PCI-DIO-96 board
National Instruments PCI-DIO-96 Digital Input	PCI-DIO-96 Digital Input block
National Instruments PCI-DIO-96 Digital Output	PCI-DIO-96 Digital Output block
National Instruments PCI/PXI-6052E	Support for National Instruments PCI/PXI-6052E board
National Instruments PCI/PXI-6602	Support for National Instruments PCI/PXI-6602 board
National Instruments PCI/PXI-6602 Armed Pulse Generation	PCI/PXI-6602 Armed Pulse Generation block
National Instruments PCI/PXI-6602 Digital Input	PCI/PXI-6602 Digital Input block
National Instruments PCI/PXI-6602 Digital Output	PCI/PXI-6602 Digital Output block
National Instruments PCI/PXI-6602 Event Counter	PCI-6602 Event Counter block
National Instruments PCI/PXI-6602 Incremental Encoder	PCI/PXI-6602 Incremental Encoder block
National Instruments PCI/PXI-6602 Pulse Generation	PCI/PXI-6602 Pulse Generation block
National Instruments PCI/PXI-6602 Pulse Width/Period Measurement	PCI/PXI-6602 Pulse Width/Period Measurement block

National Instruments PCI/PXI-6711	Support for National Instruments PCI/PXI-6711 board
National Instruments PCI/PXI-6711 Analog Output (D/A)	PCI/PXI-6711 Analog Output block
National Instruments PCI/PXI-6711 Digital Input	PCI/PXI-6711 Digital Input block
National Instruments PCI/PXI-6711 Digital Output	PCI/PXI-6711 Digital Output
National Instruments PCI/PXI-6711 Pulse Generation	PCI/PXI-6711 Pulse Generation block
National Instruments PCI/PXI-6711 Pulse Width/Period Measurement	PCI/PXI-6711 Pulse Width/Period Measurement block
National Instruments PCI/PXI-6713	Support for National Instruments PCI/PXI-6713 board
National Instruments PCI/PXI-6713 Analog Output (D/A)	PCI/PXI-6713 Analog Output block
National Instruments PCI/PXI-6713 Digital Input	PCI/PXI-6713 Digital Input
National Instruments PCI/PXI-6713 Digital Output	PCI/PXI-6713 Digital Output block
National Instruments PCI/PXI-6713 Pulse Generation	PCI/PXI-6713 Pulse Generation block
National Instruments PCI/PXI-6713 Pulse Width/Period Measurement	PCI/PXI-6713 Pulse Width/Period Measurement block
National Instruments PCI/PXI-6731	Support for National Instruments PCI/PXI-6731 board
National Instruments PCI/PXI-6731 Analog Output (D/A)	PCI/PXI-6731 Analog Output block
National Instruments PCI/PXI-6731 Digital Input	PCI/PXI-6731 Digital Input block
National Instruments PCI/PXI-6731 Digital Output	PCI/PXI-6731 Digital Output

National Instruments PCI/PXI-6731 Pulse Generation	PCI/PXI-6731 Pulse Generation block
National Instruments PCI/PXI-6731 Pulse Width/Period Measurement	PCI/PXI-6731 Pulse Width/Period Measurement block
National Instruments PCI/PXI-6733	Support for National Instruments PCI/PXI-6733 board
National Instruments PCI/PXI-6733 Analog Output (D/A)	PCI/PXI-6733 Analog Output block
National Instruments PCI/PXI-6733 Digital Input	PCI/PXI-6733 Digital Input block
National Instruments PCI/PXI-6733 Digital Output	PCI/PXI-6733 Digital Output
National Instruments PCI/PXI-6733 Pulse Generation	PCI/PXI-6733 Pulse Generation block
National Instruments PCI/PXI-6733 Pulse Width/Period Measurement	PCI/PXI-6733 Pulse Width/Period Measurement block
National Instruments PXI-6040E	Support for National Instruments PXI-6040E board
National Instruments PXI-6040E Analog Input (A/D)	PXI-6040E Analog Input block
National Instruments PXI-6040E Analog Output (D/A)	PXI-6040E Analog Output block
National Instruments PXI-6040E Digital Input	PXI-6040E Digital Input block
National Instruments PXI-6040E Digital Output	PXI-6040E Digital Output block
National Instruments PXI-6040E Pulse Generation	PXI-6040E Pulse Generation block
National Instruments PXI-6040E Pulse Width/Period Measurement	PXI-6040E Pulse Width/Period Measurement block
National Instruments PXI-6070E	Support for National Instruments PXI-6070E board

National Instruments PXI-6070E Analog Input (A/D)	PXI-6070E Analog Input block
National Instruments PXI-6070E Analog Output (D/A)	PXI-6070E Analog Output block
National Instruments PXI-6070E Digital Input	PXI-6070E Digital Input block
National Instruments PXI-6070E Digital Output	PXI-6070E Digital Output block
National Instruments PXI-6070E Pulse Generation	PXI-6070E Pulse Generation block
National Instruments PXI-6070E Pulse Width/Period Measurement	PXI-6070E Pulse Width/Period Measurement block
National Instruments PXI-6071E	Support for National Instruments PXI-6071E board
National Instruments PXI-6071E Analog Input (A/D)	PXI-6071E Analog Input block
National Instruments PXI-6071E Analog Output (D/A)	PXI-6071E Analog Output block
National Instruments PXI-6071E Digital Input	PXI-6071E Digital Input block
National Instruments PXI-6071E Digital Output	PXI-6071E Digital Output block
National Instruments PXI-6071E Pulse Generation	PXI-6071E Pulse Generation block
National Instruments PXI-6071E Pulse Width/Period Measurement	PXI-6071E Pulse Width/Period Measurement block
National Instruments PXI-6508	Support for National Instruments PXI-6508 board
National Instruments PXI-6508 Digital Input	PXI-6508 Digital Input block
National Instruments PXI-6508 Digital Output	PXI-6508 Digital Output block

National Instruments PXI-6527	Support for National Instruments PXI-6527 board
National Instruments PXI-6527 Digital Input	PXI-6527 Digital Input block
National Instruments PXI-6527 Digital Output	PXI-6527 Digital Output block
National Instruments PXI-6704	Support for National Instruments PXI-6704 board
National Instruments PXI-6704 Analog Output (D/A)	PXI-6704 Analog Output block
National Instruments PXI/PCI-6052E Analog Input (A/D)	PXI/PCI-6052E Analog Input block
National Instruments PXI/PCI-6052E Analog Output (D/A)	PXI/PCI-6052E Analog Output block
National Instruments PXI/PCI-6052E Digital Input	PXI/PCI-6052E Digital Input block
National Instruments PXI/PCI-6052E Digital Output	PXI/PCI-6052E Digital Output block
National Instruments PXI/PCI-6052E Pulse Generation	PXI/PCI-6052E Pulse Generation
National Instruments PXI/PCI-6052E Pulse Width/Period Measurement	PXI/PCI-6052E Pulse Width/Period Measurement

## North Atlantic Industries

NAII (Apex) 73LD3	Support for NAII (Apex) 73LD3 positioning sensor I/O board-
NAII (Apex) 73SD3	Support for NAII (Apex) 73SD3 positioning sensor I/O board-
NAII (Apex) 76CL1	Support for NAII (Apex) 76CL1 positioning sensor I/O board-
NAII (Apex) 76CS1	Support for NAII (Apex) 76CS1 positioning sensor I/O board-
NAII (Apex) 76LD1	Support for NAII (Apex) 76LD1 positioning sensor I/O board-
NAII 73LD3 LVDT/RVDT Converter	73LD3 LVDT/RVDT Converter block
NAII 73SD3 Synchro/Resolver	NAII 73SD3 Synchro/Resolver block
NAII 76CL1 D/L	NAII 76CL1 D/L block
NAII 76CL1 L/D	Support for NAII 76CL1 L/D positioning sensor I/O board-
NAII 76CS1 D/S	NAII 76CS1 D/S block
NAII 76CS1 S/D	NAII 76CS1 S/D block
NAII 76LD1 L/D	NAII 76LD1 L/D block
NAII PC-12SD (PC-77SD1)	Support for NAII PC-12SD (PC-77SD1) positioning sensor I/O board-
NAII PC-12SD (PC-77SD1) Synchro/Resolver	PC-12SD (PC-77SD1) Synchro/Resolver block

## Philips

	Softing CAN-AC2-PCI with SJA1000 Setup block
Softing CAN-AC2-104 with Philips SJA1000 FIFO Read	Softing CAN-AC2-104 with Philips SJA1000 FIFO Read block
Softing CAN-AC2-104 with Philips SJA1000 FIFO Read Filter	Softing CAN-AC2-104 with Philips SJA1000 FIFO Read Filter Block
Softing CAN-AC2-104 with Philips SJA1000 FIFO Read RCV Level	Softing CAN-AC2-104 with Philips SJA1000 FIFO Read RCV Level block
Softing CAN-AC2-104 with Philips SJA1000 FIFO Read XMT Level	Softing CAN-AC2-104 with Philips SJA1000 FIFO Read XMT Level block
Softing CAN-AC2-104 with Philips SJA1000 FIFO Reset RCV	Softing CAN-AC2-104 with Philips SJA1000 FIFO Reset RCV block
Softing CAN-AC2-104 with Philips SJA1000 FIFO Reset XMT	Softing CAN-AC2-104 with Philips SJA1000 FIFO Reset XMT block
Softing CAN-AC2-104 with Philips SJA1000 FIFO Setup	Softing CAN-AC2-104 with Philips SJA1000 FIFO Setup block
Softing CAN-AC2-104 with Philips SJA1000 FIFO Write	Softing CAN-AC2-104 with Philips SJA1000 FIFO Write block
Softing CAN-AC2-104 with SJA1000 Receive	Softing CAN-AC2-104 with SJA1000 Receive block
Softing CAN-AC2-104 with SJA1000 Send	Softing CAN-AC2-104 with SJA1000 Send block
Softing CAN-AC2-104 with SJA1000 Setup	Softing CAN-AC2-104 with SJA1000 Setup block
Softing CAN-AC2-ISA with Philips PCA 82C200 Receive	Softing CAN-AC2-ISA with Philips PCA 82C200 Receive block
Softing CAN-AC2-ISA with Philips PCA 82C200 Send	Softing CAN-AC2-ISA with Philips PCA 82C200 Send block

Softing CAN-AC2-ISA with Philips PCA 82C200 Setup	Softing CAN-AC2-ISA with Philips PCA 82C200 Setup block
Softing CAN-AC2-PCI with Philips SJA1000 FIFO Read	Softing CAN-AC2-PCI with Philips SJA1000 FIFO Read block
Softing CAN-AC2-PCI with Philips SJA1000 FIFO Read Filter	Softing CAN-AC2-PCI with Philips SJA1000 FIFO Read Filter block
Softing CAN-AC2-PCI with Philips SJA1000 FIFO Read RCV Level	Softing CAN-AC2-PCI with Philips SJA1000 FIFO Read RCV Level block
Softing CAN-AC2-PCI with Philips SJA1000 FIFO Read XMT Level	Softing CAN-AC2-PCI with Philips SJA1000 FIFO Read XMT Level block
Softing CAN-AC2-PCI with Philips SJA1000 FIFO Reset RCV	Softing CAN-AC2-PCI with Philips SJA1000 FIFO Reset RCV block
Softing CAN-AC2-PCI with Philips SJA1000 FIFO Reset XMT	Softing CAN-AC2-PCI with Philips SJA1000 FIFO Reset XMT block
Softing CAN-AC2-PCI with Philips SJA1000 FIFO Setup	Softing CAN-AC2-PCI with Philips SJA1000 FIFO Setup block
Softing CAN-AC2-PCI with Philips SJA1000 FIFO Write	Softing CAN-AC2-PCI with Philips SJA1000 FIFO Write Driver block
Softing CAN-AC2-PCI with SJA1000 Receive	Softing CAN-AC2-PCI with SJA1000 Receive block
Softing CAN-AC2-PCI with SJA1000 Send	Softing CAN-AC2-PCI with SJA1000 Send block

## Quanser

Quanser Q4	Support for the Quanser Q4 multifunction measurement and control board.
Quanser Q4 Analog Input	Q4 Analog Input block
Quanser Q4 Analog Output	Q4 Analog Output block
Quanser Q4 Counter	Q4 Counter block
Quanser Q4 Digital Input	Q4 Digital Input block
Quanser Q4 Digital Output	Q4 Digital Output block
Quanser Q4 Incremental Encoder	Q4 Incremental Encoder block
Quanser Q8	Support for the Quanser Q8 multifunction measurement and control board.
Quanser Q8 Analog Input	Q8 Analog Input block
Quanser Q8 Analog Output	Q8 Analog Output block
Quanser Q8 Counter	Q8 Counter block
Quanser Q8 Digital Input	Q8 Digital Input block
Quanser Q8 Digital Output	Q8 Digital Output block
Quanser Q8 Incremental Encoder	Q8 Incremental Encoder block

## Quatech

ASCII Decode V2	ASCII Decode V2
ASCII Encode/Decode (Composite)	ASCII Encode/Decode
FIFO Read Binary (Composite)	FIFO Read Binary
FIFO Read HDRS (Composite)	FIFO Read HDRS block
FIFO Read/Write (Composite)	FIFO Read/Write
Modem Control (Composite)	Modem Control block
Modem Status (Composite)	Modem Status block
RS-232/RS-422/RS-485 Send/Receive (Composite)	RS-232/RS-422/RS-485 Send/Receive block
RS232 State (Composite)	RS232 State block

## Real Time Devices

Real Time Devices DM6420	Support for the RTD DM6420 I/O board.
Real Time Devices DM6420 Analog Input (A/D)	DM6420 Analog Input block
Real Time Devices DM6420 Analog Output (D/A)	DM6420 Analog Output block
Real Time Devices DM6420 Digital Input	DM6420 Digital Input block
Real Time Devices DM6420 Digital Output	DM6420 Digital Output block
Real Time Devices DM6430	Support for the RTD DM6430 I/O board.
Real Time Devices DM6430 Analog Input (A/D)	DM6430 Analog Input block
Real Time Devices DM6430 Analog Output (D/A)	DM6430 Analog Output block
Real Time Devices DM6430 Digital Input	DM6430 Digital Input block
Real Time Devices DM6430 Digital Output	DM6430 Digital Output
Real Time Devices DM6604	Support for the RTD DM6604 I/O board.
Real Time Devices DM6604 Analog Output (D/A)	DM6604 Analog Output block
Real Time Devices DM6604 Digital Input	DM6604 Digital Input block
Real Time Devices DM6604 Digital Output	DM6604 Digital Output block
Real Time Devices DM6804	Support for the RTD DM6804 I/O board.

Real Time Devices DM6804 Digital Input	DM6804 Digital Input block
Real Time Devices DM6804 Digital Output	DM6804 Digital Output block
Real Time Devices DM6814	Support for the RTD DM6814 I/O board.
Real Time Devices DM6814 Digital Input	DM6814 Digital Input block
Real Time Devices DM6814 Digital Output	DM6814 Digital Output block
Real Time Devices DM6814 Incremental Encoder	DM6814 Incremental Encoder block
Real Time Devices DM6816	Support for the RTD DM6816 I/O board.
Real Time Devices DM6816 PWM	DM6816 PWM block
Real Time Devices DM7420	Support for the RTD DM7420 I/O board.
Real Time Devices DM7420 Analog Input (A/D)	DM7420 Analog Input block
Real Time Devices DM7420 Digital Input	DM7420 Digital Input block
Real Time Devices DM7420 Digital Output	DM7420 Digital Output block

## **Sensoray**

Sensoray 526	Support for the Sensoray 526 measurement and control board
Sensoray526 AD	Sensoray526 AD block
Sensoray526 DA	Sensoray526 DA block
Sensoray526 DI	Sensoray526 DI block
Sensoray526 DO	Sensoray526 DO block
Sensoray526 Dual AD	Sensoray526 Dual AD block
Sensoray526 Dual DA	Sensoray526 Dual DA block
Sensoray526 Encoder Input	Sensoray526 Encoder Input block

## Softing

	Softing CAN-AC2-PCI with SJA1000 Setup block
Softing CAN-AC2-104 with Philips SJA1000 FIFO Read	Softing CAN-AC2-104 with Philips SJA1000 FIFO Read block
Softing CAN-AC2-104 with Philips SJA1000 FIFO Read Filter	Softing CAN-AC2-104 with Philips SJA1000 FIFO Read Filter Block
Softing CAN-AC2-104 with Philips SJA1000 FIFO Read RCV Level	Softing CAN-AC2-104 with Philips SJA1000 FIFO Read RCV Level block
Softing CAN-AC2-104 with Philips SJA1000 FIFO Read XMT Level	Softing CAN-AC2-104 with Philips SJA1000 FIFO Read XMT Level block
Softing CAN-AC2-104 with Philips SJA1000 FIFO Reset RCV	Softing CAN-AC2-104 with Philips SJA1000 FIFO Reset RCV block
Softing CAN-AC2-104 with Philips SJA1000 FIFO Reset XMT	Softing CAN-AC2-104 with Philips SJA1000 FIFO Reset XMT block
Softing CAN-AC2-104 with Philips SJA1000 FIFO Setup	Softing CAN-AC2-104 with Philips SJA1000 FIFO Setup block
Softing CAN-AC2-104 with Philips SJA1000 FIFO Write	Softing CAN-AC2-104 with Philips SJA1000 FIFO Write block
Softing CAN-AC2-104 with SJA1000 Receive	Softing CAN-AC2-104 with SJA1000 Receive block
Softing CAN-AC2-104 with SJA1000 Send	Softing CAN-AC2-104 with SJA1000 Send block
Softing CAN-AC2-104 with SJA1000 Setup	Softing CAN-AC2-104 with SJA1000 Setup block
Softing CAN-AC2-ISA with Intel 82527 Receive	Softing CAN-AC2-ISA with Intel 82527 Receive block
Softing CAN-AC2-ISA with Intel 82527 Send	Softing CAN-AC2-ISA with Intel 82527 Send block

Softing CAN-AC2-ISA with Intel 82527 Setup	Softing CAN-AC2-ISA with Intel 82527 Setup block
Softing CAN-AC2-ISA with Philips PCA 82C200 Receive	Softing CAN-AC2-ISA with Philips PCA 82C200 Receive block
Softing CAN-AC2-ISA with Philips PCA 82C200 Send	Softing CAN-AC2-ISA with Philips PCA 82C200 Send block
Softing CAN-AC2-ISA with Philips PCA 82C200 Setup	Softing CAN-AC2-ISA with Philips PCA 82C200 Setup block
Softing CAN-AC2-PCI with Philips SJA1000 FIFO Read	Softing CAN-AC2-PCI with Philips SJA1000 FIFO Read block
Softing CAN-AC2-PCI with Philips SJA1000 FIFO Read Filter	Softing CAN-AC2-PCI with Philips SJA1000 FIFO Read Filter block
Softing CAN-AC2-PCI with Philips SJA1000 FIFO Read RCV Level	Softing CAN-AC2-PCI with Philips SJA1000 FIFO Read RCV Level block
Softing CAN-AC2-PCI with Philips SJA1000 FIFO Read XMT Level	Softing CAN-AC2-PCI with Philips SJA1000 FIFO Read XMT Level block
Softing CAN-AC2-PCI with Philips SJA1000 FIFO Reset RCV	Softing CAN-AC2-PCI with Philips SJA1000 FIFO Reset RCV block
Softing CAN-AC2-PCI with Philips SJA1000 FIFO Reset XMT	Softing CAN-AC2-PCI with Philips SJA1000 FIFO Reset XMT block
Softing CAN-AC2-PCI with Philips SJA1000 FIFO Setup	Softing CAN-AC2-PCI with Philips SJA1000 FIFO Setup block
Softing CAN-AC2-PCI with Philips SJA1000 FIFO Write	Softing CAN-AC2-PCI with Philips SJA1000 FIFO Write Driver block
Softing CAN-AC2-PCI with SJA1000 Receive	Softing CAN-AC2-PCI with SJA1000 Receive block
Softing CAN-AC2-PCI with SJA1000 Send	Softing CAN-AC2-PCI with SJA1000 Send block

# Speedgoat

Speedgoat IO301	Support for the Speedgoat IO301 FPGA board
Speedgoat IO301 PCI Read	Speedgoat IO301 PCI Read block
Speedgoat IO301 PCI Setup	Speedgoat IO301 PCI Setup block
Speedgoat IO301 PCI Write	Speedgoat IO301 PCI Write block
Speedgoat IO302	Support for the Speedgoat IO302 FPGA board
Speedgoat IO302 PCI Read	Speedgoat IO302 PCI Read block
Speedgoat IO302 PCI Setup	Speedgoat IO302 PCI Setup block
Speedgoat IO302 PCI Write	Speedgoat IO302 PCI Write block
Speedgoat IO303	Support for the Speedgoat IO303 FPGA board
Speedgoat IO303 PCI Read	Speedgoat IO303 PCI Read block
Speedgoat IO303 PCI Setup	
Speedgoat IO303 PCI Write	Speedgoat IO303 PCI Write block
Speedgoat IO311	Support for the Speedgoat IO311 FPGA board
Speedgoat IO311 PCI Read	Speedgoat IO311 PCI Read block
Speedgoat IO311 PCI Setup	Speedgoat IO311 PCI Setup block
Speedgoat IO311 PCI Write	Speedgoat IO311 PCI Write block
Speedgoat IO312	Support for the Speedgoat IO312 FPGA board
Speedgoat IO312 PCI Read	Speedgoat IO312 PCI Read block
Speedgoat IO312 PCI Setup	Speedgoat IO312 PCI Setup block
Speedgoat IO312 PCI Write	Speedgoat IO312 PCI Write block
Speedgoat IO313	Support for the Speedgoat IO313 FPGA board
Speedgoat IO313 PCI Read	Speedgoat IO313 PCI Read block

Speedgoat IO313 PCI Setup	Speedgoat IO313 PCI Setup block
Speedgoat IO313 PCI Write	Speedgoat IO313 PCI Write block
Speedgoat IO314	Support for the Speedgoat IO314 FPGA board
Speedgoat IO314 PCI Read	Speedgoat IO314 PCI Read block
Speedgoat IO314 PCI Setup	Speedgoat IO314 PCI Setup block
Speedgoat IO314 PCI Write	Speedgoat IO314 PCI Write block
Speedgoat IO325	Support for the Speedgoat IO325 FPGA board
Speedgoat IO325 A/D Calibration	
Speedgoat IO325 PCI Read	Speedgoat IO325 PCI Read block
Speedgoat IO325 PCI Setup	Speedgoat IO325 PCI Setup block
Speedgoat IO325 PCI Write	Speedgoat IO325 PCI Write block

## Texas Instruments

TI DM642 RX

TI DM642 TX

TI DM642 receive block

TI DM642 transmit block

## United Electronic Industries (UEI)

UEI PD2-AO Analog Output (D/A)	PD2-AO Analog Output block
UEI PD2-AO Digital Input	PD2-AO Digital Input block
UEI PD2-AO Digital Output	PD2-AO Digital Output block
UEI PD2-AO Series	Support for the UEI PD2-AO Series I/O boards
UEI PD2-DIO-128	Support for the UEI PD2-DIO-128 Series I/O boards
UEI PD2-DIO-128 Digital Input	PD2-DIO-128 Digital Input block
UEI PD2-DIO-128 Digital Output	PD2-DIO-128 Digital Output block
UEI PD2-DIO-128i	Support for the UEI PD2-DIO-128i Series I/O boards
UEI PD2-DIO-128i Digital Input	PD2-DIO-128i Digital Input block
UEI PD2-DIO-128i Digital Output	PD2-DIO-128i Digital Output block
UEI PD2-DIO-64	Support for the UEI PD2-DIO-64 series I/O boards
UEI PD2-DIO-64 Digital Input	PD2-DIO-64 Digital Input block
UEI PD2-DIO-64 Digital Output	PD2-DIO-64 Digital Output block
UEI PD2-MF 12-Bit Series	Support for the UEI PD2-MF 12-Bit Series I/O boards
UEI PD2-MF 12-Bit Series Analog Input (A/D)	PD2-MF 12-Bit Series Analog Input block
UEI PD2-MF 12-Bit Series Analog Output (D/A)	PD2-MF 12-Bit Series Analog Output block
UEI PD2-MF 12-Bit Series Digital Input	PD2-MF 12-Bit Series Digital Input block
UEI PD2-MF 12-Bit Series Digital Output	PD2-MF 12-Bit Series Digital Output block
UEI PD2-MF 12-Bit Series Frame Analog Input	PD2-MF 12-Bit Series Frame Analog Input block

UEI PD2-MF 14-Bit Series	Support for the UEI PD2-MF 14-Bit Series I/O boards
UEI PD2-MF 14-Bit Series Analog Input (A/D)	PD2-MF 14-Bit Series Analog Input block
UEI PD2-MF 14-Bit Series Analog Output (D/A)	PD2-MF 14-Bit Series Analog Output block
UEI PD2-MF 14-Bit Series Digital Input	PD2-MF 14-Bit Series Digital Input block
UEI PD2-MF 14-Bit Series Digital Output	PD2-MF 14-Bit Series Digital Output block
UEI PD2-MF 14-Bit Series Frame Analog Input	PD2-MF 14-Bit Series Frame Analog Input block
UEI PD2-MF 16-Bit Series	Support for the UEI PD2-MF 16-Bit Series I/O boards
UEI PD2-MF 16-Bit Series Analog Input (A/D)	PD2-MF 16-Bit Series Analog Input block
UEI PD2-MF 16-Bit Series Analog Output (D/A)	PD2-MF 16-Bit Series Analog Output block
UEI PD2-MF 16-Bit Series Digital Input	PD2-MF 16-Bit Series Digital Input block
UEI PD2-MF 16-Bit Series Digital Output	PD2-MF 16-Bit Series Digital Output block
UEI PD2-MF 16-Bit Series Frame Analog Input	PD2-MF 16-Bit Series Frame Analog Input block
UEI PD2-MFS 12-Bit Series	Support for the UEI PD2-MFS 12-Bit Series I/O boards
UEI PD2-MFS 12-Bit Series Analog Input (A/D)	PD2-MFS 12-Bit Series Analog Input block
UEI PD2-MFS 12-Bit Series Analog Output (D/A)	PD2-MFS 12-Bit Series Analog Output block
UEI PD2-MFS 12-Bit Series Digital Input	PD2-MFS 12-Bit Series Digital Input block

UEI PD2-MFS 12-Bit Series Digital Output	PD2-MFS 12-Bit Series Digital Output block
UEI PD2-MFS 12-Bit Series Frame Analog Input	PD2-MFS 12-Bit Series Frame Analog Input block
UEI PD2-MFS 14-Bit Series	Support for the EI PD2-MFS 14-Bit Series I/O boards
UEI PD2-MFS 14-Bit Series Analog Input (A/D)	PD2-MFS 14-Bit Series Analog Input block
UEI PD2-MFS 14-Bit Series Analog Output (D/A)	PD2-MFS 14-Bit Series Analog Output block
UEI PD2-MFS 14-Bit Series Digital Input	PD2-MFS 14-Bit Series Digital Input block
UEI PD2-MFS 14-Bit Series Digital Output	PD2-MFS 14-Bit Series Digital Output block
UEI PD2-MFS 14-Bit Series Frame Analog Input	PD2-MFS 14-Bit Series Frame Analog Input block
UEI PD2-MFS 16-Bit Series	Support for the UEI PD2-MFS 16-Bit Series I/O boards
UEI PD2-MFS 16-Bit Series Analog Input (A/D)	PD2-MFS 16-Bit Series Analog Input block
UEI PD2-MFS 16-Bit Series Analog Output (D/A)	PD2-MFS 16-Bit Series Analog Output block
UEI PD2-MFS 16-Bit Series Digital Input	PD2-MFS 16-Bit Series Digital Input block
UEI PD2-MFS 16-Bit Series Digital Output	PD2-MFS 16-Bit Series Digital Output
UEI PD2-MFS 16-Bit Series Frame Analog Input	PD2-MFS 16-Bit Series Frame Analog Input block
UEI PDL-DIO-64	Support for the UEI PDL-DIO-64 Series I/O boards
UEI PDL-DIO-64 Digital Input	PDL-DIO-64 Digital Input block
UEI PDL-DIO-64 Digital Output	PDL-DIO-64 Digital Output block

UEI PDXI-AO Analog Output (D/A)	PDXI-AO Analog Output block
UEI PDXI-AO Digital Input	PDXI-AO Digital Input block
UEI PDXI-AO Digital Output	PDXI-AO Digital Output block
UEI PDXI-AO Series	Support for the UEI PDXI-AO Series I/O boards
UEI PDXI-DIO-64	Support for the UEI PDXI-DIO-64 Series I/O boards
UEI PDXI-DIO-64 Digital Input	PDXI-DIO-64 Digital Input block
UEI PDXI-DIO-64 Digital Output	PDXI-DIO-64 Digital Output block
UEI PDXI-MF 12-Bit Series	Support for the UEI PDXI-MF 12-Bit Series I/O boards
UEI PDXI-MF 12-Bit Series Analog Input (A/D)	PDXI-MF 12-Bit Series Analog Input block
UEI PDXI-MF 12-Bit Series Analog Output (D/A)	PDXI-MF 12-Bit Series Analog Output block
UEI PDXI-MF 12-Bit Series Digital Input	PDXI-MF 12-Bit Series Digital Input block
UEI PDXI-MF 12-Bit Series Digital Output	PDXI-MF 12-Bit Series Digital Output block
UEI PDXI-MF 12-Bit Series Frame Analog Input	PDXI-MF 12-Bit Series Frame Analog Input block
UEI PDXI-MF 14-Bit Series	Support for the UEI PDXI-MF 14-Bit Series I/O boards
UEI PDXI-MF 14-Bit Series Analog Input (A/D)	PDXI-MF 14-Bit Series Analog Input block
UEI PDXI-MF 14-Bit Series Analog Output (D/A)	PDXI-MF 14-Bit Series Analog Output block
UEI PDXI-MF 14-Bit Series Digital Input	PDXI-MF 14-Bit Series Digital Input block
UEI PDXI-MF 14-Bit Series Digital Output	PDXI-MF 14-Bit Series Digital Output block

UEI PDXI-MF 14-Bit Series Frame Analog Input	PDXI-MF 14-Bit Series Frame Analog Input block
UEI PDXI-MF 16-Bit Series	Support for the UEI PDXI-MF 16-Bit Series I/O boards
UEI PDXI-MF 16-Bit Series Analog Input (A/D)	PDXI-MF 16-Bit Series Analog Input block
UEI PDXI-MF 16-Bit Series Analog Output (D/A)	PDXI-MF 16-Bit Series Analog Output block
UEI PDXI-MF 16-Bit Series Digital Input	PDXI-MF 16-Bit Series Digital Input block
UEI PDXI-MF 16-Bit Series Digital Output	PDXI-MF 16-Bit Series Digital Output block
UEI PDXI-MF 16-Bit Series Frame Analog Input	PDXI-MF 16-Bit Series Frame Analog Input block
UEI PDXI-MFS 12-Bit Series	Support for the UEI PDXI-MFS 12-Bit Series I/O boards
UEI PDXI-MFS 12-Bit Series Analog Input (A/D)	PDXI-MFS 12-Bit Series Analog Input block
UEI PDXI-MFS 12-Bit Series Analog Output (D/A)	PDXI-MFS 12-Bit Series Analog Output block
UEI PDXI-MFS 12-Bit Series Digital Input	PDXI-MFS 12-Bit Series Digital Input block
UEI PDXI-MFS 12-Bit Series Digital Output	PDXI-MFS 12-Bit Series Digital Output block
UEI PDXI-MFS 12-Bit Series Frame Analog Input	PDXI-MFS 12-Bit Series Frame Analog Input block
UEI PDXI-MFS 14-Bit Series	Support for the UEI PDXI-MFS 14-Bit Series I/O boards
UEI PDXI-MFS 14-Bit Series Analog Input (A/D)	PDXI-MFS 14-Bit Series Analog Input block
UEI PDXI-MFS 14-Bit Series Analog Output (D/A)	PDXI-MFS 14-Bit Series Analog Output block

UEI PDXI-MFS 14-Bit Series Digital Input	PDXI-MFS 14-Bit Series Digital Input block
UEI PDXI-MFS 14-Bit Series Digital Output	PDXI-MFS 14-Bit Series Digital Output block
UEI PDXI-MFS 14-Bit Series Frame Analog Input	PDXI-MFS 14-Bit Series Frame Analog Input block
UEI PDXI-MFS 16-Bit Series	Support for the UEI PDXI-MFS 16-Bit Series I/O boards
UEI PDXI-MFS 16-Bit Series Analog Input (A/D)	PDXI-MFS 16-Bit Series Analog Input block
UEI PDXI-MFS 16-Bit Series Analog Output (D/A)	PDXI-MFS 16-Bit Series Analog Output block
UEI PDXI-MFS 16-Bit Series Digital Input	PDXI-MFS 16-Bit Series Digital Input block
UEI PDXI-MFS 16-Bit Series Digital Output	PDXI-MFS 16-Bit Series Digital Output block
UEI PDXI-MFS 16-Bit Series Frame Analog Input	PDXI-MFS 16-Bit Series Frame Analog Input block

## **Versallogic**

Versallogic VSBC-6	Support for Versallogic VSBC-6 single-board computer
Versallogic VSBC-6 Analog Input (A/D)	VSBC-6 Analog Input block
Versallogic VSBC-6 Digital Input	VSBC-6 Digital Input block
Versallogic VSBC-6 Digital Output	VSBC-6 Digital Output block
Versallogic VSBC-6 Watch Dog	VSBC-6 Watch Dog block

# Serial Communications Support

---

- “Serial Drivers” on page 4-2
- “RS-232/422/485 Drivers (Composite)” on page 4-6
- “Boards and Blocks — Alphabetical List” on page 4-22

## Serial Drivers

In this section...
“RS-232 I/O Communication” on page 4-2
“Hardware Connections for RS-232” on page 4-4
“Host and Target Computer Communication” on page 4-4

### RS-232 I/O Communication

The xPC Target software interfaces the target computer 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.

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

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

For the target computer serial ports, the software 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.

The software 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, Fastcom: 422/2-PCI-335 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 following models:

- EMM-XT, which can support two ports each for RS-232 and RS-422.
- EMM-4485–XT, which supports four ports for RS-422 and RS-485.
- EMM-4M-XT, which supports four ports for RS-232, RS-422, and RS-485.

See the manufacturer documentation for details on how to configure your jumpers.

---

**Note** When you configure the board for the RS-485 protocol, configure your model to manually control the RTS bit to turn the transmitter on and off. You must use a convention, such as counting sample time steps, to decide when the model should turn off the transmitter after sending data.

---

The xPC Target block library provides a set of functionally similar drivers for these boards. See “RS-232/422/485 Simulink Block Reference” on page 4-14 for a description of the driver blocks that support the different protocols.

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

- The composite drivers support RS-232 I/O for the target computer 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. The xPC Target block library 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 composite drivers also include internal drivers, which you might need to access if you need to modify the composite subsystems for your use. Note that you normally should not use the internal drivers, described in Appendix A, “Serial Communications Support with Internal Drivers”.

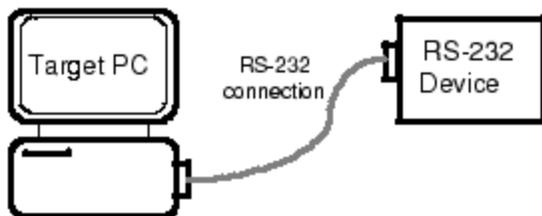
- The obsolete drivers support RS-232 I/O only for the target computer serial ports. These drivers support synchronous, asynchronous, and binary (asynchronous) communication mode. The xPC Target software uses a

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. See “Obsolete RS-232 Drivers” on page 5-2 for a description of these drivers.

### Hardware Connections for RS-232

The xPC Target software supports serial communication with the COM1 and COM2 ports on the target computer.

Your target applications can use these RS-232 ports as I/O devices. Typically, the target computer 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 Computer Communication

---

**Note** RS-232 Host-Target communication mode will be removed in a future release. Use TCP/IP instead.

---

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

For example, if the target computer uses COM1 for the communication with the host computer, 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.

In addition, 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.

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

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

In this section...
“Introduction” on page 4-6
“Adding RS-232 Blocks” on page 4-7
“Building and Running the Target Application (Composite)” on page 4-13
“RS-232/422/485 Simulink Block Reference” on page 4-14

### Introduction

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.

The xPC Target software supports the target computer 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 computer serial port, Quatech RS-232/422/485, Diamond Systems RS-232, and Commtech Fastcom drivers enables you to do so. See Appendix A, “Serial Communications Support with Internal Drivers” 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 two independent RS-422 channels.
- The Fastcom 422/2-PCI board can handle baud rates up to 1.5 megabits/second.
- The Fastcom 422/2-PCI board hardware FIFO is fixed at 128 bytes for receive and transmit.

Note the following characteristics of the Commtech Fastcom: 422/2-PCI-335 and Fastcom: 422/4-PCI-335 adapter boards (<http://www.commtech-fastcom.com>):

- The Fastcom 422/2-PCI-335 board has two independent RS-422/485 channels, the Fastcom 422/4-PCI-335 board has four independent RS-422/485 channels.
- The Fastcom 422/2-PCI-335 and Fastcom 422/4-PCI-335 board can handle baud rates up to 6.25 megabits/second.

---

**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 additional notes for specific boards.

---

## 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 computer, Quatech QSC-100 or ESC-100, or Diamond Systems Emerald-MM or Emerald-MM-8 serial device connected to the target computer, 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 computer 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 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 three sections, Composite, Obsolete, and Obsolete V1.0.

---

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 **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. Note the following:
  - Pay particular attention to the **Parameter** group Board Setup entry.
  - When you select the Receive Setup entry, for each channel, set the value of the **Receive Sample Time** parameter to a sample time value faster than the data being sent. Do not leave this value at -1. You must set this for all channels, including channels that you are not using;

otherwise, you will receive an error when generating code for the target application.

**11** Add a Pulse Generator and Target Scope block.

**12** Configure the Pulse Generator block so that its **Pulse type** is **Sample based**.

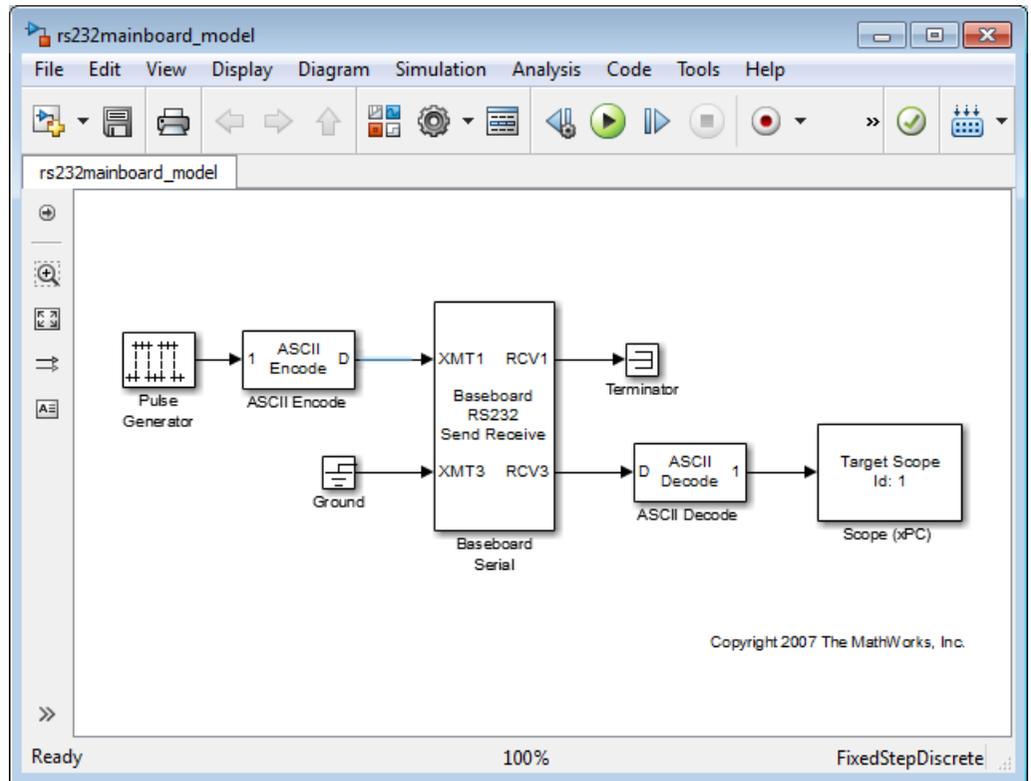
The dialog changes to display a **Sample time** parameter. Enter a **Sample time** that is slower than the one you set for **Receive Setup**.

**13** 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.

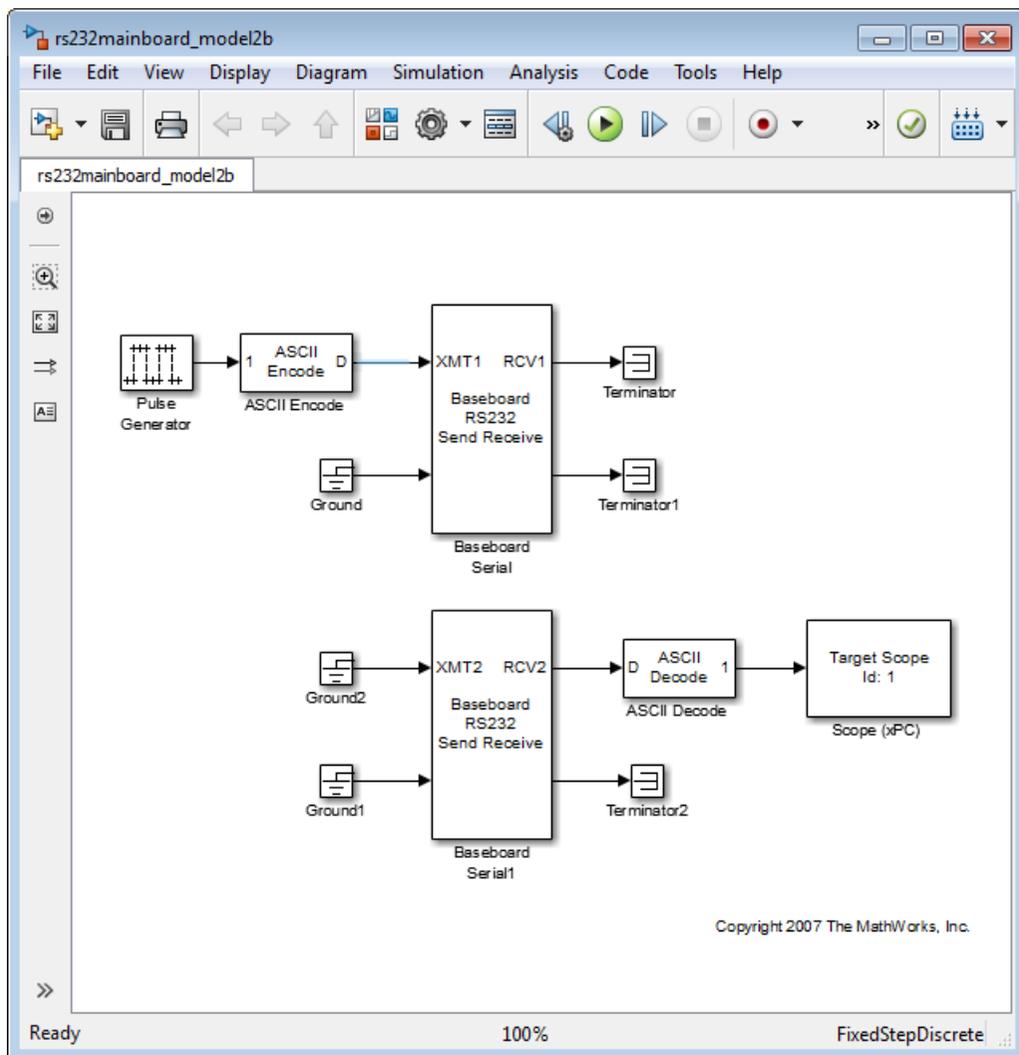
**14** From the Simulink Library Browser, select **Sources**. Depending on your configuration, drag and drop the Ground block. Connect this block to the unused XMT3 port to suppress unused port messages.

Your model might use one block or two.

The single-block model uses the Com1 / Com3 port pair:



The two-block model uses two sets of Custom port pairs:



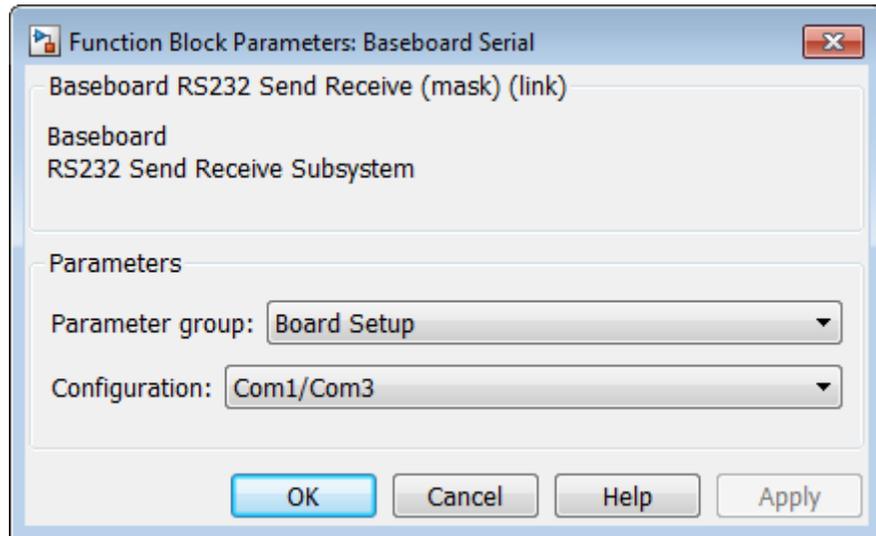
- 15 Double-click a Baseboard RS232 Send Receive block. Enter values to configure the port(s) on the target computer for this board.

---

**Note** This is a dynamic dialog box that changes depending on the **Parameter group** selection.

---

For example, if the **Parameter group** is Board Setup and the target computer port is connected to COM1/COM3, your Send Receive block dialog box looks like this:



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

**16** 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)

The xPC Target software and Simulink Coder™ create C code from your Simulink model. You can then use a C compiler to create executable code that runs on the target computer. This topic assumes that you know how to

configure your model to create an xPC Target application. (See “Build and Download Target Application”.)

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

---

**Note** You cannot use a serial port to communicate between the host computer and target computer 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 **Code** menu, click **C/C++ Code > Build Model**.

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

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

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

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

This section includes the following topics:

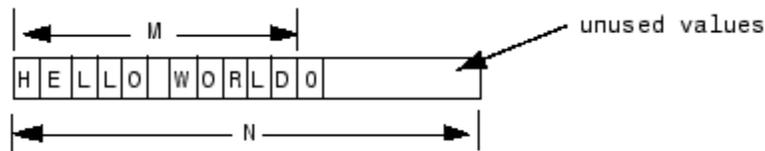
- “Signal Data Types” on page 4-15 — Describes signal data types that composite drivers support.
- ASCII Encode/Decode (Composite) — (Generic) Describes encoder and decoder blocks. Encoders convert input signals for the send/receive subsystem to ASCII strings. ASCII decoders parse the string from the Send/Receive subsystem.
- FIFO Read/Write (Composite) — (Generic) Describes FIFO read and write blocks.
- RS232 State (Composite) — (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 (Composite) — Provides blocks for sending and receiving.
- Modem Control (Composite) — Controls the state of either or both of the RTS and DTR output lines.
- Modem Status (Composite) — Reads the states of the four input modem control lines.

## Signal Data Types

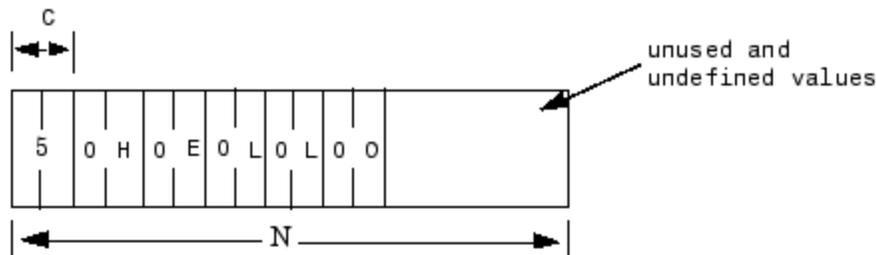
Signals between blocks in composite drivers can be one of several basic data types, 8-bit and 16- or 32-bit. All of these types are structures.

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 figure illustrates 8-bit int NULL-terminated and 8-bit uint NULL-terminated data types.



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. This figure illustrates count + 16-bit int and count + 16-bit uint data types. It also applies to count + 32-bit int and count + 32-bit uint data types



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.

### Handling Zero Length Messages

As a general rule, configure a FIFO read block of your model serial I/O to execute faster than the model receives data. Doing so prevents the receive FIFO buffer from overflowing. This implies that you must configure your model to deal with the possibility that there is no message on a FIFO read block output.

Receive FIFOs might not always have enough characters to satisfy a FIFO read operation. Any model that receives serial I/O might have a FIFO read block that will execute in this situation. This will cause a FIFO read block to perform one of the following, depending on how you configure the behavior:

- Return the last message it received
- Return a zero length message

The xPC Target library of composite serial drivers has three FIFO read blocks, FIFO Read HDRs, FIFO Read Binary, and FIFO Read (described in FIFO Read/Write). For the FIFO Read HDRs or FIFO Read Binary blocks, you configure this behavior with the **Output behavior** parameter. The FIFO Read block always returns either a new message or a zero length message.

If you need to execute model code only if a new message arrives, check the first element of the returned vector, depending on the string data type, as described in the following. If the message has nonzero length, enable a subsystem to process the new string; otherwise, do not process it.

- In the 8-bit data type, the returned string is NULL-terminated. Therefore, if the first element is 0, the string has zero length and the FIFO read did not detect a new message.
- In the 16- and 32-bit data types, the first element is the number of characters in the string. This value is 0 if the FIFO read did not detect a new message.

### **Controlling When You Send a Message**

You can use the structure of both serial data types (“Signal Data Types” on page 4-15) to control when a message is sent. In both cases, a 0 in the first position indicates an empty string.

- 8-bit data types — A value of 0 value in the first position is the NULL terminator for the string.
- 16- and 32-bit data types — The first position is the number of characters that follow.

If you connect an empty string to the XMT port on one of the send/receive subsystems, no characters are pushed onto the transmit FIFO. You can get this empty string using one of the following:

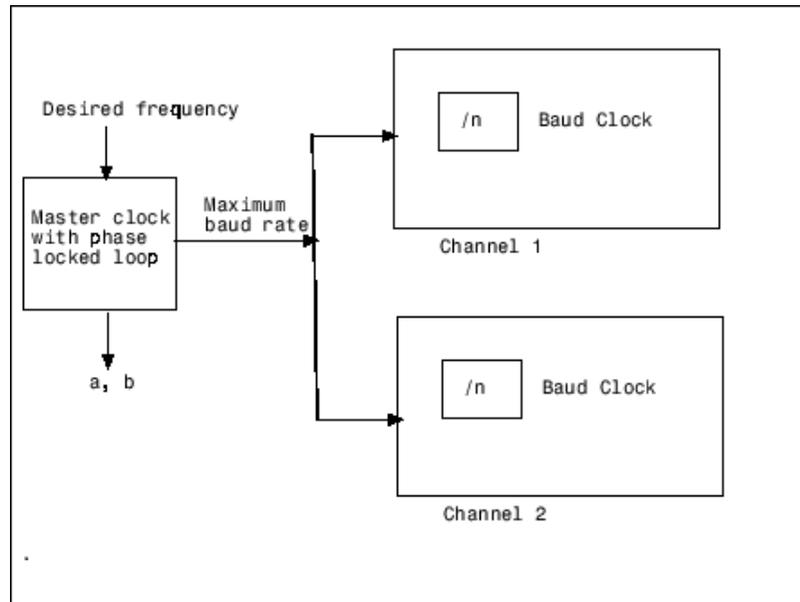
- If you have a single string that you want to send occasionally, use the Product block to multiply the entire string by either 0 or 1. In this case, the 0 or 1 value becomes a transmit enable. To optionally optimize this operation, you can use a Demux block to extract the first element and multiply just that element by 0 or 1, then use the Mux block to combine it again.
- Use a Manual Switch, Multiport Switch, or Switch block (configured for two ports to choose between different messages with one of the choices being a vector of all 0 values). The Switch block only selects between vectors that all have the same width. However, because the string length does not need to use the whole vector, you just need to fill them to the same width with 0 values following your data.

### **Defining the Commtech Fastcom Baud Rate for Commtech Fastcom 422/2-PCI Boards**

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. Note, this section applies to only Commtech Fastcom 422/2-PCI boards, not Commtech 422/2-PCI-335 and 422/4-PCI-335 boards.

- **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 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:

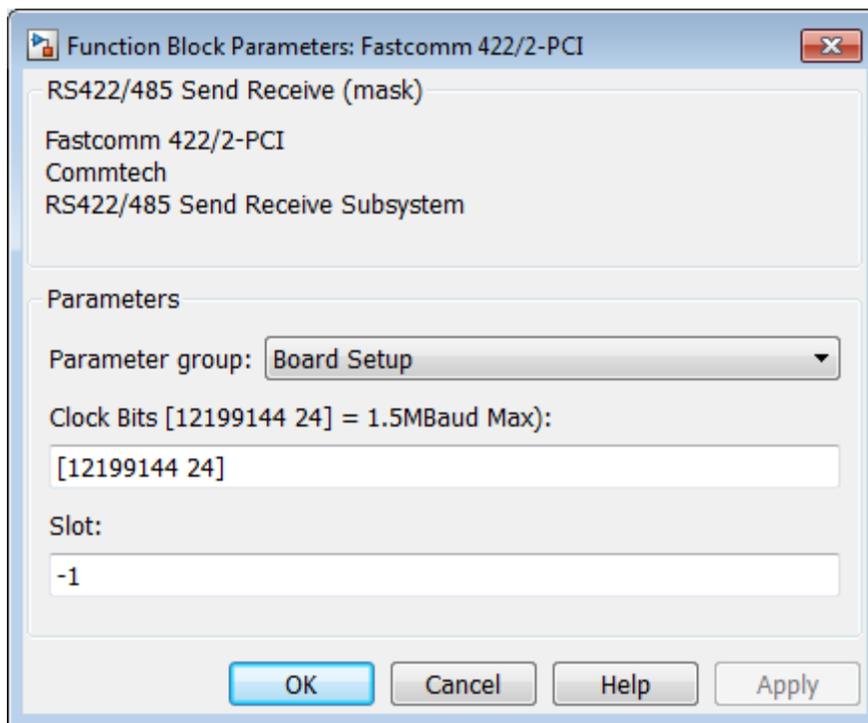
- In the MATLAB Command Window, type a command like the following. With a desired frequency (for example,  $1.5e6$ ) as the desired input, the `fc422mexcalcbits` utility calculates parameter values that you can use to configure the board rate for your board.

```
[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 does not meet your requirements, 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 Fastcomm 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.

---

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

# ASCII Encode/Decode (Composite)

<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 is {'double'}. This parameter specifies the Simulink data types allowed for the input ports. A cell vector with the same number</p>

# ASCII Encode/Decode (Composite)

---

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 `cdmabcdeffgh90`, you must specify the value of the **Number of variables** parameter as 2.

The first variable is assigned the value `0xcd`. Next, the string `mabcdeffgh` 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 `mabcdeffgh` 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.

# ASCII Decode V2

---

**Purpose** ASCII Decode V2

**Library** xPC Target Library for RS232

**Description** 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.

## Block Parameters

### Format

Enter a `scanf` like format string. Each format specifier such as `%d` must match a corresponding part of the input vector. Literal strings in the format must match the first character plus the number of characters. Acceptable format specifiers are the following, enclosed in single quotes. Failure to include these quotes causes simulation failures: `%c`, `%d`, `%i`, `%o`, `%u`, `%x`, `%e`, `%f`, and `%g`. These follow the normal description for `scanf`. The number of format specifiers actually satisfied by the input string is known as the count output.

An example format string is:

```
'foo %d bar %f\n'
```

In this example, assume that the data from the FIFO read is `'foo 5'`. In this case, the count output is 1 and the second output will be unchanged from the last time both were found in a string. If the block expects two values, and the returned count output value is less than two, there is an error in the data.

**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 (Composite)

---

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: {'volts', 'amps'}. Also configure the **Terminating string** parameter for carriage return (13) and line feed (10): [13 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 `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 you have a zero length vector, you might want to have your application perform a particular operation, or ignore the case.) If the delimiter is found, the block returns elements up to and including the delimiter in the output vector. Selecting this check box causes 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 indicates the number of characters being returned using the extra element as:

- 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. For example, if you enter a value 10, but on execution the FIFO contains 11 characters plus the null terminator, the block will not return any characters. On the other hand, if it contains 5, the block returns 5 characters plus the null terminator.

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

# FIFO Read/Write (Composite)

---

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,

The value from the maximum input port is the maximum number of characters to be removed from the FIFO. 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.

# FIFO Read/Write (Composite)

---

## 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.

**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.

---

**Note** If the same header has arrived in the FIFO more than once since the block was last executed, the block will discard the older data and only return the latest instance of the header. In this way, the block catches up with data that arrives faster than the block executes.

---

The `xpcdemos` directory contains the following examples 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 the characters defining the end of string, 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 the FIFO has 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.

## FIFO Read HDRS (Composite)

---

### **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 (Composite)

---

**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.

---

**Note** If the same header has arrived in the FIFO more than once since the block was last executed, the block will discard the older data and only return the latest instance of the header. In this way, the block catches up with data that arrives faster than the block executes.

---

The `xpcdemos` directory contains the following examples 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 the FIFO has 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.

# FIFO Read Binary (Composite)

---

## **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.

<b>Purpose</b>	Modem Control block
<b>Library</b>	xPC Target Library for RS232
<b>Description</b>	<p>The Modem Control block controls the state of either or both of the RTS and DTR output lines on the specified port.</p> <p>This block requires an input of type <code>double</code>. 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.</p> <p><b>Port</b> (Quatech, Commtech) From the list, choose a port. The <b>Port</b> parameter defines the port to configure for this driver block.</p> <p><b>RTS</b> Select this check box to control the RTS line for this board.</p> <p><b>DTS</b> Select this check box to control the DTR line for this port.</p> <p><b>Port</b> (Diamond) From the list, choose a port. The <b>Port</b> parameter defines the port to configure for this driver block.</p> <p><b>First port address</b> (Diamond) For Emerald-MM, this value should be the same as the <b>First port address</b> parameter value you select in the Parameter Group: Board Setup dialog of the Send/Receive block. See the Diamond product documentation for information on jumper settings.</p> <p>For Emerald-MM-8, this parameter contains a value based on the <b>Base address</b> value of the configuration register in the Parameter Group: Board Setup dialog of the Send/Receive block. See the Diamond product documentation.</p>

# Modem Control (Composite)

---

## **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 computer, enter

-1

to automatically locate the board.

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

```
getxpcpci
```

<b>Purpose</b>	Modem Status block
<b>Library</b>	xPC Target Library for RS232
<b>Description</b>	<p>The Modem Status block reads the state of the four input modem control lines.</p> <p>This block has an output of type <code>Boolean</code>. If the input voltage is positive, the output is true. If the input voltage is negative, the output is false.</p> <p><b>Port</b> (Quatech, Commtech) From the list, choose a port. The <b>Port</b> parameter defines the port to configure for this driver block.</p> <p><b>CTS</b> Select this check box to monitor the CTS line.</p> <p><b>DSR</b> Select this check box to monitor the DSR line.</p> <p><b>RI</b> Select this check box to monitor the RI line.</p> <p><b>DCD</b> Select this check box to monitor the DCD line.</p> <p><b>Sample Time</b> Base sample time or a multiple of the base sample time.</p> <p><b>Port</b> (Diamond) From the list, choose a port. The <b>Port</b> parameter defines the port to configure for this driver block.</p> <p><b>First port address</b> (Diamond) This parameter specifies the first port whose input modem control line states you want to read. Do not change this value.</p> <p>For Emerald-MM, this value should be the same as the <b>First port address</b> parameter value you select in the Parameter Group: Board Setup dialog.</p>

# Modem Status (Composite)

---

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 computer, enter

-1

to automatically locate the board.

If two or more boards of this type are in the target computer, enter the bus number and the PCI slot number of 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/RS-422/RS-485 Send/Receive (Composite)

---

**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:

- The non-F blocks have RCV outputs. These blocks have basic FIFO read blocks inside the subsystem. They 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.
  - 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 (Composite) and FIFO Read Binary (Composite) 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 main board, 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 computer/target computer

# RS-232/RS-422/RS-485 Send/Receive (Composite)

---

communication. If COM1 is the host computer/target computer 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.

The **Board Setup** and **Basic Setup** configuration parameter options for F-Send-Receive blocks are the same as for 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 merge the common elements and call out the differences among the different blocks.

## Block Parameters

### Parameter Group

This parameter allows you to choose which subset of configuration parameters you want to modify. Possible values are:

- Board Setup
- Basic Setup
- Transmit Setup
- Receive Setup
- FIFO Setup

The configuration parameter subsets are:

### Board Setup

The **Parameter Group** Board Setup option is supported by both F and non-F Send Receive blocks.

### 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.

# RS-232/RS-422/RS-485 Send/Receive (Composite)

---

## **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 computer. From that display, find the instance of the board controlled by this block. Each board uses a unique interrupt request line number.

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 computer 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.

## **Master Clock Frequency** (Commtech Fastcom 422/2-PCI-335, 422/4-PCI-335)

Enter the clock generator frequency for the master clock. Enter a value between 64 MHz and 50 MHz, inclusive.

## **Use standard reference and rates** (Commtech Fastcom 422/2-PCI-335, 422/4-PCI-335)

This check box enables you to use default values for the block. Select this check box to use the default values for the **Master Clock Frequency** parameter. If you clear this check box, you can enter a custom value for the **Master Clock Frequency** parameter. This same check box also appears on the **Basic Setup** option page for all ports. Selecting this box applies to all channels.

# RS-232/RS-422/RS-485 Send/Receive (Composite)

---

## Slot (PCI boards)

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

-1

to automatically locate the board.

If two or more boards of this type are in the target computer, enter the bus number and the PCI slot number of 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 the xPC Target software 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 4-7. 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.

# RS-232/RS-422/RS-485 Send/Receive (Composite)

---

## 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 4-7.

## Basic Setup

The **Parameter Group Basic Setup** option is supported by both F and non-F Send Receive blocks.

## 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.

## Use standard reference and rates (Commtech Fastcom 422/2-PCI-335, 422/4-PCI-335)

This check box enables you to use default values for the block. Select this check box to use the default values for the **Baud divisor** and **Sampling rate** parameters. Selecting this check box allows you to enter a value for the **Baud rate** parameter and also selects the **Prescale** parameter. If you clear this check box, you can enter custom values for the **Baud divisor** and **Sampling rate** parameters, but must use the default value for the **Baud rate** parameter. Clearing this check box also clears the **Prescale** check box.

If selected:

- Reference is set to 14.7456 MHz.

# RS-232/RS-422/RS-485 Send/Receive (Composite)

---

- **Prescale** is disabled.
- Sampling rate is 16X (standard).
- Select the baud rate from **Baud Rate**.

If not selected, you must provide a custom baud rate that is not on the drop-down list to get the desired reference. For example, if you have a desired baud rate of 1.0 MHz, you can use the following equation to calculate the reference:

$$\text{reference} = \text{baud rate} \times \text{sampling rate} \times \text{divisor} \times \text{prescale}$$

The following values give an acceptable reference that is less than or equal to 50 MHz.

$$\text{reference} = 1.10\text{e}6 \times 16 \times 1 \times 1 = 16 \text{ MHz}$$

## **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, (Commtech Fastcom 422/2-PCI-335, 422/4-PCI-335)

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.

## **Sampling rate** (Commtech Fastcom 422/2-PCI-335, 422/4-PCI-335)

From the list, select 8x or 16x (standard) per port. This parameter is enabled if you clear the **Use standard reference and rates** parameter.

## **Prescale** (Commtech Fastcom 422/2-PCI-335, 422/4-PCI-335)

For very slow baud rates, you might need to enable this divide by 4 prescaler to achieve the desired rate. In most cases, you do not need to enable this.

# RS-232/RS-422/RS-485 Send/Receive (Composite)

---

## 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.

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

# RS-232/RS-422/RS-485 Send/Receive (Composite)

---

value of **Receive FIFO interrupt level**. If the hardware FIFO contains at least one character, an interrupt is signaled.

(Commtech) The Fastcom 422/2-PCI and Fastcom 422/2-PCI F blocks do not have this parameter as a drop-down list. Instead, these blocks allow you to enter any number of bytes up to the hardware FIFO size, with 64 as the default. Typically, 8 to 16 bytes is large enough, but the size you choose depends on the speed of the incoming characters. Be careful that you do not enter a size that causes too many interrupts.

## **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 is used 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.

## **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 driver asserts the selected line upon transmission.

## **Hardware handshake** (Commtech Fastcom 422/2-PCI-335, 422/4-PCI-335)

Enable the hardware-based handshake for flow control. This feature of the UART is used to prevent loss due to hardware FIFO overflow. From the list, select RTS/CTS to enable this behavior; otherwise, select none.

## **Software handshake** (Commtech Fastcom 422/2-PCI-335, 422/4-PCI-335)

Allows the UART to send the XOFF character if the receive hardware FIFO gets too full. It then sends the XON character when the receive FIFO empties. Because the UART handles this,

# RS-232/RS-422/RS-485 Send/Receive (Composite)

---

the XON/XOFF characters are sent immediately, even if the transmitter is empty. From the list, select XON/XOFF to enable this behavior; otherwise, select none.

**XON character** (Commtech Fastcom 422/2-PCI-335, 422/4-PCI-335)  
Normally, enter 17 (Control Q ). This is the default.

**XOFF character** (Commtech Fastcom 422/2-PCI-335, 422/4-PCI-335)  
Normally, enter 19 (Control S ). This is the default.

**RS485 auto turnaround** (Commtech Fastcom 422/2-PCI-335, 422/4-PCI-335)

Select this check box to enable RS-485 automatic turnaround. In RS-485 mode, where the transmitters and receivers all use the same differential pair of wires, you can enable only one transmitter at a time. Select this check box to turn on the automatic transmitter control in the UART. The RTS output is routed to the enable input of the RS-485 transmitter.

**RS485 turnaround delay** (Commtech Fastcom 422/2-PCI-335, 422/4-PCI-335)

Enter the desired RS-485 turnaround delay. The UART automatically asserts RTS when there are characters in the transmit FIFO. It deasserts RTS when the transmit FIFO empties and the RS-485 turnaround default delay of 0 to 15 bit times elapses. The transmitter output goes tristate (high impedance) when disabled, which allows another device to transmit.

## Transmit Setup

The **Parameter Group** Transmit Setup option is supported by only non-F Send Receive blocks.

### 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.

# RS-232/RS-422/RS-485 Send/Receive (Composite)

---

**Use standard reference and rates** (Commtech Fastcom 422/2-PCI-335, 422/4-PCI-335)

This check box enables you to use default values for the block.

## **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 Setup**

The **Parameter Group Receive Setup** option is supported by only non-F Send Receive blocks.

## **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.

**Use standard reference and rates** (Commtech Fastcom 422/2-PCI-335, 422/4-PCI-335)

This check box enables you to use default values for the block. Select this check box to use the default values for the **Baud divisor** and **Sampling rate** parameters. Selecting this check box allows you to enter a value for the **Baud rate** parameter and also

# RS-232/RS-422/RS-485 Send/Receive (Composite)

---

selects the **Prescale** parameter. If you clear this check box, you can enter custom values for the **Baud divisor** and **Sampling rate** parameters, but must use the default value for the **Baud rate** parameter. Clearing this check box also clears the **Prescale** check box.

If selected:

- Reference is set to 14.7456 MHz.
- **Prescale** is disabled.
- Sampling rate is 16X (standard).
- Select the baud rate from **Baud Rate**.

If not selected, you must provide a custom baud rate that is not on the drop-down list to get the desired reference. For example, if you have a desired baud rate of 1.0 MHz, you can use the following equation to calculate the reference:

$$\text{reference} = \text{baud rate} \times \text{sampling rate} \times \text{divisor} \times \text{prescale}$$

The following values give an acceptable reference that is less than or equal to 50 MHz.

$$\text{reference} = 1.10\text{e}6 \times 16 \times 1 \times 1 = 16 \text{ MHz}$$

## 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.

# RS-232/RS-422/RS-485 Send/Receive (Composite)

---

## 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.

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 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.

# RS-232/RS-422/RS-485 Send/Receive (Composite)

---

## **FIFO Setup**

The **Parameter Group** FIFO Setup option is supported by only 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.

# RS232 State (Composite)

---

**Purpose** RS232 State block

**Library** xPC Target Library for RS232

**Description** 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. 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. 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. 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.

## Block Parameters

### Overrun error output

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.

### Parity error output

Select this check box to retrieve parity error output. This output is true if any byte in the input vector fails the parity check.

### Framing error output

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.

### Break interrupt output

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

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.

---

# RS232 State (Composite)

---

# Serial Communications Obsolete Drivers Support

---

- “Obsolete RS-232 Drivers” on page 5-2
- “Obsolete Boards and Blocks — Alphabetical List” on page 5-35

## Obsolete RS-232 Drivers

In this section...
“Introduction” on page 5-2
“xPC Target RS-232 Drivers (Obsolete)” on page 5-2

### Introduction

The xPC Target software interfaces the target PC to serial devices using either the COM1 or COM2 port of the main board.

This topic describes the obsolete xPC Target blocks that communicate with the COM1 or COM2 port of your main board. The obsolete drivers support RS-232 I/O only for the target computer serial ports. These drivers support synchronous, asynchronous, and binary (asynchronous) communication mode. The xPC Target software 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. These are older blocks for serial communication. Consider using the composite serial communication blocks described in “RS-232/422/485 Drivers (Composite)” on page 4-6 instead.

### xPC Target RS-232 Drivers (Obsolete)

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

- “Simulink Blocks for RS-232 I/O (Obsolete)” on page 5-3 — Add setup, send, send/receive, and receive blocks to your Simulink model.
- “MATLAB Message Structures for RS-232 I/O (Obsolete)” on page 5-3 — Create message structures to sequence instructions to and from the RS-232 device.
- “RS-232 Synchronous Mode (Obsolete)” on page 5-4 — Add synchronous driver blocks to have the device wait for a response before continuing with other computations.

- “RS-232 Asynchronous Mode (Obsolete)” on page 5-15 — 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 (Obsolete)” on page 5-28 — Description of the RS-232 blocks for the obsolete drivers.
- “RS-232 MATLAB Structure Reference (Obsolete)” on page 5-28 — Description of the RS-232 MATLAB structure for messages.
- “RS-232 Binary Mode (Obsolete)” on page 5-32 — Add binary driver blocks to transfer raw data.

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

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 required 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.

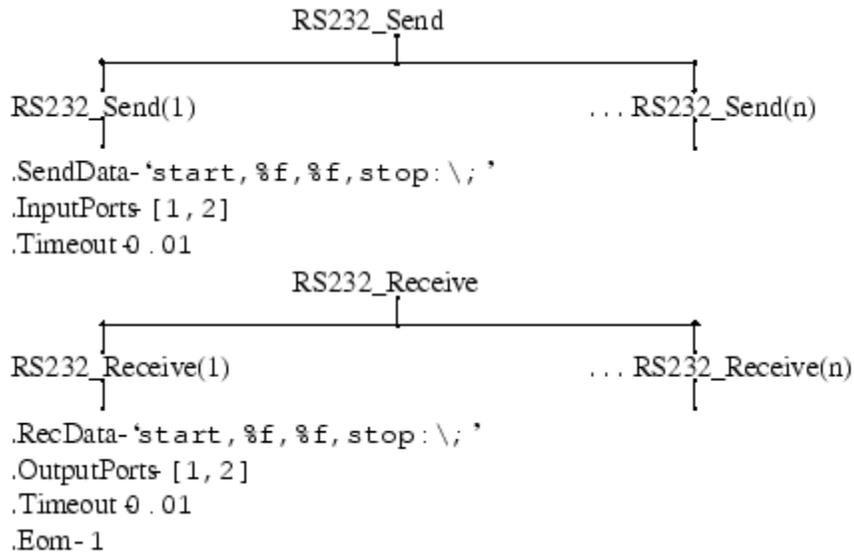
### **MATLAB Message Structures for RS-232 I/O (Obsolete)**

Communication is through a series of messages passed back and forth between the target computer 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 computer must know how to interpret the data returned from the RS-232 device.

The xPC Target software 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 5-24.

### RS-232 Synchronous Mode (Obsolete)

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 timeout. This section includes the following topics:

- “Notes for RS-232 Synchronous Mode” on page 5-5 — Overview of RS-232 communication with xPC Target blocks.
- “Adding RS-232 Driver Blocks (Synchronous)” on page 5-5 — 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 5-11 — 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 computer sends a message string to the device.

In synchronous mode, the data is sent to the external device. Before continuing with other computations, the block waits (or *blocks*) until a response is received from the device or the block reaches a timeout. This implies that you have placed the Send & Receive block 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 computer 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 Obsolete drivers. Refer to the Obsolete 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 Double-click Obsolete.

The RS232 Obsolete driver sublibrary opens.

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

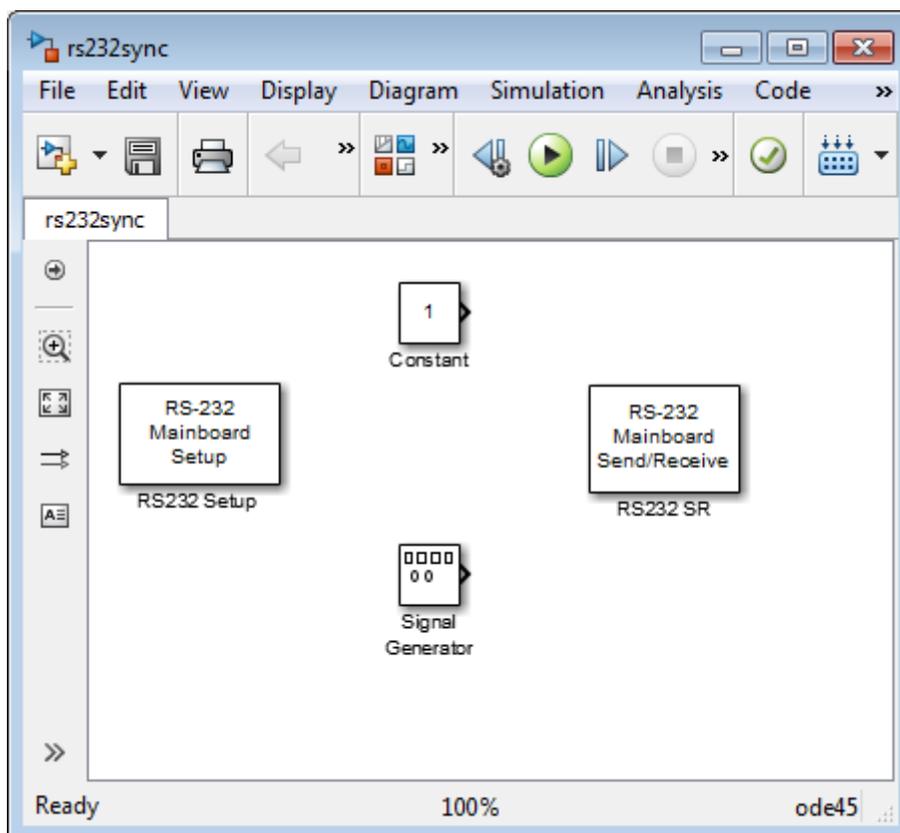
---

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

---

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

Your model looks like this:




---

**Note** Inputs on the RS-232 Send/Receive block are not defined, and are not visible. The inputs are defined in a MATLAB message structure and become visible only after you load that structure into the MATLAB workspace and update your Simulink model.

---

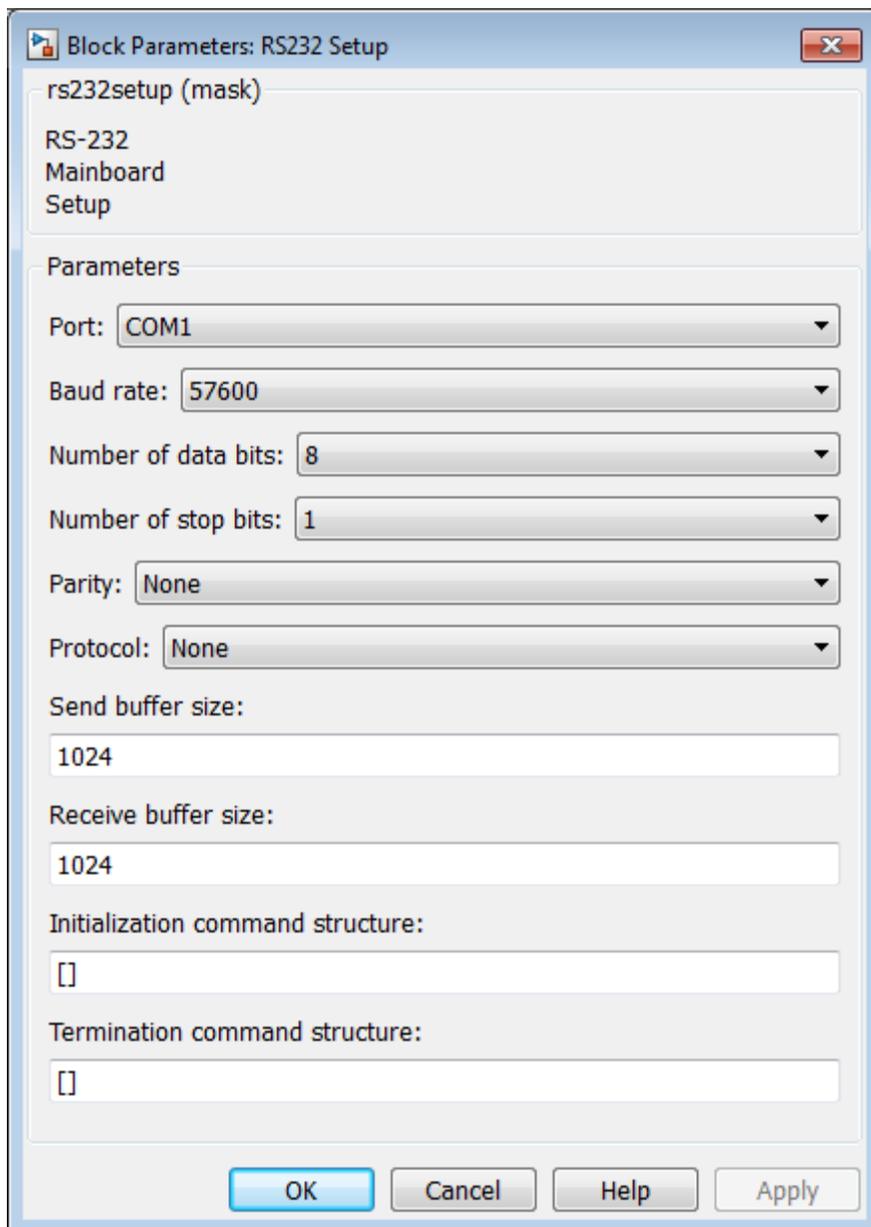
- 8 Double-click the RS-232 Setup block. Enter values to configure the COM1 port on the target computer.

---

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

---

For example, if the target computer is connected to COM1, and serial communication is set to 57600 baud, 8 data bits, and 1 stop bit, your Block Parameter dialog box looks like this:

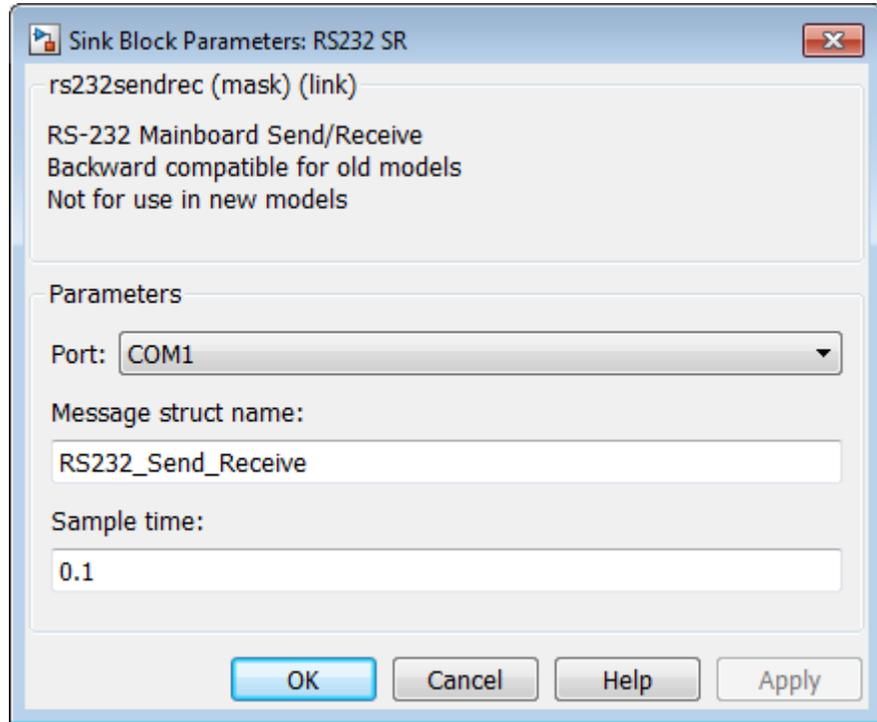


For more information on entering the block parameters, see **RS-232 Mainboard Setup**. For the procedure to create the initialization and termination structures, see “**RS-232 MATLAB Structure Reference (Obsolete)**” on page 5-28.

- 9 Click **OK**. The Block Parameters dialog box closes.
- 10 Double-click the RS-232 Send/Receive 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. The name of the message structure is not the name of the script file, but the name of the structure created with the script.

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)*. For the procedure to create the send/receive structure, see “RS-232 MATLAB Structure Reference (Obsolete)” on page 5-28.

**12** 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 5-11.

**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 a script file and loading that script into the MATLAB workspace.

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

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 (Obsolete)” on page 5-28.

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.

```
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;
```

---

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

---

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

```
RS232sync_messages.m
```

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

```
RS232sync_messages
```

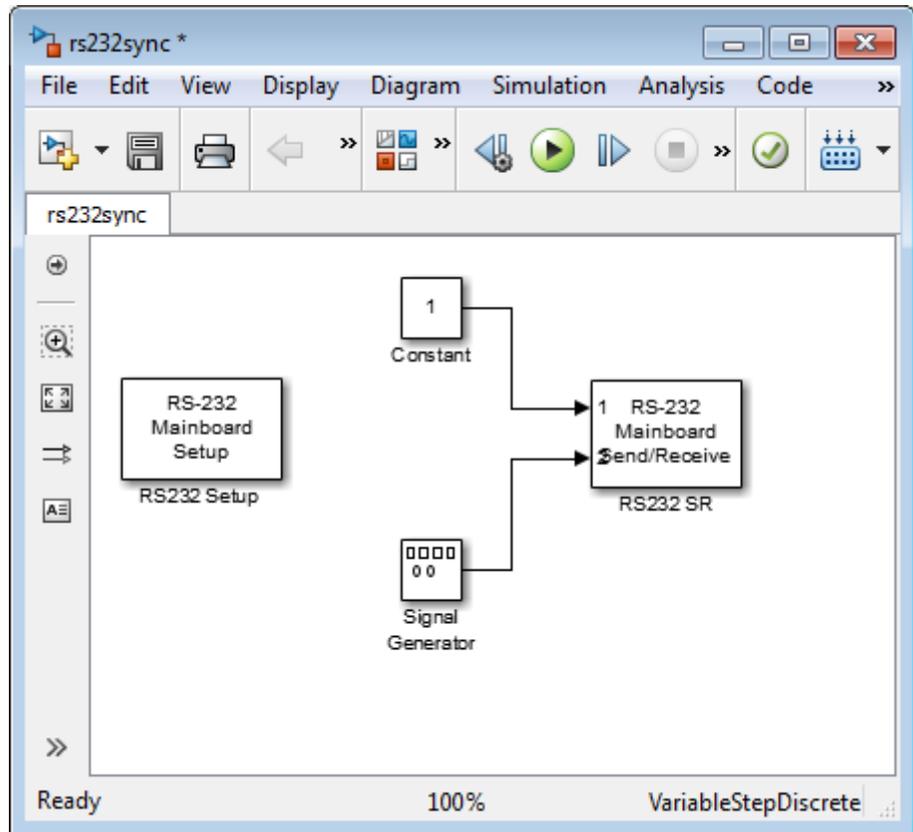
The MATLAB interface loads and runs the script to create the message structures in the MATLAB workspace used by the RS-232 driver blocks.

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

The Simulink software updates the RS-232 driver blocks with the information from the structures. For example, it 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 looks like this:



- 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 script `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 load the message structures automatically 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 on your target computer. It will *block* and wait for a reply each time the application sends a message.

### **RS-232 Asynchronous Mode (Obsolete)**

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” on page 5-15
- “Adding RS-232 Driver Blocks (Asynchronous)” on page 5-16
- “Creating RS-232 Message Structures (Asynchronous)” on page 5-24
- “Building and Running the Target Application (Asynchronous)” on page 5-27

**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 computer in a simple loop-back test. This test lets you check that the RS-232 Send and RS-232 Receive blocks work 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 computer through COM1 are received by COM2 on the same target computer.

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 computer 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 **RS232**.

- 3 Double-click Obsolete.

The RS232 Obsolete driver sublibrary opens.

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

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

---

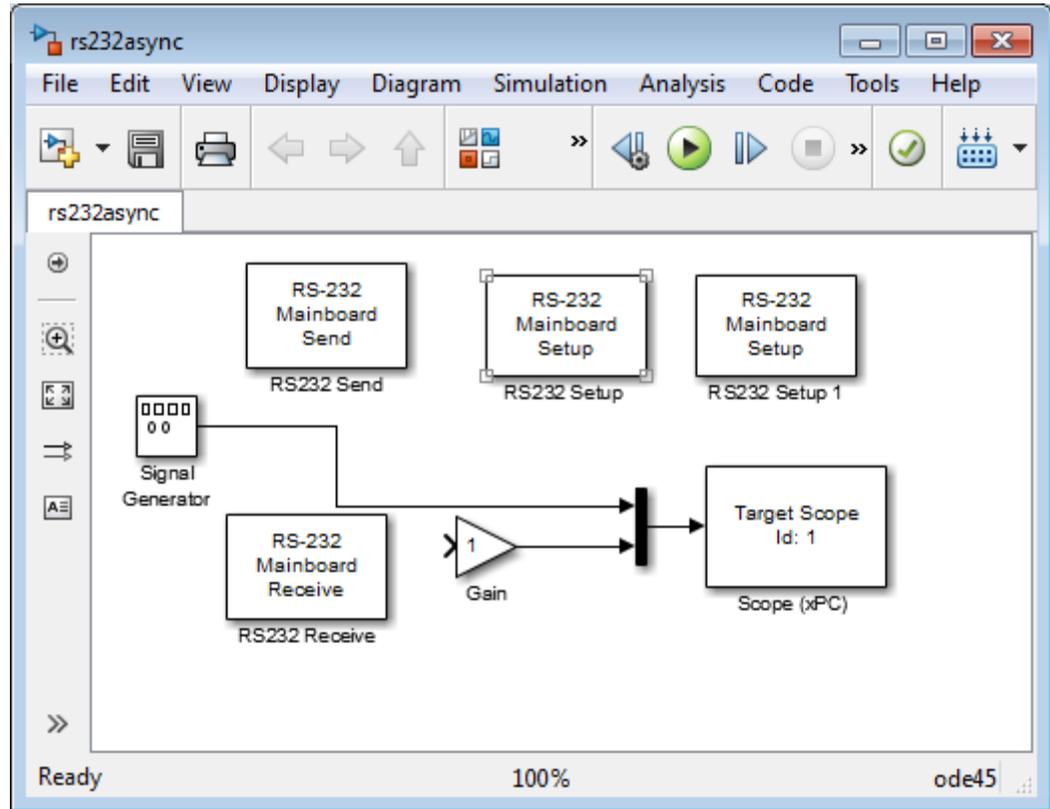
**Note** This library contains one send and one receive block.

---

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**.

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

Your model looks like this:

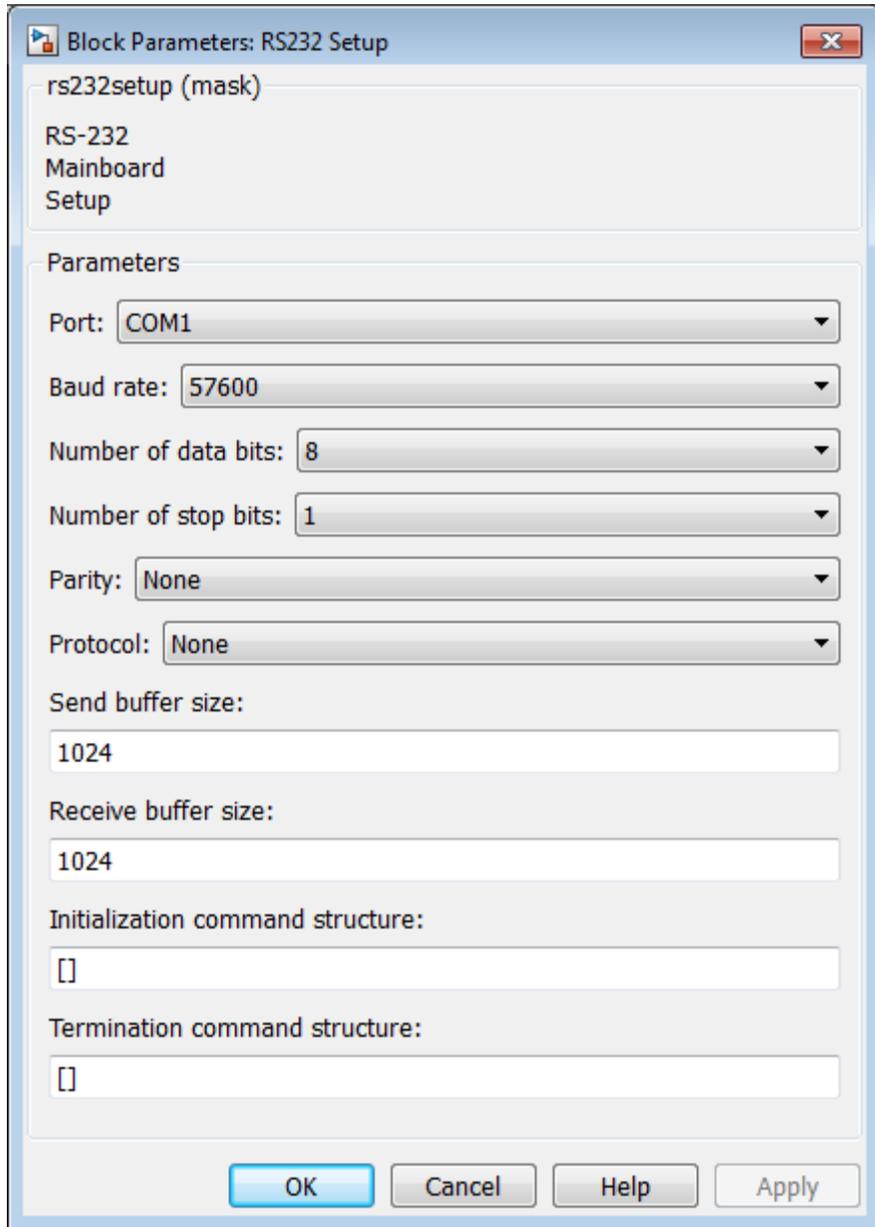


Inputs on the RS-232 Send block and the outputs on the RS-232 Receive block are not defined and are not visible. The inputs and outputs are defined in a MATLAB message structure, and become visible only after you load that structure into the MATLAB workspace and update your Simulink model.

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

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 looks like this:



---

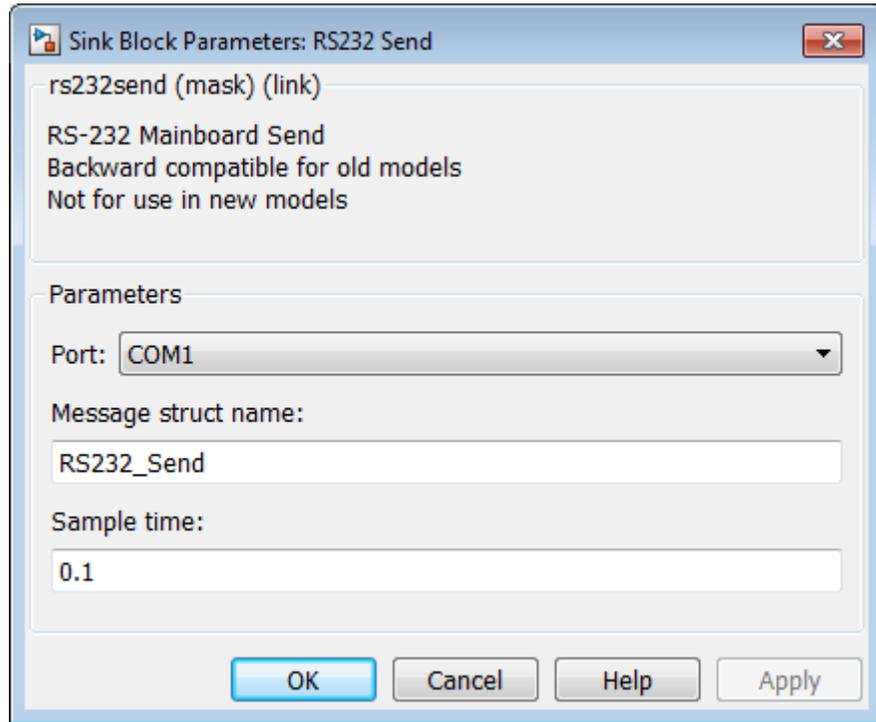
**Note** If you are not using an initialization or termination structure, in the **Initialization Struct** and **Termination Struct** boxes, enter the empty matrix [ ].

---

For more information on entering the block parameters, see **RS-232 Mainboard Setup**. For the procedure to create the initialization and termination structures, see “**RS-232 MATLAB Structure Reference (Obsolete)**” on page 5-28.

- 9** Click **OK**. The Block Parameters dialog box closes.
- 10** 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.
- 11** Double-click the Send block. The Block Parameters dialog box opens.
- 12** 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.

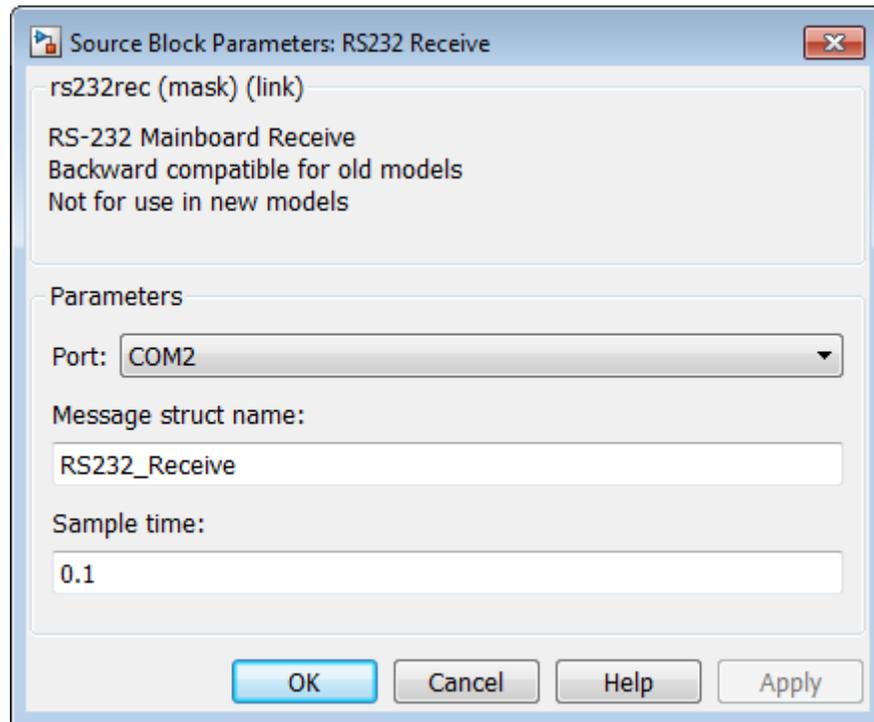
Your Block Parameters dialog box looks like this:



For information on entering the block parameters, see *RS-232 Mainboard Send (Asynchronous)*. For the procedure to create the send structure, see “RS-232 MATLAB Structure Reference (Obsolete)” on page 5-28.

- 13** Click **OK**. The Block Parameters dialog box closes.
- 14** Double-click the RS-232 Receive block.
- 15** The Block Parameters dialog box opens.
- 16** 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 looks like this:



For information on entering the block parameters, see *RS-232 Mainboard Receive (Asynchronous)*. For the procedure to create the send structure, see “*RS-232 MATLAB Structure Reference (Obsolete)*” on page 5-28.

- 17** Click **OK**. The Block Parameters dialog box closes.
- 18** 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**.
- 19** 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 5-11.

**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 a script and load that script into the MATLAB workspace. See `xpcrs232v2` 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 **Script**.

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 (Obsolete)” on page 5-28.

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`.

```
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

- Field names in the structures are case sensitive.
  - 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 script. For example, enter

```
RS232async_messages.m
```

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

```
RS232async_messages
```

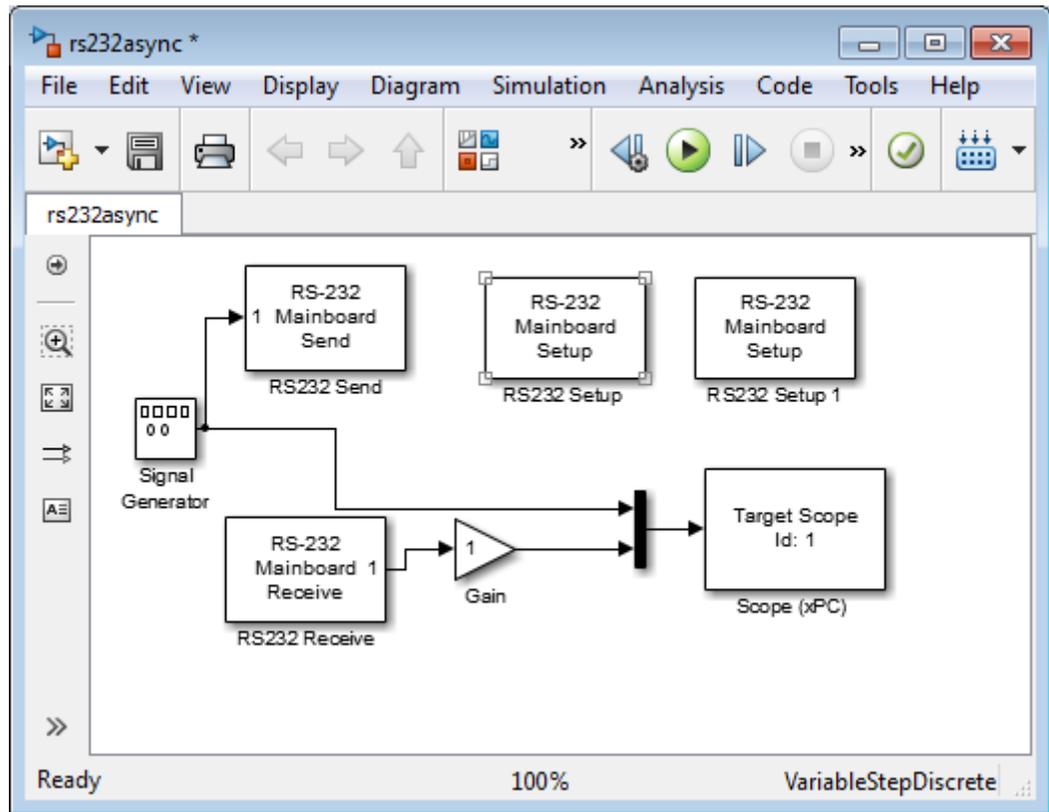
The MATLAB interface loads and runs the script to create the message structures in the MATLAB workspace used by the RS-232 driver blocks.

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

The Simulink interface updates the RS-232 driver blocks with the information from the structures. For example, it 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 looks like this:



- 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 script 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).** The xPC Target and Simulink Coder software create C code from your Simulink model. You can then use a C compiler to create executable code that runs on the target computer.

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 computer and target computer 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 **Code** menu, click **C/C++ Code > 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.

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 computer to target computer communication using a serial port because that would block that COM port and the example would not operate.

---

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

The xPC Target block library 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** — 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)** — Sequences the send and receive messages for synchronous serial communication.
- **RS-232 Mainboard Send (Asynchronous)** — Sequences the send messages.
- **RS-232 Mainboard Receive (Asynchronous)** — Sequences the receive messages.

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

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 5-29 — 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 5-30 — 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 5-31 — 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 5-31 — 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	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,  RS232_Send_Receive(1).InputPorts= [1 2];
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.

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

**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	<p>Data and format sent to the RS-232 device. Default value = <code>''</code>.</p> <p>Note that the <code>SendData</code> 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 <code>SendData</code> 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 = <code>[]</code>. 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>

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 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,  <code>RS232_Send_Receive.OutputPorts = [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.

**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
%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 (Obsolete)

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 5-33 — When to use RS-232 binary mode
- “RS-232 Binary Mode I/O” on page 5-33 — How to select drivers from the xPC Target block library
- RS-232 Binary Receive — Explanation of block parameters, inputs, and outputs
- RS-232 Binary Send — Explanation of block parameters and input
- “Example Using RS-232 Binary Mode I/O” on page 5-34 — 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 the Simulink software 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.

**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

```
xpplib
```

The xPC Target Block Library opens.

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

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

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

The **Library: xpplib/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 the xPC Target product. 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, which receives the indicated number of bytes. The data is then decoded and displayed on a target scope. 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 according to 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).

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

RS-232 Binary Pack  
RS-232 Binary Receive  
RS-232 Binary Send  
RS-232 Binary Unpack Block  
RS-232 Mainboard Receive (Asynchronous)  
RS-232 Mainboard Send (Asynchronous)  
RS-232 Mainboard Send/Receive (Synchronous)  
RS-232 Mainboard Setup

# RS-232 Binary Pack

---

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

**Purpose** RS-232 Binary Receive block (obsolete)

**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

---

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 the Simulink and Simulink Coder software 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.

<b>Purpose</b>	RS-232 Binary Send block (obsolete)
<b>Library</b>	xPC Target Library for RS232
<b>Description</b>	<p>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.</p> <p>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 <b>Maximum width per packet</b> parameter.</p>
<b>Block Parameters</b>	<p><b>COM Port</b> From the list, select COM1, COM2, COM3, or COM4. This is the port you want to use to send the data.</p> <p><b>Maximum width per packet</b> Enter the width of the incoming data. This value is a constant, unlike the Receive block.</p> <p><b>Sample time</b> Enter the frequency this data is sent.</p>

# RS-232 Binary Unpack Block

---

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

# RS-232 Mainboard Receive (Asynchronous)

---

<b>Purpose</b>	RS-232 Mainboard Receive block (Asynchronous) (obsolete)
<b>Library</b>	xPC Target Library for RS232
<b>Block Parameters</b>	<p><b>Port</b> 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> 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 5-24.</p> <p><b>Sample time</b> 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)

---

<b>Purpose</b>	RS-232 Mainboard Send block (Asynchronous) (obsolete)
<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 5-24.</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)

---

<b>Purpose</b>	RS-232 Mainboard Send/Receive Block (Synchronous) (obsolete)
<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 5-11.</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 enough 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

---

**Purpose** RS-232 Mainboard Setup block (obsolete)

**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 computer and target computer are connected using serial communication, one COM port on your target computer is dedicated for communication with your host computer. You cannot use this COM port in your block diagram as an I/O device. For example, if the target computer uses COM1 for communication with the host computer, 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 computer to target computer 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 computer 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.

**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 5-11 and “Creating RS-232 Message Structures (Asynchronous)” on page 5-24.

**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.

# RS-232 Mainboard Setup

---

# GPIB I/O Support

---

- “GPIB Drivers” on page 6-2
- “Using GPIB Drivers” on page 6-5
- “GPIB MATLAB Structure Reference” on page 6-13
- “Boards and Blocks — Alphabetical List” on page 6-20

## GPIB Drivers

### In this section...

“Introduction” on page 6-2

“Hardware Connections for GPIB” on page 6-2

“Simulink Blocks for GPIB” on page 6-3

“MATLAB Message Structures for GPIB” on page 6-3

### Introduction

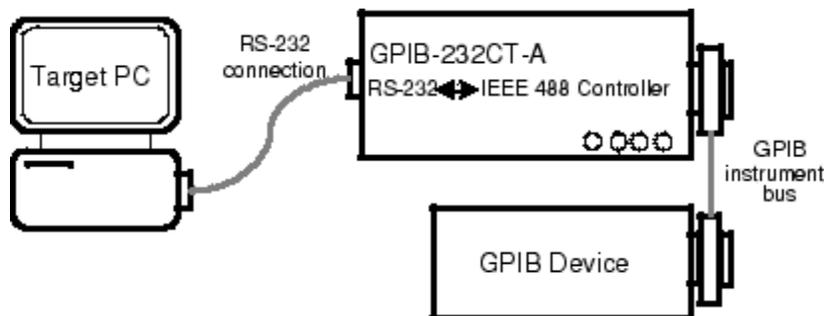
The xPC Target software interfaces the target computer to a GPIB instrument bus using an external GPIB controller from National Instruments. This external controller is connected to the target computer with a serial cable.

The xPC Target software 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.

### Hardware Connections for GPIB

The xPC Target software 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 computer 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.

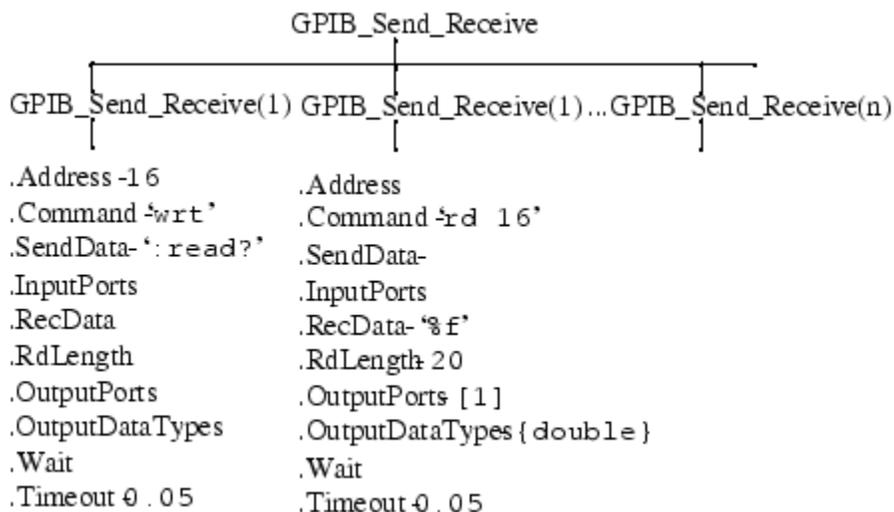
- GPIB Setup — Provide one setup block for each GPIB controller. The setup block does not have any inputs or outputs, but sends the initialization and termination messages.
- GPIB Send/Receive — The send/receive block has inputs and outputs from your Simulink model, and sequences both the send and receive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

## MATLAB Message Structures for GPIB

Communication is through a series of messages passed back and forth between the target computer 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 computer 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. The xPC Target software 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 6-10.

## Using GPIB Drivers

### In this section...

“Introduction” on page 6-5

“Adding GPIB Driver Blocks” on page 6-5

“Creating GPIB Message Structures” on page 6-10

### Introduction

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.

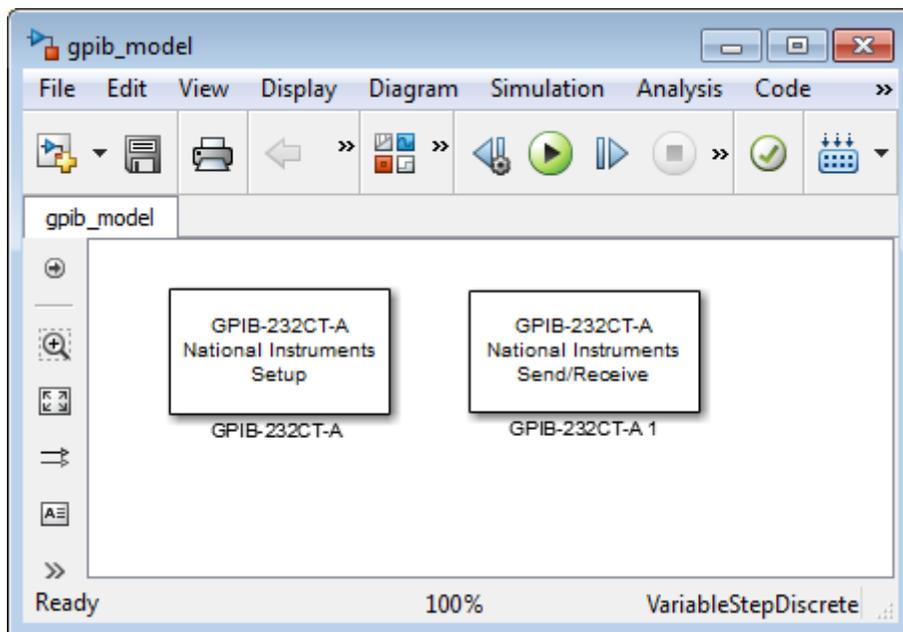
- 3 Double-click the National Instruments group block.

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 looks like this:



---

**Note** 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.

---

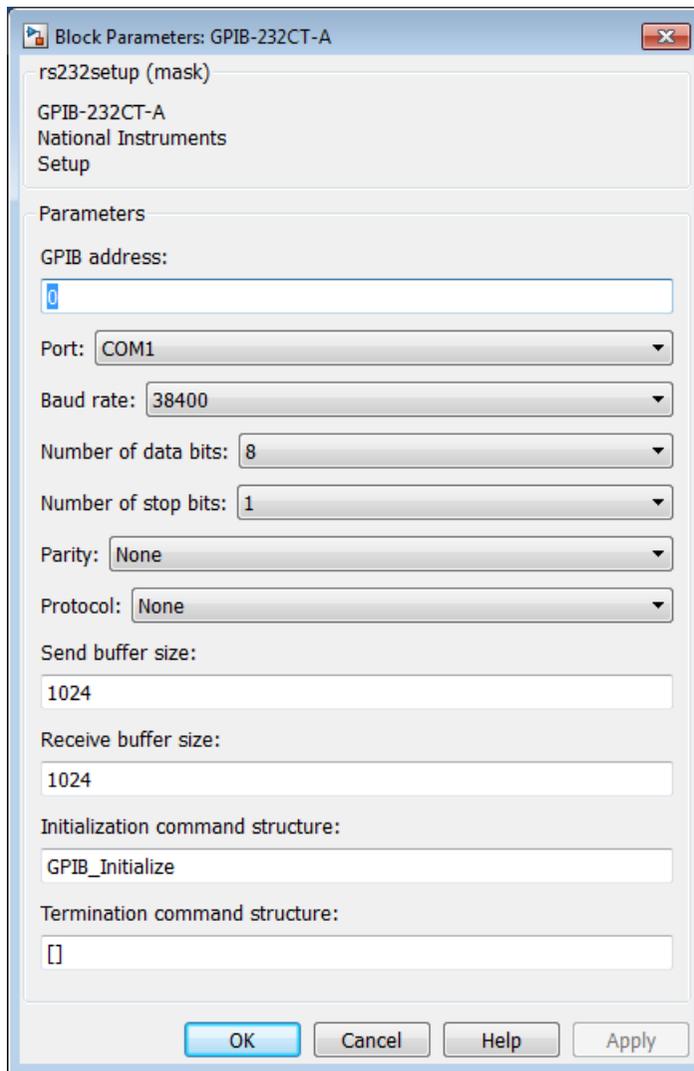
- 5 Double-click the GPIB Setup block. Enter values that correspond to the DIP switch settings you set on the GPIB-232CT-A controller. In the **Initialization Struct** box, enter the name for the MATLAB structure this block uses to send initialization messages to the GPIB device.

---

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

---

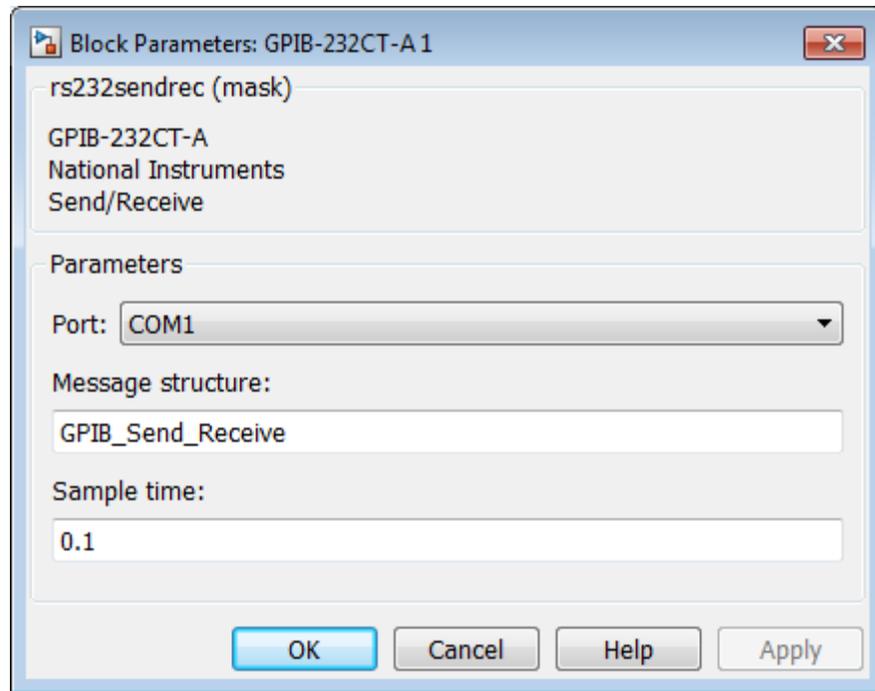
For example, if the target computer is connected to COM1, and you set the switches on the controller to 38400 baud, 8 data bits, and 1 stop bit, your Block Parameter dialog box looks like this:



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 6-10.

- 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 computer 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 **Solver** pane of the **Simulation > Model Configuration Parameters** dialog box.

Your Block Parameter dialog box looks like this:



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 6-10.

## 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 a script file and load that script into the MATLAB workspace.

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

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 6-13 and “GPIB Send/Receive Message Structure” on page 6-14.

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
```

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

```
GPIB_Messages.m
```

- 4 Close the text editing window.
- 5 In the MATLAB Command Window, type the name of the script you created with the GPIB structures. For example, type

```
GPIB_Messages
```

The MATLAB interface loads and runs the script to create the message structures in the MATLAB workspace used by the GPIB driver blocks.

- 6 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.

- 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 script `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 computer.

## GPIB MATLAB Structure Reference

In this section...
“Introduction” on page 6-13
“GPIB Initialization and Termination Message Structures” on page 6-13
“GPIB Send/Receive Message Structure” on page 6-14
“Shortcuts and Features for Messages” on page 6-17
“Supported Data Types for Message Fields” on page 6-19

### Introduction

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.

### 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 timeout 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 timeout value is exceeded, a timeout 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	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.  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

<b>Message Fields</b>	<b>Description</b>
	<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 <code>.Command</code> and <code>.SendData</code>. Default value = <code>[]</code>. 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 <code>scanf</code> statement. The read data is mapped to the output ports defined in the field <code>.OutputPorts</code>. 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 <code>read</code> command and defines the keyword <code>LENGTH</code>. Default value = 0.</p>
OutputPorts	<p>Defines the output ports from the driver block. Data received from a GPIB device with the <code>read</code> 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 <code>double</code>. Also, if the block has more output ports than output data types listed, the default type for the undefined ports is <code>double</code>.</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 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 required escape characters are added automatically.

```
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**

**Purpose** Support for GPIB-232CT-A controller

**Board** GPIB-232CT-A

**General Description** The GPIB-232CT-A is a GPIB controller external to the target computer. It is connected to the target computer 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 computer 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>Solver</b> pane of the <b>Simulation &gt; Model Configuration Parameters</b> dialog box.</p>

<b>Purpose</b>	GPIO-232CT-A Setup block
<b>Library</b>	xPC Target Library for GPIO
<b>Note</b>	The setup block parameters must be set to match the jumper settings on the GPIO-232CT-A controller.
<b>Block Parameters</b>	<p><b>GPIO address</b> Enter the identification number for the GPIO controller. When the GPIO-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 computer uses to communicate with the GPIO-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 X0n/X0ff. If your serial device does not support hardware handshaking, or your application software requires X0n/X0ff handshaking, you might need to select X0n/X0ff.</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 6-10.

## **Termination command struct**

Enter the name of the structure containing the termination information.

# CAN I/O Support

---

- “CAN Basics” on page 7-2
- “Model Execution Driven by CAN Messages” on page 7-79
- “Initialization and Termination CAN Messages” on page 7-83
- “CAN-AC2 and CANopen Devices” on page 7-86
- “CAN Data Frames” on page 7-87
- “Timeouts When Receiving CAN Messages” on page 7-88
- “Blocks — Alphabetical List” on page 7-91

## CAN Basics

In this section...
“xPC Target CAN Library” on page 7-2
“Data Types” on page 7-4
“Sending and Receiving Remote Frames” on page 7-5
“Supported Softing Boards” on page 7-5

### xPC Target CAN Library

The xPC Target block library offers support to connect a target computer to a CAN network using the CAN driver blocks provided by the xPC Target I/O CAN 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.

These drivers support CAN specifications 2.0A and 2.0B and use the dynamic object mode of the CAN-AC2 firmware to achieve good real-time performance.

The xPC Target CAN library intentionally restricts its support to Softing boards with two CAN ports. To test the PCI and PC/104 boards, see the `xpcdemos` folder for simple loopback test models. Type the following commands to open the corresponding models. Some models use `CAN_MESSAGE` data types, some use double data types. Models that use double data types are considered legacy examples.

<b>CAN-AC2-PCI Example</b>	<b>CAN-AC2-104 Example</b>	<b>Description</b>
CAN I/O - Simple Use Case — CAN_MESSAGE data type	CAN I/O - Simple Use Case — CAN_MESSAGE data type	Shows simple CAN I/O communication.
CAN I/O - Simple Use Case — Double data type	CAN I/O - Simple Use Case — Double data type	
CAN I/O - Message-Based Interrupts — CAN_MESSAGE data type	CAN I/O - Message-Based Interrupts — CAN_MESSAGE data type	Shows asynchronous message-based event support using interrupt driven CAN I/O communication.
CAN I/O - Message-Based Interrupts — Double data type	CAN I/O - Message-Based Interrupts — Double data type	

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 non-linking makes target applications smaller if no CAN communication functionality is required. If the model to be built contains CAN driver blocks, xPC Target links in the CAN library code required to run the model.

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.

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 the xPC Target product, and that is the main reason why this mode has been chosen over the other two modes, which are FIFO and static object mode. In this mode, you use a separate port for each ID that you are sending or receiving. The blocks for these boards send messages in priority order, from the lowest ID to the highest.

## Data Types

The xPC Target CAN blocks support the following message data types. If you have Softing ISA CAN boards, you can use only the double data type for messages. If you have Softing PCI and PC/104 CAN boards, you can use either CAN\_MESSAGE or double data types data types for new messages.

- **CAN\_MESSAGE** — Structure that contains an array of eight unsigned 8-bit integers that contains the data. This array also contains the ID, and the standard and extended ID range. CAN blocks can pass data of this type without having to manipulate it, enabling you to create simpler models. Use messages of this data type for new CAN application models. Consider updating existing CAN application models to use CAN\_MESSAGE data types.

**CAN\_MESSAGE** — Use CAN Pack blocks to pack individual signals into a CAN message for sending. Use CAN Unpack blocks to unpack individual signals from CAN messages,

You can construct a CAN\_MESSAGE by using the CAN Pack. You can use this block to pass raw data packed elsewhere, by packing a data pattern manually. You can also use a CANDB file and select a predefined data pattern.

- **Double** — Double that represents 8 bytes of message in a signal. CAN blocks manipulate data of this type before passing it. Do not use this data type to create new CAN application models. The maximum size of the data frame of a CAN message is 8 bytes. This size is the same as the C data type double uses on PC-compatible systems. At the same time, the double data type is the default data type for Simulink signals. Represent the CAN data frame within a Simulink model with a scalar Simulink signal. You can represent the data frame 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. Use these blocks to pack data types other than doubles into 64 bits (8 bytes or a double) as well as for the opposite operation. Simulink signals of data type double represent CAN data frames.

## Updating Existing Models to Use CAN\_MESSAGE Data Types

If you have Softing PCI and PC/104 CAN boards, you can update existing models to use CAN\_MESSAGE data types. As a rule:

- If your existing models contain CAN Bit-Packing and CAN Bit-Unpacking blocks, replace them with the CAN Pack and CAN Unpack blocks.
- Open and reconfigure the CAN Send and Receive blocks to use CAN\_MESSAGE data types.
- Remove Object Mode CAN Message and CANDBC Translator blocks. Updated models no longer require the conversions that these blocks perform.

See the CAN Blocks Transition document for more information.

## Sending and Receiving Remote Frames

Ordinarily, a subsystem sends its value over the CAN bus according to its own schedule, and another subsystem stores that value for use whenever downstream processing requires it. Under some circumstances, a subsystem explicitly requests the current value from another subsystem. This process is called sending a remote frame.

- To prepare a remote frame, use the CAN Pack block with the **Remote frame** check box selected. To request the current value, send the remote frame to the responding subsystem.
- To process a remote frame, use the CAN Unpack block with the **Output remote** check box selected to enable the Remote port. When the Remote port becomes true, send the current value to the requesting subsystem.

## Supported Softing Boards

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 16)
CAN-AC2-104	PC/104	Standard & Extended	Yes (up to 3)

For more information on the board specifications, visit <http://www.softing.com>.

- CAN-AC2

CAN board for the ISA bus offering two high-speed CAN ports. 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. Do not use this board for new projects; use the CAN-AC2-PCI instead. Softing plans no new firmware versions for this board.

- CAN-AC2-PCI

CAN board for the PCI bus offering two CAN ports. The CAN controllers 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 computer and lies usually in the range between 2 GB and 4 GB. Any new projects using a desktop PC as the target system should use this board and not the ISA board.

- CAN-AC2-104

CAN board for the PC/104 bus offering two CAN ports. The CAN controllers 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 KB address range and the memory-mapped area uses a 4 KB address range between 640 KB and 1 MB.

### **CAN-AC2 (ISA) with 82C200 Controller**

The CAN-AC2 (ISA) driver blocks support the Softing CAN-AC2 (ISA) board 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 Philips C200 block group contains the three available CAN blocks: Setup, Send, and Receive.

# Softing CAN-AC2-ISA with Philips PCA 82C200 Setup

---

<b>Purpose</b>	Softing CAN-AC2-ISA with Philips PCA 82C200 Setup block
<b>Library</b>	xPC Target Library for CAN
<b>Description</b>	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.
<b>Block Parameters</b>	<p><b>CAN 1 – baud rate</b></p> <p>Defines the most common baud rates for CAN port 1. If your model requires a special baud rate, select the value <code>User defined</code>. In this case, use <b>CAN 1 - user defined baud rate</b> to provide the four values for the timing information. The vector elements have the following meanings:</p> <p>[ Prescaler, Synchronization-Jump-Width, Time-Segment-1, Time-Segment-2 ]</p> <p>For more information about these values, see the Softing user manual for this board.</p> <p><b>CAN 2 – baud rate</b></p> <p>Defines the most common baud rates for CAN port 2. If your model requires a special baud rate, select the value <code>User defined</code>. In this case, use <b>CAN 1 - User defined baud rate</b> to provide the four values for the timing information. The vector elements have the following meanings:</p> <p>[ Prescaler, Synchronization-Jump-Width, Time-Segment-1, Time-Segment-2 ]</p> <p>For more information about these values, see the Softing user manual for this board.</p> <p><b>Initialization command structure</b></p> <p>Defines CAN messages sent during initialization and termination of the Setup block.</p>

# Softing CAN-AC2-ISA with Philips PCA 82C200 Setup

---

## **Termination**

Defines CAN messages sent during 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 (C8000 to D8000) of the 640 KB to 1 MB address range for memory-mapped devices, the valid settings when used within an xPC Target system are

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 when you use the loopback model provided to test the board and drivers.

# Softing CAN-AC2-ISA with Philips PCA 82C200 Send

---

<b>Purpose</b>	Softing CAN-AC2-ISA with Philips PCA 82C200 Send block
<b>Library</b>	xPC Target Library for CAN
<b>Description</b>	The Send driver block transmits data to a CAN network from within a block model.

---

**Tip** To prepare a remote frame, use the CAN Pack block with the **Remote frame** check box selected. To request the current value, send the remote frame to the responding subsystem.

---

## 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 2047. 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 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

# Softing CAN-AC2-ISA with Philips PCA 82C200 Send

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 selected, the block shows as many output ports as input ports. The data type of each output port is a double and the value is identical to the return argument of function `CANPC_write_object(...)`, described in the Softing user manual. Refer to the manual for more information. The function return codes are:

Code	Description
0	Function executed without detecting an error.
-1	Request overrun.
-4	Timeout firmware communication.
-99	Board not initialized.

## 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 . 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

<b>Purpose</b>	Softing CAN-AC2-ISA with Philips PCA 82C200 Receive block
<b>Library</b>	xPC Target library for CAN
<b>Description</b>	The Receive driver block retrieves data from a CAN network to be used within a block model.

---

**Tip** To process a remote frame, use the CAN Unpack block with the **Output remote** check box selected to enable the Remote port. When the Remote port becomes true, send the current value to the requesting subsystem.

---

## 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 2047. 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

# Softing CAN-AC2-ISA with Philips PCA 82C200 Receive

in the Softing user manual. Refer to the manual for more information. The function return codes are:

Code	Description
0	No new data received.
1	Data frame received.
2	Remote frame received.
-1	Receive data frame overrun.
-2	Receive remote frame overrun.
-3	Object not active.
-7	Timeout firmware communication
-99	Board not initialized.

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 timeout 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

# Softing CAN-AC2-ISA with Philips PCA 82C200 Receive

---

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.

You can use as many instances of the Receive block in the model as required. 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-AC2 (ISA) with 82527 Controller**

The CAN-AC2 (ISA) driver blocks support the Softing 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 CAN-AC2-ISA Intel527 block group contains the three available CAN blocks: Setup, Send, and Receive.

# Softing CAN-AC2-ISA with Intel 82527 Setup

<b>Purpose</b>	Softing CAN-AC2-ISA with Intel 82527 Setup block
<b>Library</b>	xPC Target library for CAN
<b>Description</b>	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.
<b>Block Parameters</b>	<p><b>CAN 1 - baud rate</b> Defines the most common baud rates for CAN port 1. If your model requires a special baud rate, select the value <code>User defined</code>. In this case, use the <b>CAN 1 - user-defined baud rate</b> parameter to provide the four values for the timing information. The vector elements have the following meanings:</p> <p>[ Prescaler, Synchronization-Jump-Width, Time-Segment-1, Time-Segment-2 ]</p> <p>For more information about these values, see the Softing user manual for this board.</p> <p><b>CAN 2 - baud rate</b> Defines the most common baud rates for CAN port 1. If your model requires a special baud rate, select the value <code>User defined</code>. In this case, use <b>CAN 1 - user-defined baud rate</b> parameter the four values for the timing information. The vector elements have the following meanings:</p> <p>[ Prescaler, Synchronization-Jump-Width, Time-Segment-1, Time-Segment-2 ]</p> <p>For more information about these values, see the Softing user manual for this board.</p> <p><b>Initialization command structure</b> Define CAN messages sent during initialization and termination of the Setup block.</p> <p><b>Termination</b> Define CAN messages sent during termination of the Setup block.</p>

# Softing CAN-AC2-ISA with Intel 82527 Setup

---

## 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 (C8000 to D8000) 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

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 when you use the loopback model provided to test the board and drivers.

# Softing CAN-AC2-ISA with Intel 82527 Send

---

<b>Purpose</b>	Softing CAN-AC2-ISA with Intel 82527 Send block
<b>Library</b>	xPC Target Library for CAN
<b>Description</b>	The Send driver block transmits data to a CAN network from within a block model.

---

**Tip** To prepare a remote frame, use the CAN Pack block with the **Remote frame** check box selected. To request the current value, send the remote frame to the responding subsystem.

---

## 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 2047 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.

# Softing CAN-AC2-ISA with Intel 82527 Send

---

## 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 selected, the block shows as many output ports as input ports. The data type of each output port is a double and the value is identical to the return argument of function `CANPC_write_object(...)`, described in the Softing user manual. Refer to the manual for more information. The function return codes are:

Code	Description
0	Function executed without detecting an error.
-1	Request overrun.
-4	Timeout firmware communication.
-99	Board not initialized.

## 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 required. 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

---

<b>Purpose</b>	Softing CAN-AC2-ISA with Intel 82527 Receive block
<b>Library</b>	xPC Target Library for CAN
<b>Description</b>	The Receive driver block retrieves data from a CAN network to be used within a block model.

---

**Tip** To process a remote frame, use the CAN Unpack block with the **Output remote** check box selected to enable the Remote port. When the Remote port becomes true, send the current value to the requesting subsystem.

---

## 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 2047 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.

# Softing CAN-AC2-ISA with Intel 82527 Receive

---

## 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 function return codes are:

Code	Description
0	No new data received.
1	Data frame received.
2	Remote frame received.
-1	Receive data frame overrun.
-2	Receive remote frame overrun.
-3	Object not active.
-7	Timeout firmware communication
-99	Board not initialized.

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 timeout 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

# Softing CAN-AC2-ISA with Intel 82527 Receive

---

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 required. 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-AC2-PCI with SJA1000 Controller**

The CAN-AC2-PCI driver blocks support the Softing 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 CAN-AC2-PCI SJA 1000 block group contains the three available CAN blocks: Setup, Send, and Receive, plus a FIFO Mode group, which is discussed in “FIFO Mode” on page 8-2.

# 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 16 boards for each target system, making up to 32 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 computer, you can use the board number (1...16) 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 your model requires a special baud rate, 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 ]

# Softing CAN-AC2-PCI with SJA1000 Setup

---

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

## **CAN 1 - user defined baud rate**

See **CAN 1- baud rate**.

## **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 your model requires a special baud rate, select the value **User defined**. 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.

## **CAN 2 - user defined baud rate**

See **CAN 2 - baud rate**.

## **Initialization command structure and Termination**

Defines CAN messages sent during initialization and termination of the Setup block.

## **Termination**

Defines CAN messages sent during termination of the Setup block.

# Softing CAN-AC2-PCI with SJA1000 Setup

---

## Show bus-off status output

Select this check box to enable the bus-off status output port. The output signal can be 1 (bus-off state); otherwise, it is 0. If the bus-off status persists, the block optionally initiates recovery, depending on the setting of **Bus-off recovery**.

Clear this check box to disable the output port.

## Bus-off recovery

Use this parameter to specify how the model is to perform bus-off recovery for the CAN network.

From the list, select:

- Off

Does not perform bus-off recovery.

- Auto

Upon detection, the model waits 1 second to see if the bus-off state persists. After 1 second, bus-off recovery occurs.

- Manual Trigger Input

Enables you to input a signal when you want bus-off recovery to occur. Selecting this option creates an input port for 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). When autosearch is set, the xPC Target kernel finds the board regardless of the PCI slot it is plugged into. If two or more boards of this type are in the target computer, 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`.

## Softing CAN-AC2-PCI with SJA1000 Setup

---

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 when you use the loopback 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.  
To use CAN\_MESSAGE data types to transmit messages, first use the CAN Pack block to create CAN messages for transmission.

---

## Note

- Do not mix message data types with each send block. Either transmit only CAN\_MESSAGE data types, or transmit only 8 bytes in double data types.
- To prepare a remote frame, use the CAN Pack block with the **Remote frame** check box selected. To request the current value, send the remote frame to the responding subsystem.

## 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.

If the CAN message data type is a CAN\_MESSAGE data type, this range must match the range packed in the message parameter (for example, the message from the CAN Pack block).

## 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 2047 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.

- If the CAN message data type is CAN\_MESSAGE, connect each input port to one CAN Pack block. Verify that the **Identifier** parameter for each CAN Pack block matches the send block input port. Do not connect a vector of messages to an input port. Each input port number also indicates sending priority, with the lowest number having the highest priority.

If the CAN message data type is a CAN\_MESSAGE data type, these identifiers must match those packed in the message parameter (for example, the message from the CAN Pack block).

- If the CAN message data type is not CAN\_MESSAGE data type, 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 0 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

# Softing CAN-AC2-PCI with SJA1000 Send

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.

If the CAN message data type is a CAN\_MESSAGE data type, these identifiers must match those packed in the message parameter (for example, the message from the CAN Pack block).

## 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, given in the table. Refer to the Softing manual for more information. The function return codes are:

Code	Description
0	Function executed without detecting an error.
-1	Request overrun.
-2	CAN_MESSAGE data type only. Input port message from CAN Pack block has incorrect <b>Identifiers</b> or <b>CAN identifier range</b> value. This is not a Softing standard.
-3	CAN_MESSAGE data type only. Input port message from CAN Pack block has incorrect <b>Data frame sizes</b> value. This is not a Softing standard.
-4	Timeout firmware communication.
-99	Board not initialized.

## Sample time

Defines the sample time at which the Send block is executed during a model (target application) run.

## Softing CAN-AC2-PCI with SJA1000 Send

---

You can use as many instances of the Send block in the model as required. 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 required.

To use CAN\_MESSAGE data types to transmit messages, use the CAN Unpack block to unpack individual signals from the received CAN messages.

---

## Note

- Do not mix message data types in a block. Each block can receive only CAN\_MESSAGE data types, or receive only 8 bytes in double data types.
- To process a remote frame, use the CAN Unpack block with the **Output remote** check box selected to enable the Remote port. When the Remote port becomes true, send the current value to the requesting subsystem.

## 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.

### 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.

If the CAN message data type is a CAN\_MESSAGE data type, this range must match the **Identifier type** parameter of the connected CAN Unpack block.

## 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 2047 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.

## Output port options

If the CAN message data type is CAN\_MESSAGE, connect each output port to one CAN Unpack block. Verify that the **Identifier** parameter for each CAN Unpack block matches the receive block output port.

If the CAN message data type is not CAN\_MESSAGE data type, 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.

From the list, select:

### Data

Output vector of one double with just data.

# Softing CAN-AC2-PCI with SJA1000 Receive

---

## Data - Status

Output vector of two doubles. First element of the vector contains the data, the second element contains the status.

## Data - Timestamp

Output vector of two doubles. First element of the vector contains the data, the second element contains the timestamp.

## Data - Status - Timestamp

Output vector of three doubles. First element of the vector contains the data, the second element contains the status, the third element contains the timestamp.

## CAN\_MESSAGE data type

One CAN\_MESSAGE. You can extract the data, status, and timestamp from this message using a CAN Unpack block.

Refer to the Softing manual for more information. The function return codes are:

Code	Description
0	No new data received.
1	Data frame received.
2	Remote frame received.
-1	Receive data frame overrun.
-2	Receive remote frame overrun.
-3	Object not active.
-7	Timeout firmware communication
-99	Board not initialized.

## 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 Receive block in the model as required. For example, by using two instances of the block, you can check for CAN messages at different sample times. Or you can use multiple instances to structure your model more efficiently.

## **CAN-AC2-104 with SJA1000 Controller**

The CAN-AC2-104 driver blocks support the Softing CAN-AC2-104 (PC/104) board. 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 CAN-AC2-104 SJA 1000 block group contains the CAN blocks: Setup, Send, and Receive, plus a FIFO Mode group, which is discussed in “FIFO Mode” on page 8-2.

# Softing CAN-AC2-104 with SJA1000 Setup

---

<b>Purpose</b>	Softing CAN-AC2-104 with SJA1000 Setup block
<b>Library</b>	xPC Target Library for CAN
<b>Description</b>	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.
<b>Block Parameters</b>	<p><b>Board</b></p> <p>Defines the board being accessed by this driver block instance. If multiple boards are present in the target computer, you can use the board number (1...3) to differentiate the boards. The physical board referenced by the board number depends on the <b>PCI Slot</b> 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 <b>I/O base address</b> parameter.</p> <p><b>CAN 1 - baud rate</b></p> <p>Defines the most common baud rates for CAN port 1. If your model requires a special baud rate, select the value <b>User defined</b>. In this case, use the <b>CAN 1 - user defined baud rate</b> parameter to provide the four values for the timing information. The vector elements have the following meanings:</p> <p>[ Prescaler, Synchronization-Jump-Width, Time-Segment-1, Time-Segment-2 ]</p> <p>For more information about these values, see the Softing user manual for this board.</p> <p><b>CAN 1 - user defined baud rate</b></p> <p>See <b>CAN 1- baud rate</b>.</p> <p><b>CAN 2 - baud rate</b></p> <p>Defines the most common baud rates for CAN port 2. If your model requires a special baud rate, select the value <b>User defined</b>. In this case, use the <b>CAN 2 - user defined baud rate</b> parameter</p>

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 - user defined baud rate**

See **CAN 2 - baud rate**.

## **Initialization command structure and Termination**

Define CAN messages sent during initialization and termination of the Setup block.

## **Show bus-off status output**

Select this check box to enable the bus-off status output port. The output signal can be 1 (bus-off state); otherwise, it is 0. If the bus-off status persists, the block optionally initiates recovery, depending on the setting of **Bus-off recovery**.

Clear this check box to disable the output port.

## **Bus-off recovery**

Use this parameter to specify how the model is to perform bus-off recovery for the CAN network.

From the list, select:

- Off

Does not perform bus-off recovery.

- Auto

Upon detection, the model waits 1 second to see if the bus-off state persists. After 1 second, bus-off recovery occurs.

- Manual Trigger Input

# Softing CAN-AC2-104 with SJA1000 Setup

---

Enables you to input a signal when you want bus-off recovery to occur. Selecting this option creates an input port for the Setup block.

## **Termination**

Define CAN messages sent during termination of the Setup block.

## **I/O base address**

Defines the I/O base address of the board to be accessed by this block instance. The I/O base address is given by the DIP switch setting on the board itself. The I/O address range is 3 bytes and is mainly used to transfer 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:

C8000 - D8000

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 when you use the loopback model provided to test the board and drivers.

# Softing CAN-AC2-104 with SJA1000 Setup

---

## **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.

To use CAN\_MESSAGE data types to transmit messages, first use the CAN Pack block to create CAN messages for transmission.

---

## Note

- Do not mix message data types with each send block. Either transmit only CAN\_MESSAGE data types, or transmit only 8 bytes in double data types.
- To prepare a remote frame, use the CAN Pack block with the **Remote frame** check box selected. To request the current value, send the remote frame to the responding subsystem.

## 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.

### 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.

If the CAN message data type is a CAN\_MESSAGE data type, this range must match the range packed in the message parameter (for example, the message from the CAN Pack block).

## 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 2047 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.

- If the CAN message data type is CAN\_MESSAGE, connect each input port to one CAN Pack block. Verify that the **Identifier** parameter for each CAN Pack block matches the send block input port. Do not connect a vector of messages to an input port. Each input port number also indicates sending priority, with the lowest number having the highest priority.

If the CAN message data type is a CAN\_MESSAGE data type, these identifiers must match those packed in the message parameter (for example, the message from the CAN Pack block).

- If the CAN message data type is not CAN\_MESSAGE data type, 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 0 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

# Softing CAN-AC2-104 with SJA1000 Send

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.

If the CAN message data type is a CAN\_MESSAGE data type, these identifiers must match those packed in the message parameter (for example, the message from the CAN Pack block).

## 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, given in the table. Refer to the Softing manual for more information. The function return codes are:

Code	Description
0	Function executed without detecting an error.
-3	CAN_MESSAGE data type only. Input port message from CAN Pack block has incorrect <b>Data frame sizes</b> value.
-1	Request overrun.
-2	CAN_MESSAGE data type only. Input port message from CAN Pack block has incorrect <b>Identifiers</b> or <b>CAN identifier range</b> value. This is not a Softing standard.
-3	CAN_MESSAGE data type only. Input port message from CAN Pack block has incorrect <b>Data frame sizes</b> value. This is not a Softing standard.
-4	Timeout firmware communication.
-99	Board not initialized.

## **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 required. 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 required.

To use CAN\_MESSAGE data types to transmit messages, use the CAN Unpack block to unpack individual signals from the received CAN messages.

---

## Note

- Do not mix message data types in a block. Each block can receive only CAN\_MESSAGE data types, or receive only 8 bytes in double data types.
- To process a remote frame, use the CAN Unpack block with the **Output remote** check box selected to enable the Remote port. When the Remote port becomes true, send the current value to the requesting subsystem.

## 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.

If the CAN message data type is a CAN\_MESSAGE data type, this range must match the **Identifier type** parameter of the connected CAN Unpack block.

## 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 2047 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.

## Output port options

If the CAN message data type is CAN\_MESSAGE, connect each output port to one CAN Unpack block. Verify that the **Identifier** parameter for each CAN Unpack block matches the receive block output port.

If the CAN message data type is not CAN\_MESSAGE data type, 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.

From the list, select:

### Data

Output vector of one double with just data.

### Data - Status

Output vector of two doubles. First element of the vector contains the data, the second element contains the status.

# Softing CAN-AC2-104 with SJA1000 Receive

---

## Data - Timestamp

Output vector of two doubles. First element of the vector contains the data, the second element contains the timestamp.

## Data - Status - Timestamp

Output vector of three doubles. First element of the vector contains the data, the second element contains the status, the third element contains the timestamp.

## CAN\_MESSAGE data type

One CAN\_MESSAGE. You can extract the data, status, and timestamp from this message using a CAN Unpack block.

Refer to the Softing manual for more information. The function return codes are:

Code	Description
0	No new data received.
1	Data frame received.
2	Remote frame received.
-1	Receive data frame overrun.
-2	Receive remote frame overrun.
-3	Object not active.
-7	Timeout firmware communication
-99	Board not initialized.

## 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 required. For example, by using two instances of the block, you can check for 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.

# CAN Bit-Packing

---

**Purpose** CAN Bit-Packing block

**Library** xPC Target Library for CAN

**Description** This block is for message data of type double. This block constructs CAN data frames, and its output port is ordinarily 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.

---

**Note** Do not use this block to construct CAN\_MESSAGE data types or to prepare a remote frame. Use the CAN Pack block.

---

## 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.

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

---

**Note** The example in this topic describes the use of the Softing CAN blocks with standard double data types for message storage. For an alternative and easier way to pack CAN\_MESSAGE data types, use the CAN Pack block.

---

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.

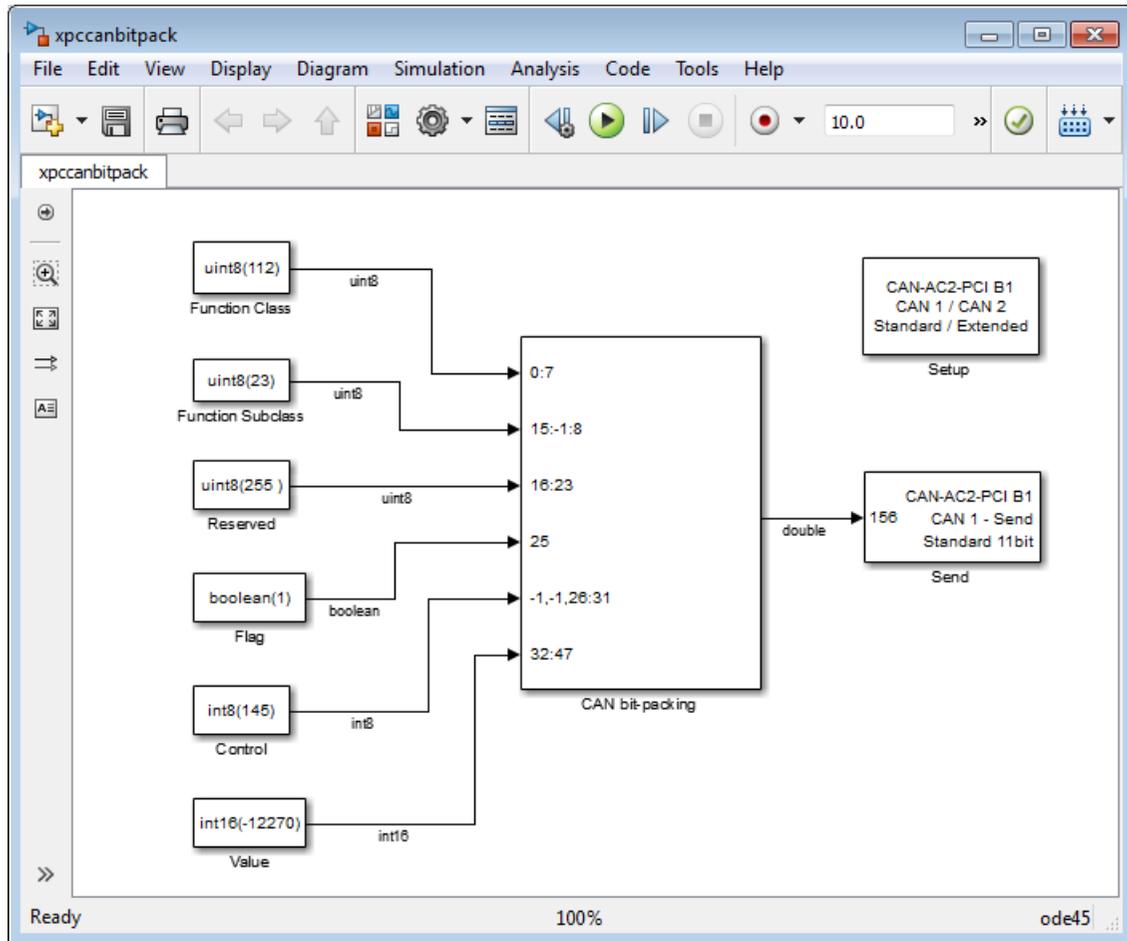
Byte 0	Function class of type uint8
Byte 1	Function subclass of type uint8 with reversed bit order
Byte 2	Reserved, all bits must be 1
Byte 3	Bit 0 must be 0, Bit 1 must be a boolean (flag), bits 2 to 7 must be bit 2 to 7 of an incoming int8 value (control)
Byte 4 and 5	Value of type int16

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] }
```

# CAN Bit-Packing

And the Simulink model simulating the required behavior would be as shown.



- The function class is 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 function subclass is 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 easiest to define this mapping by the MATLAB colon operator (:) and an increment of -1.
- 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 flag is 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 control is 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 value is 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 (:).

# CAN Bit-Packing

---

- The output of the CAN bit-packing block 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 6 bytes, meaning that only the first six bytes of the incoming double value are transmitted as part of the CAN message.

**Purpose** CAN Bit-Unpacking block

**Library** xPC Target Library for CAN

**Description** This block is for message data of type double. This block extracts 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.

---

**Note** Do not use this block to unpack CAN\_MESSAGE data types or to process a remote frame. Use the CAN Unpack block.

---

## 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`

# CAN Bit-Unpacking

---

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.

If you unpack the data frame into a signed type (`int8`, `int16`, or `int32`), the block performs sign extension. 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 sign extension. This functionality enables you to pack and unpack negative numbers without losing precision. In the preceding example, you can pack and unpack numbers in the range `[-16 : 15]` (a fictitious `int5` type).

## Example

---

**Note** The example in this topic describes the use of the Softing CAN blocks with standard double data types for message storage. For an alternative and easier way to unpack `CAN_MESSAGE` data types, use the CAN Unpack block.

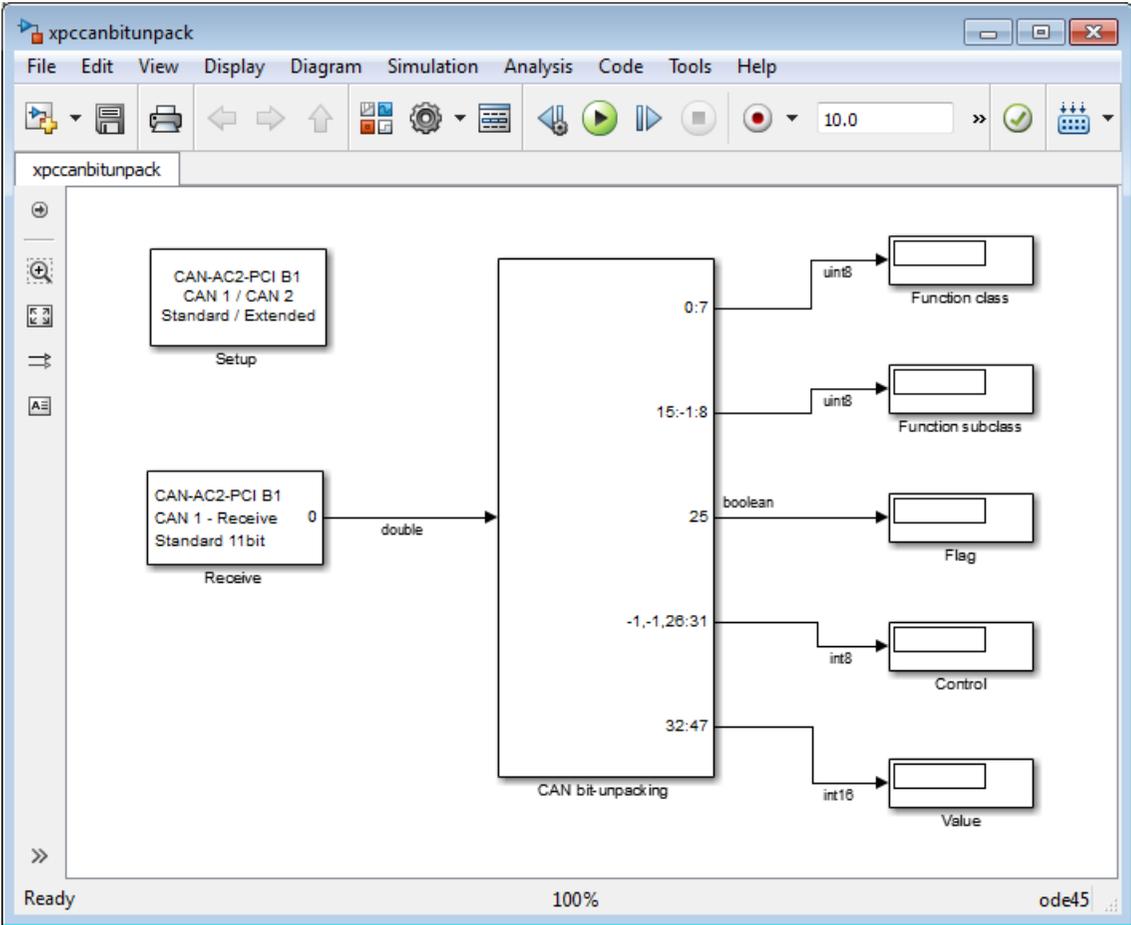
---

The functionality of the block is easiest explained by means of an example. The same example as used above shows 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. 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] }
```

and the data type edit field as

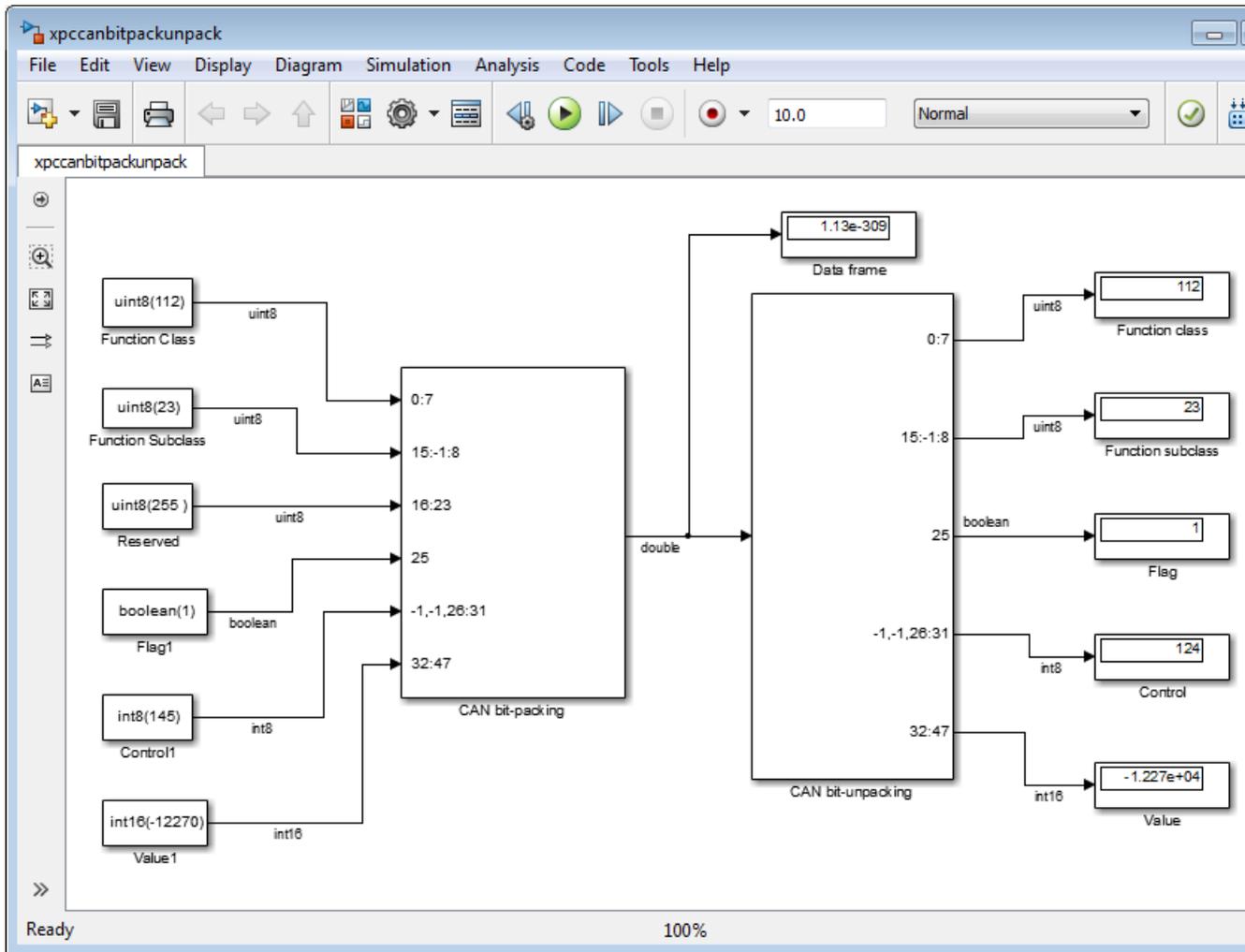
```
{ 'uint8' , 'uint8' , 'boolean' , 'int8' , 'int16' }
```



This leads to the following Simulink model.

In many cases it makes sense to test the bit-packing and unpacking operations in a Simulink model (simulation) before building the target application. Both blocks work the same way either in the Simulink system 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



**Purpose** CAN Timeout Detection block

**Library** xPC Target Library for CAN

**Description** This block uses the timestamp information to calculate the timeout condition. To use this block, select the Output timestamp check box of a CAN Unpack block. Then connect the **Timestamp output** port to the input port of the CAN Timeout Detection block.

For examples of this block usage, see

- “Timeouts When Receiving CAN Messages” on page 7-88
- `xpccanpcis` — Loopback example for the CAN-AC2-PCI board using CAN\_MESSAGE data types for message storage
- `xpccanpci` — Loopback example for the CAN-AC2-PCI board using standard double data types for message storage
- `xpccan104s` — Loopback example for the CAN-AC2-104 board using CAN\_MESSAGE data types for message storage
- `xpccanpc104` — Loopback example for the CAN-AC2-104 board using standard double data types for message storage

## Block Parameters

### Timeout

Specify the timeout value, in seconds. The output of the block is:

- 0, if no timeout has been detected
- 1, if a timeout has been detected

# CAN Pack

## Purpose

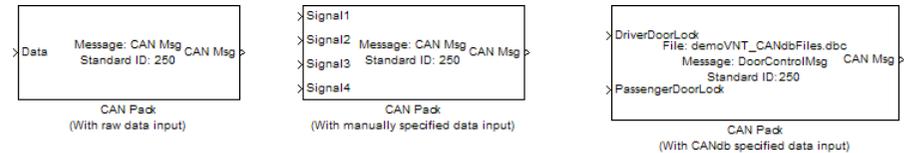
Pack individual signals into CAN message

## Library

CAN Communication

Embedded Coder/ Embedded Targets/ Host Communication

## Description



The CAN Pack block loads signal data into a message at specified intervals during the simulation.

---

**Note** To use this block, you also need a license for Simulink software.

---

CAN Pack block has one input port by default. The number of block inputs is dynamic and depends on the number of signals you specify for the block. For example, if your block has four signals, it has four block inputs.

This block has one output port, CAN Msg. The CAN Pack block takes the specified input parameters and packs the signals into a message.

## Other Supported Features

The CAN Pack block supports:

- The use of Simulink Accelerator™ mode. Using this feature, you can speed up the execution of Simulink models.
- The use of model referencing. Using this feature, your model can include other Simulink models as modular components.
- Code generation using Simulink Coder to deploy models to targets.

---

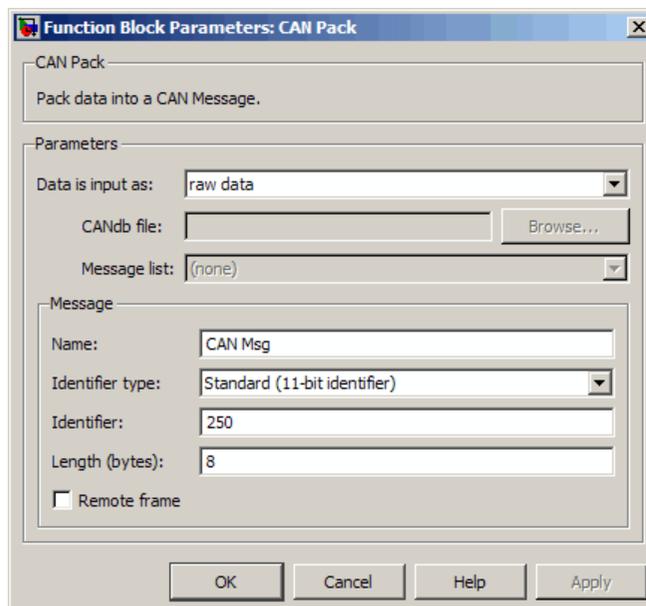
**Note** Code generation is not supported if your signal information consists of signed or unsigned integers greater than 32-bits long.

---

For more information on these features, see the Simulink documentation.

## Dialog Box

Use the Function Block Parameters dialog box to select your CAN Pack block parameters.



### Parameters

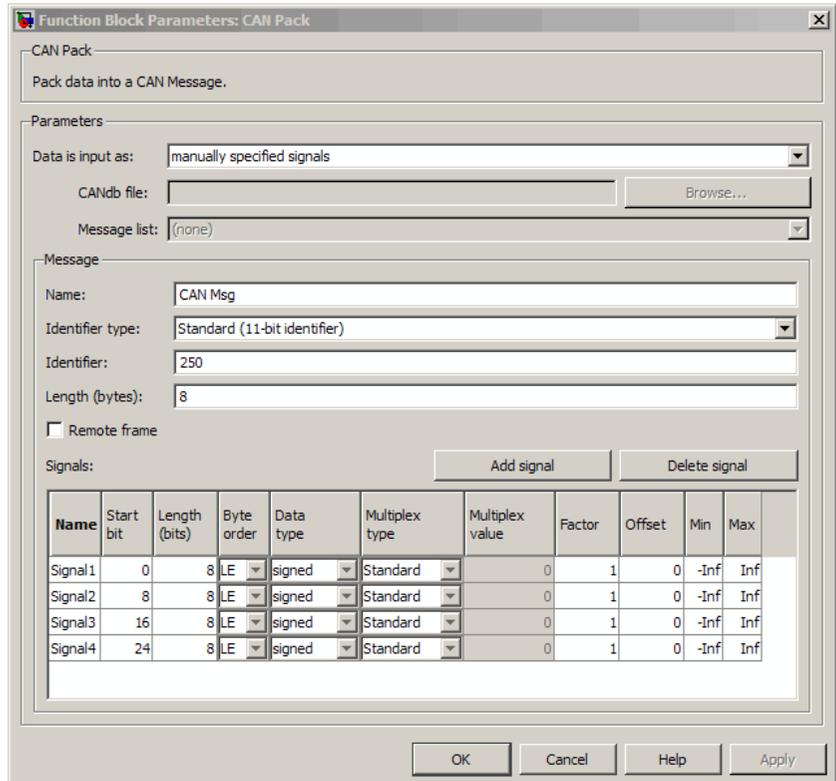
#### Data is input as

Select your data signal:

- **raw data:** Input data as a uint8 vector array. If you select this option, you only specify the message fields. all other signal

parameter fields are unavailable. This option opens only one input port on your block.

- **manually specified signals:** Allows you to specify data signal definitions. If you select this option, use the **Signals** table to create your signals. The number of block inputs depends on the number of signals you specify.



The dialog box is titled "Function Block Parameters: CAN Pack". It contains the following sections:

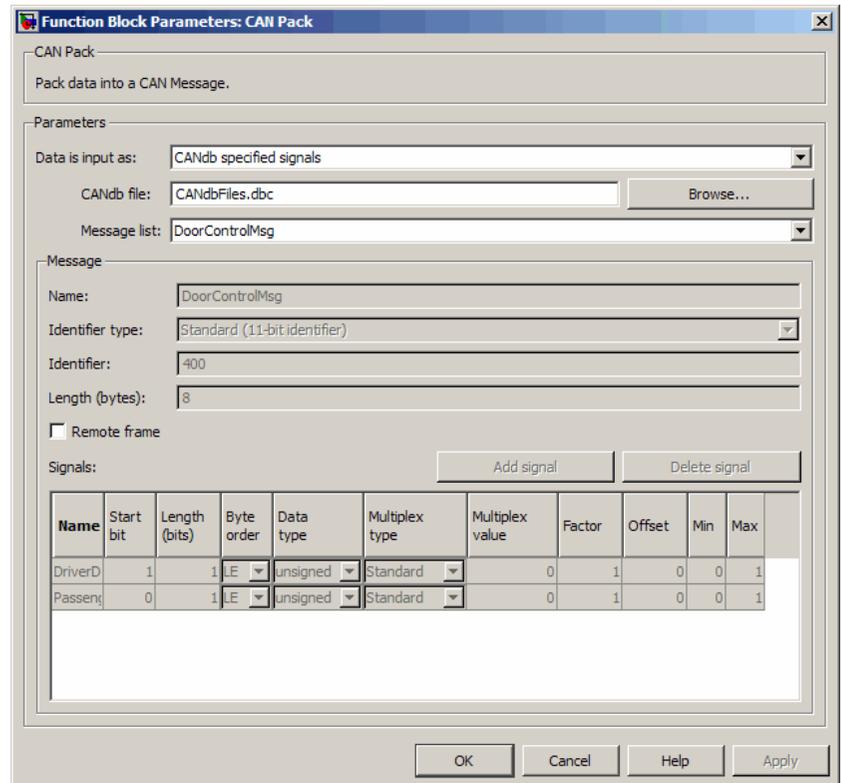
- CAN Pack:** Pack data into a CAN Message.
- Parameters:**
  - Data is input as: manually specified signals (dropdown)
  - CANdb file: [text field] Browse...
  - Message list: (none) (dropdown)
- Message:**
  - Name: CAN Msg
  - Identifier type: Standard (11-bit identifier) (dropdown)
  - Identifier: 250
  - Length (bytes): 8
  - Remote frame
- Signals:** Add signal, Delete signal
- Signals Table:**

Name	Start bit	Length (bits)	Byte order	Data type	Multiplex type	Multiplex value	Factor	Offset	Min	Max
Signal1	0	8	LE	signed	Standard	0	1	0	-Inf	Inf
Signal2	8	8	LE	signed	Standard	0	1	0	-Inf	Inf
Signal3	16	8	LE	signed	Standard	0	1	0	-Inf	Inf
Signal4	24	8	LE	signed	Standard	0	1	0	-Inf	Inf

Buttons: OK, Cancel, Help, Apply

- **CANdb specified signals:** Allows you to specify a CAN database file that contains message and signal definitions. If you select this option, select a CANdb file. The number of

block inputs depends on the number of signals specified in the CANdb file for the selected message.



## CANdb file

This option is available if you specify that your data is input via a CANdb file in the **Data is input as** list. Click **Browse** to find the CANdb file on your system. The message list specified in the CANdb file populates the **Message** section of the dialog box. The CANdb file also populates the **Signals** table for the selected message.

---

**Note** File names that contain non-alphanumeric characters such as equal signs, ampersands, and so forth are not valid CAN database file names. You can use periods in your database name. Rename CAN database files with non-alphanumeric characters before you use them.

---

## Message list

This option is available if you specify that your data is input via a CANdb file in the **Data is input as** field and you select a CANdb file in the **CANdb file** field. Select the message to display signal details in the **Signals** table.

## Message

### Name

Specify a name for your CAN message. The default is **CAN Msg**. This option is available if you choose to input raw data or manually specify signals. This option is unavailable if you choose to use signals from a CANdb file.

### Identifier type

Specify whether your CAN message identifier is a **Standard** or an **Extended** type. The default is **Standard**. A standard identifier is an 11-bit identifier and an extended identifier is a 29-bit identifier. This option is available if you choose to input raw data or manually specify signals. For **CANdb** specified signals, the **Identifier type** inherits the type from the database.

### Identifier

Specify your CAN message ID. This number must be a positive integer from 0 through 2047 for a standard identifier and from 0 through 536870911 for an extended identifier. You can also specify hexadecimal values using the **hex2dec** function. This option is available if you choose to input raw data or manually specify signals.

**Length (bytes)**

Specify the length of your CAN message from 0 to 8 bytes. If you are using **CANdb specified signals** for your data input, the CANdb file defines the length of your message. If not, this field defaults to 8. This option is available if you choose to input raw data or manually specify signals.

**Remote frame**

Specify the CAN message as a remote frame.

**Signals Table**

This table appears if you choose to specify signals manually or define signals using a CANdb file.

If you are using a CANdb file, the data in the file populates this table automatically and you cannot edit the fields. To edit signal information, switch to manually specified signals.

If you have selected to specify signals manually, create your signals manually in this table. Each signal you create has the following values:

**Name**

Specify a descriptive name for your signal. The Simulink block in your model displays this name. The default is **Signal [row number]**.

**Start bit**

Specify the start bit of the data. The start bit is the least significant bit counted from the start of the message data. The start bit must be an integer from 0 through 63.

**Length (bits)**

Specify the number of bits the signal occupies in the message. The length must be an integer from 1 through 64.

**Byte order**

Select either of the following options:

- **LE:** Where the byte order is in little-endian format (Intel). In this format you count bits from the start, which is the least

significant bit, to the most significant bit, which has the highest bit index. For example, if you pack one byte of data in little-endian format, with the start bit at 20, the data bit table resembles this figure.

Bit Number		Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Data Byte Number	Byte 0	7	6	5	4	3	2	1	0
	Byte 1	15	14	13	12	11	10	9	8
	Byte 2	23	22	21	20	19	18	17	16
	Byte 3	31	30	29	28	27	26	25	24
	Byte 4	39	38	37	36	35	34	33	32
	Byte 5	47	46	45	44	43	42	41	40
	Byte 6	55	54	53	52	51	50	49	48
	Byte 7	63	62	61	60	59	58	57	56

### Little-Endian Byte Order Counted from the Least Significant Bit to the Highest Address

- BE: Where byte order is in big-endian format (Motorola®). In this format you count bits from the start, which is the least significant bit, to the most significant bit. For example, if you

pack one byte of data in big-endian format, with the start bit at 20, the data bit table resembles this figure.

		Bit Number							
		Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Data Byte Number	Byte 0	7	6	5	4	3	2	1	0
	Byte 1	15	14	13	12	11	10	9	8
	Byte 2	23	22	21	20	19	18	17	16
	Byte 3	31	30	29	28	27	26	25	24
	Byte 4	39	38	37	36	35	34	33	32
	Byte 5	47	46	45	44	43	42	41	40
	Byte 6	55	54	53	52	51	50	49	48
	Byte 7	63	62	61	60	59	58	57	56

### Big-Endian Byte Order Counted from the Least Significant Bit to the Lowest Address

#### Data type

Specify how the signal interprets the data in the allocated bits.  
Choose from:

- signed (default)
- unsigned

- single
- double

## Multiplex type

Specify how the block packs the signals into the CAN message at each timestep:

- **Standard:** The signal is packed at each timestep.
- **Multiplexor:** The Multiplexor signal, or the mode signal is packed. You can specify only one Multiplexor signal per message.
- **Multiplexed:** The signal is packed if the value of the Multiplexor signal (mode signal) at run time matches the configured **Multiplex value** of this signal.

For example, a message has four signals with the following types and values.

Signal Name	Multiplex Type	Multiplex Value
Signal-A	Standard	N/A
Signal-B	Multiplexed	1
Signal-C	Multiplexed	0
Signal-D	Multiplexor	N/A

In this example:

- The block packs Signal-A (Standard signal) and Signal-D (Multiplexor signal) in every timestep.
- If the value of Signal-D is 1 at a particular timestep, then the block packs Signal-B along with Signal-A and Signal-D in that timestep.
- If the value of Signal-D is 0 at a particular timestep, then the block packs Signal-C along with Signal-A and Signal-D in that timestep.

- If the value of Signal-D is not 1 or 0, the block does not pack either of the Multiplexed signals in that timestep.

**Multiplex value**

This option is available only if you have selected the **Multiplex type** to be Multiplexed. The value you provide here must match the Multiplexor signal value at run time for the block to pack the Multiplexed signal. The **Multiplex value** must be a positive integer or zero.

**Factor**

Specify the **Factor** value to apply to convert the physical value (signal value) to the raw value packed in the message. See “Conversion Formula” on page 7-65 to understand how physical values are converted to raw values packed into a message.

**Offset**

Specify the **Offset** value to apply to convert the physical value (signal value) to the raw value packed in the message. See “Conversion Formula” on page 7-65 to understand how physical values are converted to raw values packed into a message.

**Min**

Specify the minimum physical value of the signal. The default value is `-inf` (negative infinity). You can specify a number for the minimum value. See “Conversion Formula” on page 7-65 to understand how physical values are converted to raw values packed into a message.

**Max**

Specify the maximum physical value of the signal. The default value is `inf`. You can specify a number for the maximum value. See “Conversion Formula” on page 7-65 to understand how physical values are converted to raw values packed into a message.

**Conversion Formula**

The conversion formula is

$$\text{raw\_value} = (\text{physical\_value} - \text{Offset}) / \text{Factor}$$

## CAN Pack

---

where `physical_value` is the value of the signal after it is saturated using the specified **Min** and **Max** values. `raw_value` is the packed signal value.

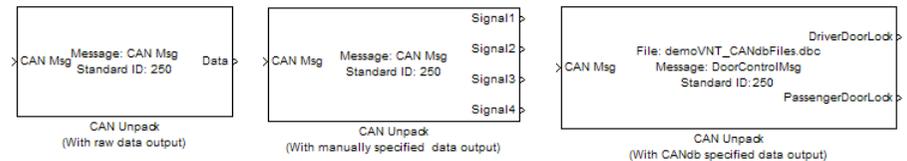
## Purpose

Unpack individual signals from CAN messages

## Library

CAN Communication

Embedded Coder/ Embedded Targets/ Host Communication



## Description

The CAN Unpack block unpacks a CAN message into signal data using the specified output parameters at every timestep. Data is output as individual signals.

---

**Note** To use this block, you also need a license for Simulink software.

---

The CAN Unpack block has one output port by default. The number of output ports is dynamic and depends on the number of signals you specify for the block to output. For example, if your block has four signals, it has four output ports.

## Other Supported Features

The CAN Unpack block supports:

- The use of Simulink Accelerator mode. Using this feature, you can speed up the execution of Simulink models.
- The use of model referencing. Using this feature, your model can include other Simulink models as modular components.
- Code generation using Simulink Coder to deploy models to targets.

# CAN Unpack

---

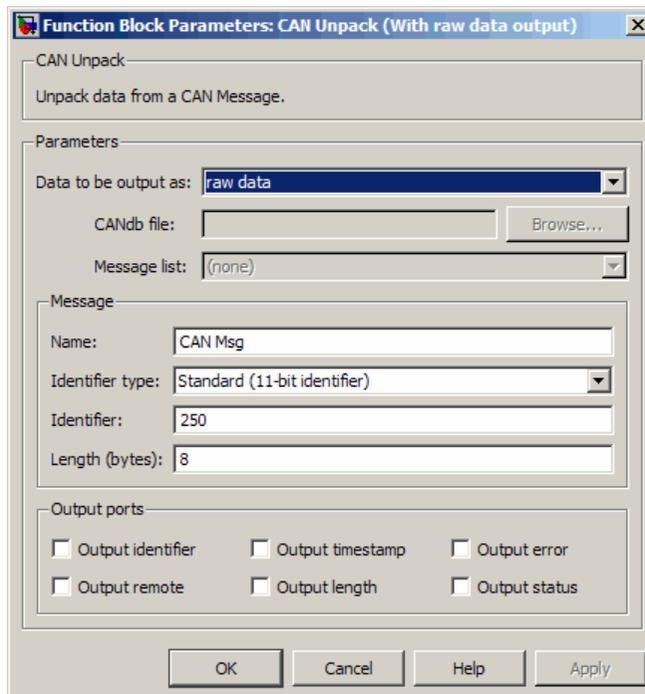
**Note** Code generation is not supported if your signal information consists of signed or unsigned integers greater than 32-bits long.

---

For more information on these features, see the Simulink documentation.

## Dialog Box

Use the Function Block Parameters dialog box to select your CAN message unpacking parameters.



## Parameters

**Data to be output as**  
Select your data signal:

- **raw data:** Output data as a uint8 vector array. If you select this option, you only specify the message fields. The other signal parameter fields are unavailable. This option opens only one output port on your block.
- **manually specified signals:** Allows you to specify data signals. If you select this option, use the Signals table to create your signals message manually.

Function Block Parameters: CAN Unpack (With manually specified data output)

CAN Unpack  
Unpack data from a CAN Message.

Parameters

Data to be output as: manually specified signals

CANdb file:  Browse...

Message list: (none)

Message

Name: CAN Msg

Identifier type: Standard (11-bit identifier)

Identifier: 250

Length (bytes): 8

Signals:

Name	Start bit	Length (bits)	Byte order	Data type	Multiplex type	Multiplex value	Factor	Offset	Min	Max
Signal1	0	8	LE	signed	Standard	0	1	0	-Inf	Inf
Signal2	8	8	LE	signed	Standard	0	1	0	-Inf	Inf
Signal3	16	8	LE	signed	Standard	0	1	0	-Inf	Inf
Signal4	24	8	LE	signed	Standard	0	1	0	-Inf	Inf

Output ports

Output identifier  Output timestamp  Output error

Output remote  Output length  Output status

# CAN Unpack

The number of output ports on your block depends on the number of signals you specify. For example, if you specify four signals, your block has four output ports.

- **CANdb specified signals:** Allows you to specify a CAN database file that contains data signals. If you select this option, select a CANdb file.

Function Block Parameters: CAN Unpack (With CANdb specified data output)

CAN Unpack  
Unpack data from a CAN Message.

Parameters

Data to be output as: CANdb specified signals

CANdb file: CANdbFiles.dbc

Message list: DoorControlMsg

Message

Name: DoorControlMsg

Identifier type: Standard (11-bit identifier)

Identifier: 400

Length (bytes): 8

Signals:

Name	Start bit	Length (bits)	Byte order	Data type	Multiplex type	Multiplex value	Factor	Offset	Min	Max
DriverD	1	1	LE	unsigned	Standard	0	1	0	0	1
Passent	0	1	LE	unsigned	Standard	0	1	0	0	1

Output ports

Output identifier  Output timestamp  Output error  
 Output remote  Output length  Output status

The number of output ports on your block depends on the number of signals specified in the CANdb file. For example, if

the selected message in the CANdb file has four signals, your block has four output ports.

## **CANdb file**

This option is available if you specify that your data is input via a CANdb file in the **Data to be output as** list. Click **Browse** to find the CANdb file on your system. The messages and signal definitions specified in the CANdb file populate the **Message** section of the dialog box. The signals specified in the CANdb file populate **Signals** table.

---

**Note** File names that contain non-alphanumeric characters such as equal signs, ampersands, and so forth are not valid CAN database file names. You can use periods in your database name. Rename CAN database files with non-alphanumeric characters before you use them.

---

## **Message list**

This option is available if you specify that your data is to be output as a CANdb file in the **Data to be output as** list and you select a CANdb file in the **CANdb file** field. You can select the message that you want to view. The **Signals** table then displays the details of the selected message.

## **Message**

### **Name**

Specify a name for your CAN message. The default is **CAN Msg**. This option is available if you choose to output raw data or manually specify signals.

### **Identifier type**

Specify whether your CAN message identifier is a **Standard** or an **Extended** type. The default is **Standard**. A standard identifier is an 11-bit identifier and an extended identifier is a 29-bit identifier. This option is available if you choose to output raw

data or manually specify signals. For CANdb-specified signals, the **Identifier type** inherits the type from the database.

## Identifier

Specify your CAN message ID. This number must be an integer from 0 through 2047 for a standard identifier and from 0 through 536870911 for an extended identifier. If you specify `1`, the block unpacks the messages that match the length specified for the message. You can also specify hexadecimal values using the `hex2dec` function. This option is available if you choose to output raw data or manually specify signals.

## Length (bytes)

Specify the length of your CAN message from 0 to 8 bytes. If you are using `CANdb specified signals` for your output data, the CANdb file defines the length of your message. If not, this field defaults to `8`. This option is available if you choose to output raw data or manually specify signals.

## Signals Table

This table appears if you choose to specify signals manually or define signals using a CANdb file.

If you are using a CANdb file, the data in the file populates this table automatically and you cannot edit the fields. To edit signal information, switch to manually specified signals.

If you have selected to specify signals manually, create your signals manually in this table. Each signal you create has the following values:

### Name

Specify a descriptive name for your signal. The Simulink block in your model displays this name. The default is `Signal [row number]`.

### Start bit

Specify the start bit of the data. The start bit is the least significant bit counted from the start of the message. The start bit must be an integer from 0 through 63.

**Length (bits)**

Specify the number of bits the signal occupies in the message. The length must be an integer from 1 through 64.

**Byte order**

Select either of the following options:

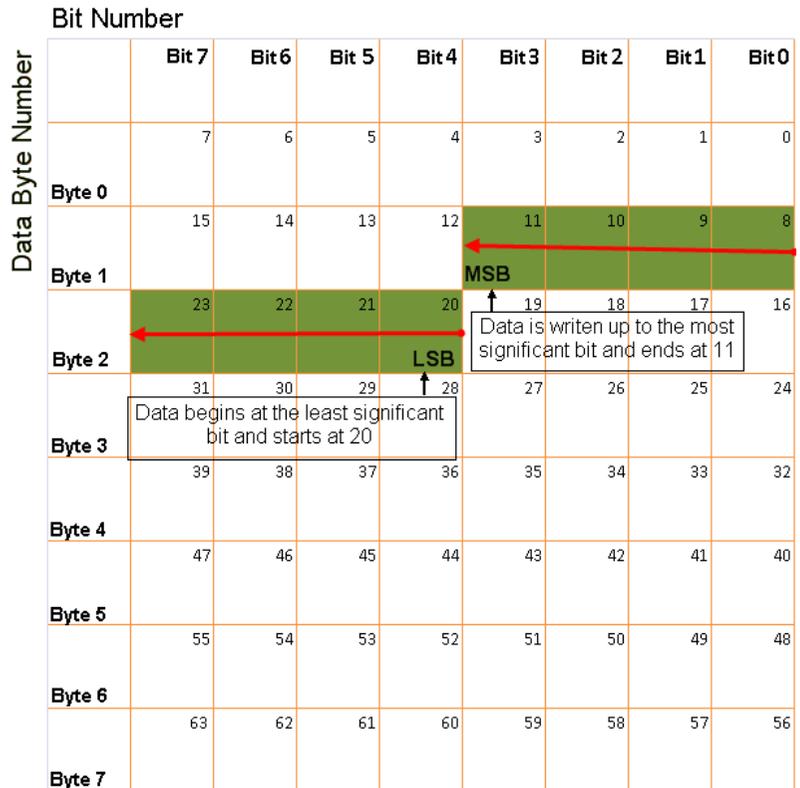
- LE: Where the byte order is in little-endian format (Intel). In this format you count bits from the start, which is the least significant bit, to the most significant bit, which has the highest bit index. For example, if you pack one byte of data in little-endian format, with the start bit at 20, the data bit table resembles this figure.

# CAN Unpack

		Bit Number							
		Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Data Byte Number	Byte 0	7	6	5	4	3	2	1	0
	Byte 1	15	14	13	12	11	10	9	8
	Byte 2	23	22	21	20	19	18	17	16
	Byte 3	31	30	29	28	27	26	25	24
	Byte 4	39	38	37	36	35	34	33	32
	Byte 5	47	46	45	44	43	42	41	40
	Byte 6	55	54	53	52	51	50	49	48
	Byte 7	63	62	61	60	59	58	57	56

## Little-Endian Byte Order Counted from the Least Significant Bit to the Highest Address

- BE: Where the byte order is in big-endian format (Motorola). In this format you count bits from the start, which is the least significant bit, to the most significant bit. For example, if you pack one byte of data in big-endian format, with the start bit at 20, the data bit table resembles this figure.



## Big-Endian Byte Order Counted from the Least Significant Bit to the Lowest Address

### Data type

Specify how the signal interprets the data in the allocated bits.  
Choose from:

- signed (default)
- unsigned
- single
- double

## Multiplex type

Specify how the block unpacks the signals from the CAN message at each timestep:

- **Standard:** The signal is unpacked at each timestep.
- **Multiplexor:** The Multiplexor signal, or the mode signal is unpacked. You can specify only one Multiplexor signal per message.
- **Multiplexed:** The signal is unpacked if the value of the Multiplexor signal (mode signal) at run time matches the configured **Multiplex value** of this signal.

For example, a message has four signals with the following values.

Signal Name	Multiplex Type	Multiplex Value
Signal-A	Standard	N/A
Signal-B	Multiplexed	1
Signal-C	Multiplexed	0
Signal-D	Multiplexor	N/A

In this example:

- The block unpacks Signal-A (Standard signal) and Signal-D (Multiplexor signal) in every timestep.
- If the value of Signal-D is 1 at a particular timestep, then the block unpacks Signal-B along with Signal-A and Signal-D in that timestep.
- If the value of Signal-D is 0 at a particular timestep, then the block unpacks Signal-C along with Signal-A and Signal-D in that timestep.
- If the value of Signal-D is not 1 or 0, the block does not unpack either of the Multiplexed signals in that timestep.

## **Multiplex value**

This option is available only if you have selected the **Multiplex type** to be Multiplexed. The value you provide here must match the Multiplexor signal value at run time for the block to unpack the Multiplexed signal. The **Multiplex value** must be a positive integer or zero.

## **Factor**

Specify the **Factor** value applied to convert the unpacked raw value to the physical value (signal value). See “Conversion Formula” on page 7-78 to understand how unpacked raw values are converted to physical values.

## **Offset**

Specify the **Offset** value applied to convert the physical value (signal value) to the unpacked raw value. See “Conversion Formula” on page 7-78 to understand how unpacked raw values are converted to physical values.

## **Min**

Specify the minimum raw value of the signal. The default value is `-inf` (negative infinity). You can specify a number for the minimum value. See “Conversion Formula” on page 7-78 to understand how unpacked raw values are converted to physical values.

## **Max**

Specify the maximum raw value of the signal. The default value is `inf`. You can specify a number for the maximum value. See “Conversion Formula” on page 7-78 to understand how unpacked raw values are converted to physical values.

## **Output Ports**

Selecting an **Output ports** option adds an output port to your block.

### **Output identifier**

Select this option to output a CAN message identifier. The data type of this port is `uint32`.

## **Output remote**

Select this option to output the message remote frame status.  
This option adds a new output port to the block. The data type of this port is **uint8**.

## **Output timestamp**

Select this option to output the message time stamp. This option adds a new output port to the block. The data type of this port is **double**.

## **Output length**

Select this option to output the length of the message in bytes.  
This option adds a new output port to the block. The data type of this port is **uint8**.

## **Output error**

Select this option to output the message error status. This option adds a new output port to the block. The data type of this port is **uint8**.

## **Output status**

Select this option to output the message received status. The status is 1 if the block receives new message and 0 if it does not.  
This option adds a new output port to the block. The data type of this port is **uint8**.

If you do not select an **Output ports** option, the number of output ports on your block depends on the number of signals you specify.

## **Conversion Formula**

The conversion formula is

$$\text{physical\_value} = \text{raw\_value} * \text{Factor} + \text{Offset}$$

where **raw\_value** is the unpacked signal value. **physical\_value** is the scaled signal value which is saturated using the specified **Min** and **Max** values.

## Model Execution Driven by CAN Messages

In certain applications, the model (target application) execution can be paced by incoming CAN messages. 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 support firing 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. You can then drive the target application by sending CAN messages, interpreting them in the CAN Receive blocks, and firing a hardware interrupt in response.

To set this up, do the following:

- 1 Replace the timer interrupt line in the kernel setup with the board's hardware interrupt line.
- 2 Set up the CAN Setup and CAN Receive blocks.

Both steps are slightly different for each of the three supported CAN boards.

### In this section...

“CAN-AC2 (ISA)” on page 7-79

“CAN-AC2-PCI” on page 7-80

“CAN-AC2-104 (PC/104)” on page 7-81

### 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, select **Simulation > Model Configuration Parameters**.

The Configuration Parameters dialog box is displayed.

- 2** Select node **Code Generation**.
- 3** In the **Target selection** section, set the **System target file** field 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. At the model level, open the **Simulation > Model Configuration Parameters** dialog box. At the **xPC Target options** node, set **Real-time interrupt source** Auto (PCI only). This option enables the xPC Target software to automatically determine the IRQ that the BIOS assigned to the board and use it. Alternatively, 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, select **Simulation > Model Configuration Parameters**.

The Configuration Parameters dialog box is displayed.

- 2** Select node **Code Generation**.
- 3** In the **Target selection** section, set the **System target file** field 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.

Alternatively, select **Auto (PCI only)** to enable the xPC Target software to automatically determine the IRQ that the BIOS assigned to the board and use it.

- 6** In the **I/O board generating the interrupt** field, select `Softing_CAN-AC2-PCI`.
- 7** Click **OK** and save the model.
- 8** 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, select **Simulation > Model Configuration Parameters**.

The Configuration Parameters dialog box is displayed.

- 2** Select node **Code Generation**.
- 3** In the **Target selection** section, set the **System target file** field 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** In the **I/O board generating the interrupt** field, select `Softing_CAN-AC2-104`.
- 7** Click **OK** and save the model.
- 8** 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. The corresponding CAN message on the other CAN node should only be generated if the xPC Target application is running, otherwise unexpected interrupt messages might be displayed on the target screen.

## Initialization and Termination CAN Messages

### In this section...

“Defining Initialization and Termination CAN Messages” on page 7-83

“A/D Converter with CANopen Interface” on page 7-84

### 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 those CAN application layers are not directly supported, communication with those nodes can usually be done over standard CAN messages as long as the nodes have been 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 length of the row vector defines the data frame size.

- If the CAN message data type is a `CAN_MESSAGE` data type, the value of the frame size vector must match the **Length** parameter of the connected CAN Pack block.
- If the CAN message data type is a double, this field 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 1 byte. In this case, 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).

**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 specify those idle times.

## **A/D Converter with CANopen Interface**

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 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. The xPC Target software does not support the CANopen protocol.

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 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.

## CAN Data Frames

CAN data frames have a maximum size of 8 bytes (64 bits). The CAN driver blocks in the xPC Target I/O block library use Simulink signals of data type `CAN_MESSAGE` or `double` to propagate data frames as an entity. 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 sizes.

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 `CAN Bit-Packing` and `CAN Bit-Unpacking` blocks work with data frames using `double` data types. They provide support for legacy models.
- To construct and extract `CAN_MESSAGE` data types, use the `CAN Pack` and `CAN Unpack` blocks.

---

### Tip

- To prepare a remote frame, use the `CAN Pack` block with the **Remote frame** check box selected. To request the current value, send the remote frame to the responding subsystem.
  - To process a remote frame, use the `CAN Unpack` block with the **Output remote** check box selected to enable the **Remote port**. When the **Remote port** becomes true, send the current value to the requesting subsystem.
- 

The main purpose of the two blocks is to be used in conjunction with `CAN Send` and `CAN Receive` driver blocks for `double` data type. You can use them for other types of `double` type data manipulation. Their functionality is entirely independent of any CAN driver blocks or CAN library.

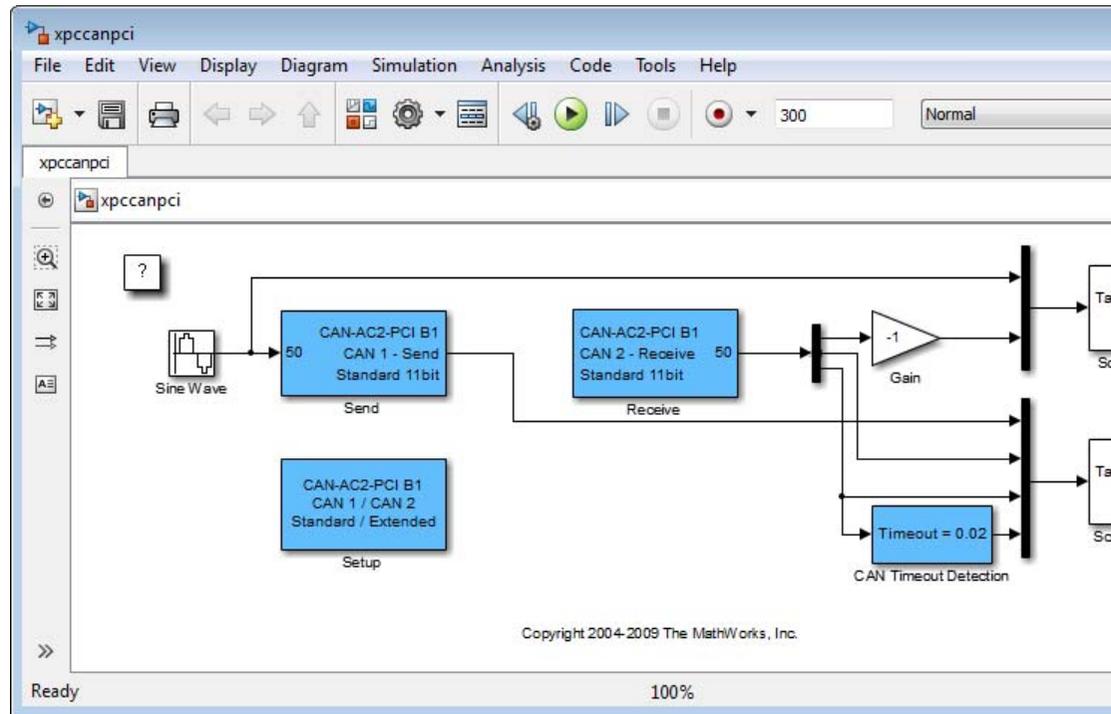
## Timeouts When Receiving CAN Messages

The model in this topic supports the legacy double data type.

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 timeout condition.

A Simulink model using this block in conjunction with a Receive block could look as follows in the `xpccanpci` example:



To change this model to use the CAN\_MESSAGE data type, you can:

- 1 Add a CAN Unpack block to the model.
- 2 Connect the output of the receive block to the input of the CAN Unpack block.
- 3 Double-click the CAN Unpack block.
- 4 Select the **Output timestamp** check box.  
A Timestamp output port appears. Save and close the block dialog box.
- 5 Connect the Timestamp output port to the inport port of the CAN Timeout Detection block.
- 6 Save the model.

See `xpccanpcis`, a loopback example for the CAN-AC2-PCI board using `CAN_MESSAGE` data types for message storage.

## **Blocks – Alphabetical List**



# CAN I/O Support for FIFO

---

- “CAN FIFO Basics” on page 8-2
- “Acceptance Filters” on page 8-47
- “CANdb DBC Format Databases” on page 8-49
- “CAN FIFO Loopback Tests” on page 8-50
- “Blocks — Alphabetical List” on page 8-53

## CAN FIFO Basics

In this section...
“FIFO Mode” on page 8-2
“FIFO Mode Drivers for CAN Boards from Softing” on page 8-5

### FIFO Mode

This topic describes the alternative First In First Out (FIFO) mode CAN drivers provided with the xPC Target blocks. 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. 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. FIFO mode blocks send messages in the order that they are input to the block. They do not take into account the priority of the input.

---

**Note** Each CAN channel has two FIFOs, DPRAM and SRAM. The xPC Target FIFO block writes into the DPRAM FIFO. The Softing board transfers the messages from the DPRAM FIFO to the SRAM FIFO, from which the messages are transmitted onto the CAN bus. The DPRAM can buffer up to 31 messages; the SRAM can buffer up to 127 messages. This implies that while 127 messages can be buffered during congestion on the CAN bus, the 31 message limit of the DPRAM limits how many messages a model can write at one time.

---

- **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 us. 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

$$20 * 40 \text{ us} = 800 \text{ us}$$

With such an application, base sample times below 800 us are impossible even if the dynamics of the corresponding Simulink model are simple and would only need 20 us 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 deal with only a small subset of all CAN messages going over the CAN network.

## Data Types

The xPC Target CAN blocks support the following message data types. If you have Softing ISA CAN boards, you can use only the double data type for messages. If you have Softing PCI and PC/104 CAN boards, you can use either CAN\_MESSAGE or double data types for new messages.

- CAN\_MESSAGE — Structure that contains an array of eight unsigned 8-bit integers that contains the data. This structure also contains the ID, and the standard and extended ID range. CAN blocks can pass data of this type without requiring complicated bit packing and unpacking. This capability enables you to create simpler models. Use messages of this data

type for new CAN application models. Consider updating existing CAN application models to use CAN\_MESSAGE data types.

CAN\_MESSAGE — Use CAN Pack blocks to pack individual signals into a CAN message for sending. Use CAN Unpack blocks to unpack individual signals from CAN messages,

You can construct a CAN\_MESSAGE by using the CAN Pack. You can use this block to pass raw data packed elsewhere, by packing a data pattern manually. You can also use a CANDB file and select a predefined data pattern. If you have several CAN\_MESSAGE messages, use a Mux block to combine them into a vector that you connect to a CAN FIFO send block.

- Double — Double that represents 8 bytes of message in a signal. CAN blocks manipulate data of this type before passing it. Do not use this data type to create new CAN application models. The maximum size of the data frame of a CAN message is 8 bytes. This size is the same as the C data type double uses on PC-compatible systems. At the same time, the double data type is the default data type for Simulink signals. Represent the CAN data frame within a Simulink model with a scalar Simulink signal. You can represent the data frame 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. Use these blocks to pack data types other than doubles into 64 bits (8 bytes or a double) as well as for the opposite operation. Simulink signals of data type double represent CAN data frames.

**Updating Existing Models to Use CAN\_MESSAGE Data Types.** If you have Softing PCI and PC/104 CAN boards, you can update existing models to use CAN\_MESSAGE data types. As a rule:

- If your existing model contains CAN Bit-Packing and CAN Bit-Unpacking blocks, replace them with the CAN Pack and CAN Unpack blocks.
- Open and reconfigure the CAN Send and Receive blocks to use CAN\_MESSAGE data types.
- Remove FIFO Mode CAN Message and CANDBC Translator blocks. Updated models no longer require the conversions that these blocks perform.

See the CAN Blocks Transition document for more information.

## 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 us per CAN message and reading one event (CAN message) from the receive FIFO takes 17 us. Both of these latency times are smaller than the 40 us 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 nondeterministic 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 us regardless of the event read out of the FIFO, the latency time becomes deterministic and is the Read FIFO Depth multiplied by 17 us. 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. You should not use more than one Read Receive FIFO block in a Simulink model, 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 8-47 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

$$12 * 17 \text{ ns} + 8 * 4 \text{ ns} = 236 \text{ ns}$$

This is a considerable smaller value than the 800 ns 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

$$12 * 17 \text{ ns} + 20 * 4 \text{ ns} = 284 \text{ ns}$$

There is no impact on the final result. That is the trade-off. 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 run without an error one of the loopback tests provided with the xPC Target product.

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 different I/O driver blocks.

As mentioned in the standard CAN topic, 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.

### **CAN-AC2-PCI (FIFO) with SJA1000 Controller**

The CAN-AC2-PCI driver blocks support the CAN-AC2-PCI board 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 CAN-AC2-PCI SJA 1000 block group contains the FIFO mode CAN subgroup.

# 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 16 boards for each target system, making up to 32 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 computer, you can use the board number (1...16) 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 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 your model requires a special baud rate, select the value `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 ]

# Softing CAN-AC2-PCI with Philips SJA1000 FIFO Setup

---

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. For information on how to define the acceptance information to filter certain messages, see “Acceptance Filters” on page 8-47.

## **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 your model requires a special baud rate, select the value `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 ]

# Softing CAN-AC2-PCI with Philips SJA1000 FIFO Setup

---

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. For information on how to define the acceptance information to filter certain messages, see “Acceptance Filters” on page 8-47.

## **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 command structure 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 “Initialization and Termination CAN Messages” on page 7-83.

## **Termination**

Define the CAN messages sent during termination of the Setup block. For more information, see the standard CAN driver documentation in “Initialization and Termination CAN Messages” on page 7-83.

## Show bus-off status output

Select this check box to enable the bus-off status output port. The output signal can be 1 (bus-off state); otherwise, it is 0. If the bus-off status persists, the block optionally initiates recovery, depending on the setting of **Bus-off recovery**.

Clear this check box to disable the output port.

## Bus-off recovery

Use this parameter to specify how the model is to perform bus-off recovery for the CAN network.

From the list, select:

- Off

Does not perform bus-off recovery.

- Auto

Upon detection, the model waits 1 second to see if the bus-off state persists. After 1 second, bus-off recovery occurs.

- Manual Trigger Input

Enables you to input a signal when you want bus-off recovery to occur. Selecting this option creates an input port for the Setup block.

## 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. When autosearch is enabled, 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 computer, 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`.

## Softing CAN-AC2-PCI with Philips SJA1000 FIFO Setup

---

The board allows you to terminate each of the two CAN ports separately by means of DIP switches at the rear panel. Both CAN ports must be terminated if you use the loopback model provided to test the board and drivers. Refer to the Softing user manual on how to set the DIP switches.

# Softing CAN-AC2-PCI with Philips SJA1000 FIFO Write

---

**Purpose** Softing CAN-AC2-PCI with Philips SJA1000 FIFO Write Driver block

**Library** xPC Target Library for CAN

**Description** The FIFO Write driver block writes CAN messages into the transmit FIFO. The firmware running in FIFO mode processes the information in the transmit FIFO and finally puts the constructed CAN messages onto the bus. This block dynamically detects the data type and signal size of the signal you connect to the input port. If you use this block inside an enabled subsystem, the block sends the message only when you enable the subsystem.

If you use the CAN\_MESSAGE data type:

If the block input is a CAN\_MESSAGE data type, use CAN Pack blocks to pack individual signals into a CAN message. If you use CAN Pack blocks to prepare each message, the FIFO Write block reads the CAN\_MESSAGE data. To combine several messages into a vector of CAN\_MESSAGE data that the FIFO Write block sends, use a Mux block. Use the **Port (if input is CAN\_MESSAGE)** parameter to select the port to which to send the CAN\_MESSAGE data.

---

**Tip** To prepare a remote frame, use the CAN Pack block with the **Remote frame** check box selected. To request the current value, send the remote frame to the responding subsystem.

---

If you use the double data type (legacy):

If the block input consists of double values in a matrix, the block has one input port of type double. At this port, all required 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:

**Port**

The value can be either 1 (port 1) or 2 (port 2) and defines the port the CAN message is sent from.

# Softing CAN-AC2-PCI with Philips SJA1000 FIFO Write

---

## Identifier

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  $2^{29}-1$ .

## 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.

## Data

Data for the data frame itself. Use the CAN Bit-Packing block to construct the data as double values in a matrix.

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 examples of constructing the matrix signal for the FIFO write block, see “CAN FIFO Loopback Tests” on page 8-50.

For certain applications, you might 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).

# Softing CAN-AC2-PCI with Philips SJA1000 FIFO Write

## 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 input count port

Select this check box to track the number of inputs to the block and add a second input port, N, to the block. If this port value exceeds the size of the input vector of CAN\_MESSAGE messages, the block limits the transfer to the size of the input vector. If this port value is smaller than the CAN\_MESSAGE input vector size, the block transfers data up to the size of this port value. For example, assume an input of 10\*5 matrix. If you then input a 5 to input port N, the block sends only the first five messages. In an n\*6 matrix, the last entry in the row is still the individual message enabled, but the block checks only messages up to n. If the input is a matrix of doubles, the block ignores the value of the **Port** parameter. The block considers the port as part of the data.

If this check box is cleared, the block sends all elements of the input vector.

### Port (if input is CAN\_MESSAGE)

If the message is a CAN\_MESSAGE data type, from the list, select 1 or 2 for the port to send the message through. The block sends a message to a single CAN port. If you want to send messages to different ports in a model, use multiple FIFO Write blocks. This input port is the first one on the block. Its input is a vector or single CAN\_MESSAGE message.

If the message consists of double values in a matrix, changing this parameter does not change the block behavior. You specify the port number in the CAN message. A single FIFO Write block can send messages through both ports.

# Softing CAN-AC2-PCI with Philips SJA1000 FIFO Write

---

## Show status output port

Select this 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 `CAN_MESSAGE` or `double` depending on which data type you are using for your messages, 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. The function return codes are:

Code	Description
0	Function executed without detecting an error.
-1	Function encountered an error.
-4	Timeout firmware communication.
-99	Board not initialized.

## 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 required. 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

**Purpose** Softing CAN-AC2-PCI 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 which the FIFO Read driver reads it.

If you choose CAN\_MESSAGE data type as the output, use CAN Unpack blocks to unpack individual signals from a CAN message.

---

**Tip** To process a remote frame, use the CAN Unpack block with the **Output remote** check box selected to enable the Remote port. When the Remote port becomes true, send the current value to the requesting subsystem.

---

If you choose matrix as the output data type, the FIFO Read driver block has 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 dialog box (see below). For example, if the FIFO read depth is 5, then the matrix signal of port 1 has size  $5 \times 6$ . Therefore, one row for each event is read out of the receive FIFO (no new message is considered an event as well). For examples showing how to extract data from the matrix signal, see “CAN FIFO Loopback Tests” on page 8-50.

If you choose output as double values in a matrix, each row with its six elements contains all the information defining a CAN message:

**Port**

The value can be either 1 (port 1) or 2 (port 2) and reports the port at which the CAN message was received.

# Softing CAN-AC2-PCI 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 it is an extended CAN message, the range is 0 to  $2^{29}-1$ .

## Event type

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

Events	Description
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.

# Softing CAN-AC2-PCI with Philips SJA1000 FIFO Read

## 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 us.

## Data

Data of the data frame itself returned as a CAN\_MESSAGE or double value (8 bytes) depending on which data type you are using for your messages. Use the CAN Unpack block to extract the data from the CAN\_MESSAGE data value. Use the CAN Bit-Unpacking block 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.

### Output format

From the list, select:

- Matrix of doubles (legacy)

Formats messages as double values in a matrix. See “Description” on page 8-17 for details on this message format.

- Vector of CAN\_MESSAGE plus count

Formats messages as vectors of CAN\_MESSAGE structures.

Selecting this option enables the **Port** parameter. It also creates an N output port that contains the number of valid entries in the CAN\_MESSAGE data vector. The block interprets the **FIFO read depth** value as the size of the vector. The block outputs up to **FIFO read depth** number of messages. For example, if you enter a read depth of 3, but the receive FIFO has only two messages, the output vector has only

# Softing CAN-AC2-PCI with Philips SJA1000 FIFO Read

---

two messages. The `xpccanpcififo1s` example illustrates the use of this parameter.

- Function call for each CAN\_MESSAGE

Formats messages as vectors of CAN\_MESSAGE structures.

Selecting this option enables the **Port** parameter. It also creates a function-call output (f()) output port. This port contains a function-call for a single CAN\_MESSAGE data element. The block repeats the function-call for each message from the FIFO, up to the value of **FIFO read depth**. You input this function-call output into a function-call subsystem. The `xpccanpcififo1sf` example illustrates the use of this parameter.

## Port

CAN\_MESSAGE only. From the list, select 1 or 2 for the input port. The block reads the message from that FIFO (1 or 2). Its output is a vector of CAN\_MESSAGE messages. If you want to read from both ports, use two Read blocks, one for each port. Selecting Vector of CAN\_MESSAGE plus count for **Output format** enables this parameter.

Selecting Matrix of doubles (legacy) or Function call for each CAN\_MESSAGE for the **Output format** parameter disables this parameter. The block ignores the port. In this case, the block reads from both ports and records the port for each message in the output matrix.

## 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 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).

## Show status output port

Select this to enable the status output port (S). If the box is cleared (disabled), the block has one output port for the events (D). If enabled, a second port is shown. The signal leaving that port is a vector of doubles, 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 127 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

Code	Description
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.

---

**Note** 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. 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.

## 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:

Keyword	Description
SDF	Standard data frame.
SRF	Standard remote frame.
EDF	Extended data frame.
ERF	Extended remote frame.

# Softing CAN-AC2-PCI with Philips SJA1000 FIFO Read Filter

Keyword	Description
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 required. Usually, you connect several instances in parallel to the output of the FIFO Read driver block to filter out particular messages or events. For examples of doing this, see “CAN FIFO Loopback Tests” on page 8-50.

# Softing CAN-AC2-PCI with Philips SJA1000 FIFO Read XMT Level

---

<b>Purpose</b>	Softing CAN-AC2-PCI with Philips SJA1000 FIFO Read XMT Level block
<b>Library</b>	xPC Target Library for CAN
<b>Description</b>	<p>The FIFO Read XMT Level driver block reads the current number of CAN messages stored in the transmit FIFO to be processed by the firmware. The transmit FIFO can store up to 31 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 action. For example, you could stop the execution or wait for a non-full transmit FIFO.</p> <p>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).</p>
<b>Block Parameters</b>	<p><b>Board</b></p> <p>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.</p> <p><b>Sample time</b></p> <p>Defines the sample time at which the FIFO Read XMT Level driver block is executed during a model (target application) run.</p>

# Softing CAN-AC2-PCI with Philips SJA1000 FIFO Reset XMT

---

<b>Purpose</b>	Softing CAN-AC2-PCI with Philips SJA1000 FIFO Reset XMT block
<b>Library</b>	xPC Target Library for CAN
<b>Description</b>	<p>The FIFO Reset XMT driver block resets 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.</p> <p>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.</p>
<b>Block Parameters</b>	<p><b>Board</b> 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.</p> <p><b>Sample time</b> Defines the sample time at which the FIFO Reset XMT driver block is executed during a model (target application) run.</p>

# 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 127 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 increments to the lost message counter. You can use this driver block to check for this condition and take 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.

### **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

<b>Purpose</b>	Softing CAN-AC2-PCI with Philips SJA1000 FIFO Reset RCV block
<b>Library</b>	xPC Target Library for CAN
<b>Description</b>	<p>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.</p> <p>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.</p>
<b>Block Parameters</b>	<p><b>Board</b> 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.</p> <p><b>Sample time</b> Defines the sample time at which the FIFO Reset RCV driver block is executed during a model (target application) run.</p>

## **CAN-AC2-104 (FIFO) with SJA1000 Controller**

The CAN-AC2-104 driver blocks support the CAN-AC2-104 (PC/104) board 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 CAN-AC2-104 SJA 1000 block group contains the FIFO Mode subgroup.

# 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.

## Block Parameters

### Board

Defines the board accessed by this driver block instance. If multiple boards are present in the target computer, 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.

### CAN 1 - baud rate

Defines the most common baud rates for CAN port 1. If your model requires a special baud rate, select the value `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

# Softing CAN-AC2-104 with Philips SJA1000 FIFO Setup

---

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 8-47.

## **CAN 2 - baud rate**

Defines the most common baud rates for CAN port 2. If your model requires a special baud rate, select the value `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:

[ 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 8-47.

## **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

# Softing CAN-AC2-104 with Philips SJA1000 FIFO Setup

---

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 the standard CAN driver documentation in “Initialization and Termination CAN Messages” on page 7-83.

## **Termination**

Defines CAN messages sent during termination of the Setup block. For more information, see the standard CAN driver documentation in “Initialization and Termination CAN Messages” on page 7-83.

## **Show bus-off status output**

Select this check box to enable the bus-off status output port. The output signal can be 1 (bus-off state); otherwise, it is 0. If the bus-off status persists, the block optionally initiates recovery, depending on the setting of **Bus-off recovery**.

Clear this check box to disable the output port.

## **Bus-off recovery**

Use this parameter to specify how the model is to perform bus-off recovery for the CAN network.

From the list, select:

- Off

Does not perform bus-off recovery.

- Auto

Upon detection, the model waits 1 second to see if the bus-off state persists. After 1 second, bus-off recovery occurs.

- Manual Trigger Input

# Softing CAN-AC2-104 with Philips SJA1000 FIFO Setup

---

Enables you to input a signal when you want bus-off recovery to occur. Selecting this option creates an input port for the Setup block.

## **I/O base address**

Defines the I/O base address of the board to be accessed by this block instance. The I/O base address is given by the DIP switch setting on the board itself. The I/O address range is 3 bytes and is mainly used to identify the memory base address the board should use. See the Softing user manual for this board on how you can set the I/O base address. The I/O base address entered in this control must correspond with the DIP switch setting on the board. If more than one board is present in the target system, you must enter a different I/O base address for each board. In this case, the I/O base address itself defines which board is referenced by which board number.

## **Memory base address**

Defines the memory base address of the board to be accessed by this block instance. The memory base address is a software setting only (no corresponding DIP switch is found on the board). The memory address range is 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 verify 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:

C8000 - D8000

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. Both CAN ports must be terminated before you can use the loopback model provided to test the board and drivers. Refer to the Softing user manual on how to set the DIP switches.

## **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 writes CAN messages into the transmit FIFO. The firmware running in FIFO mode then processes the information in the transmit FIFO and finally puts the constructed CAN messages onto the bus. This block dynamically detects the data type and signal size of the signal that you connect to the input port.

If you use the CAN\_MESSAGE data type:

If the block input is a CAN\_MESSAGE data type, use CAN Pack blocks to pack individual signals into a CAN message. If you use CAN Pack blocks to prepare each message, the FIFO Write block reads the CAN\_MESSAGE data. To combine several messages into a vector of CAN\_MESSAGE data that the FIFO Write block sends, use a Mux block. Use the **Port (if input is CAN\_MESSAGE)** parameter to select the port to which to send the CAN\_MESSAGE data.

---

**Tip** To prepare a remote frame, use the CAN Pack block with the **Remote frame** check box selected. To request the current value, send the remote frame to the responding subsystem.

---

If you use the double data type (legacy):

If the block input consists of double values in a matrix, the block has one input port of type double. At this port, you must provide all elements required to construct valid CAN messages to be written into the transmit FIFO. For each CAN message, five elements have to be passed:

Port

The value can be either 1 (port 1) or 2 (port 2) and defines at which port the CAN message is sent from.

# Softing CAN-AC2-104 with Philips SJA1000 FIFO Write

---

## 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

## Data

Data for the data frame itself. Use the CAN Bit-Packing block to construct the data as double values in a matrix.

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 examples of constructing the matrix signal for the FIFO write block, see “CAN FIFO Loopback Tests” on page 8-50.

For certain applications, you might 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).

# Softing CAN-AC2-104 with Philips SJA1000 FIFO Write

## 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 input count port

Select this check box to track the number of inputs to the block and add a second input port, N, to the block. If this port value exceeds the size of the input vector of CAN\_MESSAGE messages, the block limits the transfer to the size of the input vector. If this port value is smaller than the CAN\_MESSAGE input vector size, the block transfers data up to the size of this port value. For example, assume an input of 10\*5 matrix. If you then input a 5 to input port N, the block sends only the first five messages. In an n\*6 matrix, the last entry in the row is still the individual message enabled, but the block checks only messages up to n. If the input is a matrix of doubles, the block ignores the value of the **Port** parameter. The block considers the port as part of the data.

If this check box is cleared, the block sends all elements of the input vector.

### Port (if input is CAN\_MESSAGE)

If the message is a CAN\_MESSAGE data type, from the list, select 1 or 2 for the port to send the message through. The block sends a message to a single CAN port. If you want to send messages to different ports in a model, use multiple FIFO Write blocks. This input port is the first one on the block. Its input is a vector or single CAN\_MESSAGE message.

If the message consists of double values in a matrix, changing this parameter does not change block behavior. You specify the port number in the CAN message. A single FIFO Write block can send messages through both ports.

# Softing CAN-AC2-104 with Philips SJA1000 FIFO Write

## 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 `CAN_MESSAGE` or `double` depending on which data type you are using for your messages. 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. The function return codes are:

Code	Description
0	Function executed without detecting an error.
-1	Function encountered an error.
-4	Timeout firmware communication.
-99	Board not initialized.

## 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 required. 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 127 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 increments to the lost message counter. You can use the FIFO Read RCV Level driver block to check for this condition and take action (for example, by stopping the execution or resetting the receive FIFO).

If you choose CAN\_MESSAGE data type as the output, use CAN Unpack blocks to unpack individual signals from a CAN message.

---

**Tip** To process a remote frame, use the CAN Unpack block with the **Output remote** check box selected to enable the Remote port. When the Remote port becomes true, send the current value to the requesting subsystem.

---

If you choose matrix as the output data type, the FIFO Read driver block has 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 examples of extracting data from the matrix signal, see "CAN FIFO Loopback Tests" on page 8-50.

If you choose output as double values in a matrix, each row with its six elements contains all the information defining a CAN message:

# Softing CAN-AC2-104 with Philips SJA1000 FIFO Read

## Port

The value is either 1 (port 1) or 2 (port 2) and reports the port at which the CAN message was received.

## Identifier

Identifier of the CAN message being received. If it is a standard CAN message, the range is 0 to 2047. If the CAN message is extended, the range is 0 to  $2^{29}-1$ .

## Event type

Defines the type of event read from the receive FIFO. The following values are defined from the Softing user manual:

Events	Description
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	

# Softing CAN-AC2-104 with Philips SJA1000 FIFO Read

Events	Description
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 $\mu$ s.

## Data

Data of the data frame itself. It is returned as a `CAN_MESSAGE` or double value (8 bytes) depending on which data type you are using for your messages. Use the `CAN Pack` to extract data from the `CAN_MESSAGE` data. Use the `CAN Bit-Unpacking` block to extract the data from the double value.

## 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.

### Output format

From the list, select:

- `Matrix of doubles (legacy)`  
Formats messages as double values in a matrix. See “Description” on page 8-17 for details on this message format.
- `Vector of CAN_MESSAGE plus count`  
Formats messages as vectors of `CAN_MESSAGE` structures.  
Selecting this option enables the **Port** parameter. It also creates an `N` output port that contains the number of valid entries in the `CAN_MESSAGE` data vector. The block

interprets the **FIFO read depth** value as the size of the vector. The block outputs up to **FIFO read depth** number of messages. For example, if you enter a read depth of 3, but the receive FIFO has only two messages, the output vector has only two messages. The `xpccanpcfifo1s` example illustrates the use of this parameter.

- Function call for each CAN\_MESSAGE

Formats messages as vectors of CAN\_MESSAGE structures.

Selecting this option enables the **Port** parameter. It also creates a function-call output (f()) output port. This port contains a function-call for a single CAN\_MESSAGE data element. The block repeats the function-call for each message from the FIFO, up to the value of **FIFO read depth**. You input this function-call output into a function-call subsystem. The `xpccanpcfifo1sf` example illustrates the use of this parameter.

## Port

CAN\_MESSAGE only. From the list, select 1 or 2 for the input port. The block reads the message from that FIFO (1 or 2). Its output is a vector of CAN\_MESSAGE messages. If you want to read from both ports, use two Read blocks, one for each port. Selecting Vector of CAN\_MESSAGE plus count for **Output format** enables this parameter.

Selecting Matrix of doubles (legacy) or Function call for each CAN\_MESSAGE for the **Output format** parameter disables this parameter. The block ignores the port. In this case, the block reads from both ports and records the port for each message in the output matrix.

## 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

# Softing CAN-AC2-104 with Philips SJA1000 FIFO Read

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 (S). 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 doubles, 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 127 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

Code	Description
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 the same type as the input port. 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.

**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:

Keyword	Description
SDF	Standard data frame.
SRF	Standard remote frame.

# Softing CAN-AC2-104 with Philips SJA1000 FIFO Read Filter

---

Keyword	Description
EDF	Extended data frame.
ERF	Extended remote frame.
EF	Error frame.
NE	No new event.
CBS	Change of bus state.

## Message type selection mode

Defines how the event type (message type) listed in **Message type command** is treated. If you select **Include**, the event type criterion is the sum of the concatenated keywords. If you select **Exclude**, the event type criterion is equal to all event types minus the sum of the concatenated keywords.

## Identifier(s)

Defines the filter criterion for the CAN message identifiers. A set of identifiers can be provided as a row vector.

## Identifier selection mode

Defines how the identifier criterion entered in the **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 required. Usually, you connect several instances in parallel to the output of the FIFO Read driver block to filter out particular messages or events. For examples of doing this, see “CAN FIFO Loopback Tests” on page 8-50.

# Softing CAN-AC2-104 with Philips SJA1000 FIFO Read XMT Level

---

<b>Purpose</b>	Softing CAN-AC2-104 with Philips SJA1000 FIFO Read XMT Level block
<b>Library</b>	xPC Target Library for CAN
<b>Description</b>	<p>The FIFO Read XMT Level driver block reads the current number of CAN messages stored in the transmit FIFO to be processed by the firmware. The transmit FIFO can store up to 31 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 action. For example, you can stop the execution or wait for the transmit FIFO to empty.</p> <p>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).</p>
<b>Block Parameters</b>	<p><b>Board</b> 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.</p> <p><b>Sample time</b> Defines the sample time at which the FIFO Read XMT Level driver block is executed during a model (target application) run.</p>

# 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

---

<b>Purpose</b>	Softing CAN-AC2-104 with Philips SJA1000 FIFO Read RCV Level block
<b>Library</b>	xPC Target Library for CAN
<b>Description</b>	<p>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 127 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 increments to the lost message counter. You can use this driver block to check for this condition and take action (for example, like stopping the execution or resetting the receive FIFO).</p> <p>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).</p>
<b>Block Parameters</b>	<p><b>Board</b> 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.</p> <p><b>Sample time</b> Defines the sample time at which the FIFO Read RCV Level driver block is executed during a model (target application) run.</p>

# Softing CAN-AC2-104 with Philips SJA1000 FIFO Reset RCV

---

<b>Purpose</b>	Softing CAN-AC2-104 with Philips SJA1000 FIFO Reset RCV block
<b>Library</b>	xPC Target Library for CAN
<b>Description</b>	<p>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.</p> <p>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.</p>
<b>Block Parameters</b>	<p><b>Board</b></p> <p>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.</p> <p><b>Sample time</b></p> <p>Defines the sample time at which the FIFO Reset RCV driver block is executed during a model (target application) run.</p>

## Acceptance Filters

You can use the CAN controller acceptance filters to select received messages by identifier for writing into the receive FIFO. Therefore, fewer read attempts are required to get the messages of importance to 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 both be in the range 0 to 2047. For extended identifiers the mask parameter and code parameter must both be 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.

The default values for the mask parameter and the code parameter in the FIFO setup driver block are both 0. Because the filtering expression always evaluates to 0, all incoming messages 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.

For example, 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 identifiers 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.

## CANdb DBC Format Databases

You can use CANdb DBC files to specify the packing and unpacking of CANdb messages in a target application. To access a CANdb DBC format file,

- 1 Drag a CAN Pack block into your model,
- 2 Configure the **Data is input as** parameter to CANdb specified signals
- 3 Configure the **CANdb file** parameter to reference your CANdb DBC format file

You use the CAN Unpack to unpack the signals from a CAN message.

The xPC Target product contains the following example that illustrate how to read CANdb messages using the CAN\_MESSAGE data type.

<b>CAN-AC2-PCI Example</b>	<b>CAN-AC2-104 Example</b>	<b>Shows CAN I/O Communication Using FIFO Mode of the...</b>
CAN I/O FIFO Mode - Loading Messages From CANdb DBC Files (with CAN_MESSAGE) — CAN_MESSAGE data type	CAN I/O FIFO Mode - Loading Messages From CANdb DBC Files (with CAN_MESSAGE) — CAN_MESSAGE data type	Softing CAN-AC2-PCI and CAN-AC2-104 boards and illustrates basic functionality of the boards. These examples load messages and signal definitions from CANdb DBC database file.

## CAN FIFO Loopback Tests

The xPC Target product contains the following examples that illustrate the use of the Softing CAN FIFO blocks. Some models use `CAN_MESSAGE` data types, some use double data types. Models that use double data types are considered legacy examples. Each example requires a loopback cable connected between the two ports on the board. The examples send messages on port 1 and receive messages on port 2.

These models have the following naming conventions:

- Model names appended with **s** — Output is a vector of `CAN_MESSAGE` data with length equal to the value of the **FIFO read depth** parameter. This length is also the maximum number of messages that the block returns. The second output port returns the number of messages that are present in the output.
- Model names appended with **sf** — Output contains one `CAN_MESSAGE` message at a time. The block calls the function-call output between 0 and **FIFO read depth** number of times, depending on how many messages are in the FIFO.

CAN-AC2-PCI Example	CAN-AC2-104 Example	Shows CAN I/O Communication Using FIFO Mode of the...
CAN I/O FIFO Mode - Simple Use Case — <code>CAN_MESSAGE</code> data type  CAN I/O FIFO Mode — Simple Use Case — Double data type	CAN I/O FIFO Mode - Simple Use Case — <code>CAN_MESSAGE</code> data type  CAN I/O FIFO Mode - Simple Use Case — Double data type	Softing CAN-AC2-PCI and CAN-AC2-104 boards and illustrates basic functionality of the boards.
CAN I/O FIFO Mode - Simple Use Case (with <code>CAN_MESSAGE</code> and Function Call Output) — <code>CAN_MESSAGE</code> data type	CAN I/O FIFO Mode - Simple Use Case (with <code>CAN_MESSAGE</code> and Function-Call Output) — <code>CAN_MESSAGE</code> data type	Softing CAN-AC2-PCI board and illustrates CAN I/O communication using the First In First Out (FIFO) mode. The FIFO Read block uses a function-call output format.

<b>CAN-AC2-PCI Example</b>	<b>CAN-AC2-104 Example</b>	<b>Shows CAN I/O Communication Using FIFO Mode of the...</b>
CAN I/O FIFO Mode - The "No New Event" Message — Double data type	CAN I/O FIFO Mode - The "No New Event" Message — Double data type	Softing CAN-AC2-PCI and CAN-AC2-104 boards and illustrates the condition and detection of the No new event message.
CAN I/O FIFO Mode - Dynamic Message Generation — CAN_MESSAGE data type  CAN I/O FIFO Mode - Dynamic Message Generation — Double data type	CAN I/O FIFO Mode - Dynamic Message Generation — CAN_MESSAGE data type  CAN I/O FIFO Mode - Dynamic Message Generation — Double data type	Softing CAN-AC2-PCI and CAN-AC2-104 boards and illustrates how CAN messages can be dynamically constructed at run-time.
CAN I/O FIFO Mode - Stopping on FIFO Overflow — CAN_MESSAGE data type  CAN I/O FIFO Mode - Stopping on FIFO Overflow — Double data type	CAN I/O FIFO Mode - Stopping on FIFO Overflow — CAN_MESSAGE data type  CAN I/O FIFO Mode - Stopping on FIFO Overflow — Double data type	Softing CAN-AC2-PCI and CAN-AC2-104 boards. For the case where data is being received faster than it is being processed, these examples illustrate how to stop execution of the model after detecting a receive FIFO overflow.
CAN I/O FIFO Mode - Resetting before FIFO Overflow — CAN_MESSAGE data type  CAN I/O FIFO Mode - Resetting before FIFO Overflow — Double data type	CAN I/O FIFO Mode - Resetting before FIFO Overflow — CAN_MESSAGE data type  CAN I/O FIFO Mode - Resetting before FIFO Overflow — Double data type	Softing CAN-AC2-PCI and CAN-AC2-104 boards. For the case where data is being received faster than it is being processed, these examples illustrate how to reset the receive FIFO before it overflows.

<b>CAN-AC2-PCI Example</b>	<b>CAN-AC2-104 Example</b>	<b>Shows CAN I/O Communication Using FIFO Mode of the...</b>
<p>CAN I/O FIFO Mode - Acceptance Filters — CAN_MESSAGE data type</p> <p>CAN I/O FIFO Mode - Acceptance Filters — Double data type</p>	<p>CAN I/O FIFO Mode - Acceptance Filters — CAN_MESSAGE data type</p> <p>CAN I/O FIFO Mode - Acceptance Filters — Double data type</p>	<p>Softing CAN-AC2-PCI and CAN-AC2-104 boards and illustrates how to use acceptance filters. The objective is to filter any CAN messages with an identifier larger than 127.</p>
<p>CAN I/O FIFO Mode - Loading Messages from CANdb DBC Files — CAN_MESSAGE data type</p>	<p>CAN I/O FIFO Mode - Loading Messages from CANdb DBC Files — CAN_MESSAGE data type</p>	<p>Softing CAN-AC2-PCI and CAN-AC2-104 boards. The example loads message and signal definitions from a CANdb DBC database file. This example assumes that the CANdb file, demoData.dbc, resides in the xpcdemos folder.</p>

## **Blocks – Alphabetical List**



# Model-Based Ethernet Communications Support

---

- “Model-Based Ethernet Communications” on page 9-2
- “Using xPC Target for Ethernet Communications” on page 9-6
- “Blocks — Alphabetical List” on page 9-7

## Model-Based Ethernet Communications

### In this section...

“What Is Model-Based Ethernet Communications?” on page 9-2

“PCI Bus and Slot Numbers” on page 9-3

“MAC Addresses” on page 9-3

“Network Buffer Pointers” on page 9-4

“Filter Type and Filter Address Blocks” on page 9-4

“Execution Priority” on page 9-4

“xPC Target Ethernet Block Library” on page 9-4

### What Is Model-Based Ethernet Communications?

The xPC Target software supports communication from the target computer to other systems or devices using raw Ethernet (Ethernet packets). Raw Ethernet is a direct method to send and receive packets with the target application using the Ethernet protocol. To transfer data using Ethernet packets, you must manually create Ethernet frames. This topic assumes that you are knowledgeable about the Ethernet standard 802.3 standard.

Before you start, provide a dedicated Ethernet card on your target computer. A dedicated Ethernet card is to be used only for model-based Ethernet communications and not for communication between the host computer and target computer. This requirement means that your target computer must have at least two Ethernet cards, one for host computer to target computer communications, and one for model-based Ethernet communication. The xPC Target model-based Ethernet communication blocks support selected members of the following Intel (Vendor ID 0x8086) chip families:

- Intel 8255X
- Intel Gigabit

## PCI Bus and Slot Numbers

To use the model-based Ethernet blocks, specify the PCI bus and slot number of the dedicated Ethernet card in the Ethernet Init block. To identify which Ethernet card is available:

- 1 Boot the target computer with which you want to perform model-based Ethernet communications.
- 2 Examine the startup screen on the target computer. Note the PCI bus and slot information on the bottom right of the status window. This represents the Ethernet card installed on the target computer for dedicated communication between the host computer and target computer.

- 3 In the MATLAB Command Window, type

```
getxpcpci('all')
```

This command determines which PCI boards are installed in the target computer.

- 4 Examine the list of PCI boards and look for the Ethernet cards.
- 5 In the list, find the Ethernet card with a bus and slot different from that displayed on the target computer monitor.
- 6 Note the PCI bus and slot of the free Ethernet card. This is the card you can use for model-based Ethernet communications.

## MAC Addresses

A number of the Ethernet blocks require you to enter MAC addresses. The MAC address must be vector-based. To obtain the vector-based version of a MAC address, use the `macaddr` command. This command converts a string-based MAC address to a vector-based one. For example:

```
macaddr('01:23:45:67:89:ab')
```

```
[1 35 69 103 137 171]
```

When an Ethernet block requires a MAC address, you can enter either of the following in the address field:

- Command `macaddr('MAC address string')`, for example:  
`macaddr('01:23:45:67:89:ab')`
- Vector-based output from the `macaddr` command, for example:  
`[1 35 69 103 137 171]`

### Network Buffer Pointers

The xPC Target Ethernet block library uses pointers to refer to network buffers. Blocks can pass pointers to these buffers as single `uint32` pointers. They can also refer to a chain of network buffer packets.

### Filter Type and Filter Address Blocks

The Filter Type and Filter Address blocks accept a chain of network buffers as input. Based on the criteria specified in the masks of these blocks, the drivers parse each buffer on the chain and either pass the packets through the port or drop the packets. When using these blocks, create your models with filter blocks to pass data only from expected sources.

### Execution Priority

The raw Ethernet blocks have the following execution priority, from first to last:

- 1 Buffer Mngmt
- 2 Ethernet Init
- 3 All other raw Ethernet and network buffer library blocks

### xPC Target Ethernet Block Library

To access the xPC Target Ethernet library blocks, in the xPC Target block library, double-click Ethernet. The xPC Target Ethernet library is displayed.

The xPC Target Ethernet library contains commonly used Ethernet blocks at the top level of the library. Use these blocks to create your models.

The Ethernet library also has a sublibrary, Network Buffers, which contains blocks specific to the management of Ethernet network buffers. The blocks in this sublibrary are core blocks that you might want to use to create other subsystems. However, the top-level Ethernet blocks provide enough functionality for model-based Ethernet communications.

## Using xPC Target for Ethernet Communications

The xPC Target examples directory contains the following examples that illustrate the use of model-based Ethernet communications.

Example	Description
Real-Time Transmit and Receive over Ethernet	Shows how to perform real-time communications over the Ethernet protocol.
Filtering on MAC Address	Shows how to filter Ethernet packets based on MAC addresses.
Filtering on EtherType	Shows how to filter Ethernet packets based on EtherTypes.

## **Blocks – Alphabetical List**

- Buffer Mngmt
- Create Ethernet Packet
- Ethernet Init
- Ethernet Rx
- Ethernet Tx
- Extract Ethernet Packet
- Filter Address
- Filter Type
- Header Extract

# Buffer Mngmt

---

<b>Purpose</b>	Buffer Mngmt block
<b>Library</b>	xPC Target Library for Ethernet
<b>Description</b>	Use the Buffer Mngmt block to initialize network buffers.
<b>Block Parameters</b>	<p>This block has two tabs, <b>Main</b> and <b>Advanced</b>.</p> <p><b>Main</b> tab</p> <p><b>Sample time</b> Base sample time or a multiple of the base sample time.</p> <p><b>Advanced</b> tab. You do not normally need to modify the values of the parameters in this tab. Changing the values of these parameters can affect the performance of your system.</p> <p><b>Buffer pool sizes (256, 512, 1024, 2048)</b> Enter a vector of the number of buffers for each pool size (256, 512, 1024, 2048).</p> <p><b>Display tuning information</b> Select this check box to enable a display of statistical data collected during the run of the model.</p>

<b>Purpose</b>	Create Ethernet Packet block
<b>Library</b>	xPC Target Library for Ethernet
<b>Description</b>	Use the Create Ethernet Packet block to create the Ethernet packets that you want to transfer.
<b>Block Parameters</b>	<p><b>Destination MAC</b> Enter the MAC address of the target computer to receive the data.</p> <p><b>EtherType (use 0 for length)</b> Enter a value to represent either the EtherType or the length that the Ethernet frame is to transfer.</p> <ul style="list-style-type: none"><li>• EtherType  If you are working with Ethernet packets that use EtherType values, enter an EtherType value to specify which prototype the Ethernet frame will transfer. This value must be a valid EtherType.</li><li>• Ethernet length  If you are working with Ethernet packets that use Ethernet lengths, enter 0.</li></ul>

# Ethernet Init

---

<b>Purpose</b>	Ethernet Init block
<b>Library</b>	xPC Target Library for Ethernet
<b>Description</b>	<p>Use the Ethernet Init block to initialize the Ethernet communication channel.</p> <p>You must use an Ethernet Init block for each Ethernet board.</p>
<b>Block Parameters</b>	<p>This block has three tabs, <b>Device</b>, <b>Address</b>, and <b>Advanced</b>.</p> <p><b>Device tab</b></p> <p><b>Device ID</b> From the list, select a unique number to identify the board across multiple blocks. The Ethernet Tx and Ethernet Rx blocks associate this ID with the Ethernet board.</p> <p><b>Driver</b> Identifies the driver for each chip family supported. Possible values are Intel 8255X and Intel Gigabit.</p> <p><b>PCI bus</b> Enter the PCI bus number of the Ethernet card that you will use for model-based communications.</p> <p><b>PCI slot</b> Enter the PCI slot number of the Ethernet card that you will use for model-based communications.</p> <p><b>Addressing tab</b></p> <p><b>Address Source</b> From the list, select:</p> <ul style="list-style-type: none"><li>• EEPROM</li></ul> <p>Allow the block to get the Ethernet card MAC address that is built into the Ethernet card.</p>

- Specify

Enable the **MAC** parameter so that you can manually enter a MAC address for the Ethernet card.

## **MAC**

Enter the MAC address for the Ethernet card. This parameter is enabled if you select **Specify** for the **Address source** parameter.

## **Rx Promiscuous**

Select this check box to direct the model to receive all packets regardless of their destination address.

## **Multicast address list**

Enter a list of multicast address vectors as a cell array. By default, this parameter contains an empty cell array. The Ethernet Rx block uses these addresses in addition to the broadcast address and unicast address.

**Advanced** tab. You do not normally need to modify the values of the parameters in this tab. Changing the values of these parameters can affect the performance of your system.

## **Rx Bad Frames**

Select this check box to direct the model to receive all packets, including erroneous ones (such as CRC error, alignment error, and so forth).

## **Rx Short Frames**

Select this check box to direct the model to receive all packets, including frames that are less than 64 bytes in length.

---

**Note** The Intel Gigabit Ethernet controller does not distinguish between bad packets and short packets. Therefore, checking either **Rx Bad Frames** or **Rx Short Frames** has the same effect for **Driver** type Intel Gigabit.

---

# Ethernet Init

---

## Max MTU

Specify a maximum transmission unit number. This parameter allows you to specify a smaller maximum transmission unit number. The default is 1518.

## Tx Threshold

Enter a value to control when the Ethernet device will begin to perform direct memory access (DMA) on packets from memory. The default is 224.

---

### Note

- This parameter only applies for **Driver** type Intel 8255X.
  - Before you change this parameter, see the *Intel 8255x 10/100 Mbps Ethernet Controller Family — Open Source Software Developer Manual*.
- 

## Tx Buffers

Enter the maximum number of buffers to hold in the queue before the driver is to drop new transmit requests. The default is 128.

## Rx Buffers

Enter the maximum number of buffers to hold in the queue before the driver is to drop new receive packets. The default is 64.

---

**Note** For both **Tx Buffers** and **Rx Buffers**, the number of buffers must be a multiple of 8.

---

## Display tuning information

Select this check box to enable a display of statistical data collected during the run of the model.

<b>Purpose</b>	Ethernet Rx block
<b>Library</b>	xPC Target Library for Ethernet
<b>Description</b>	Use the Ethernet Rx block to receive Ethernet packets and to filter on the received packets. You can filter packets by EtherType or length. You can use multiple Ethernet Rx blocks with the same device ID. However, you must configure each block to filter a unique set of packets.
<b>Block Parameters</b>	<p><b>Rx tab</b></p> <p><b>Device ID</b> From the list, select a unique number (from 1 to 8) to identify the board across multiple blocks. This is the same ID as was specified in the Ethernet Init <b>Device ID</b> parameter. The Ethernet Rx block associates this ID with the Ethernet board.</p> <p><b>Sample time</b> Base sample time or a multiple of the base sample time.</p> <p><b>Filter tab</b></p> <p><b>Filter criteria</b> From the list, select how you want to filter on EtherTypes (Ethernet II framing standard) or Ethernet lengths (IEEE 802.3 framing standard).</p> <ul style="list-style-type: none"><li>• Receive all unmatched types [0 to 65535] Select this option to output all unmatched packets, both Ethernet II framing and IEEE 802.3 framing standards.</li><li>• Receive unmatched lengths [0 to 1500] Select this option to output all packets with IEEE 802.3 framing standard.</li><li>• Receive unmatched EtherTypes [150 to 65535]</li></ul>

Select this option to output all output packets with Ethernet II framing standard.

- **Specify types to match**

Select this option to enter your own criteria of the EtherTypes to output. This option enables the **Receive these types (vector of types 0–65535)** parameter.

**Receive these types (vector of types 0–65535)**

Enter a vector of EtherTypes that you want to enable to output, for example `[hex2dec('0000')]`.

This field appears only if you select **Specify types to match** for the **Filter criteria** parameter.

<b>Purpose</b>	Ethernet Tx block
<b>Library</b>	xPC Target Library for Ethernet
<b>Description</b>	Use the Ethernet Tx block to send network packets.
<b>Block Parameters</b>	<b>ID</b> From the list, select a unique number (from 1 to 8) to identify the board across multiple blocks. This is the same ID as was specified in the Ethernet Init <b>ID</b> parameter. The Ethernet Tx block associates this ID with the Ethernet board.
	<b>Sample time</b> Base sample time or a multiple of the base sample time.

# Extract Ethernet Packet

---

<b>Purpose</b>	Extract Ethernet Packet block
<b>Library</b>	xPC Target Library for Ethernet
<b>Description</b>	Use the Extract Ethernet Packet block to extract data from an Ethernet packet.
<b>Block Parameters</b>	<b>Data Size</b> Enter the data size (in bytes) for the data you want to extract from an Ethernet packet.

<b>Purpose</b>	Filter Address block
<b>Library</b>	xPC Target Library for Ethernet
<b>Description</b>	Use the Filter Address block to filter network buffer packets, using their MAC addresses as the sort criteria. See “Filter Type and Filter Address Blocks” on page 9-4 for cautions on setting the parameters for this block.
<b>Block Parameters</b>	<p><b>MAC Address</b> Enter a cell array of the MAC address for the filter.</p> <p><b>Drop non-matches</b> Select this check box to discard packets whose MAC addresses do not match any of those entered in the parameter <b>MAC Address</b>.  Clear this check box to direct the block to output packets whose MAC addresses do not match any of those entered in the <b>MAC Address</b> parameter.</p> <p><b>Filter on destination address</b> Clear this check box (default) to direct the block to filter addresses for the source address.  Select this check box to direct the block to filter addresses for the destination address.</p>

# Filter Type

---

## **Purpose**

Filter Type block

## **Library**

xPC Target Library for Ethernet

## **Description**

Use the Filter Type block to filter network buffer packets, using their EtherType values as the sort criteria. See “Filter Type and Filter Address Blocks” on page 9-4 for cautions on setting the parameters for this block.

## **Block Parameters**

### **Match Length (1-1500)**

Select this check box to match packets whose EtherType values fall within the range 1 to 1500.

### **EtherType**

Enter a space delimited list of EtherTypes that you want to filter upon.

### **Drop non-matches**

Clear this check box (default) to direct the block to output packets whose EtherTypes or Ethernet lengths do not match any of those entered in the parameter **EtherType**.

Select this check box to direct the block to discard packets whose EtherTypes or Ethernet lengths do not match any of those entered in the parameter **EtherType**.

<b>Purpose</b>	Header Extract block
<b>Library</b>	xPC Target Library for Ethernet
<b>Description</b>	Use the Header Extract block to extract the header data of network buffer packets.

# Header Extract

---

# Network Buffer Library for Model-Based Ethernet Communications Support

---

- “Network Buffer Blocks” on page 10-2
- “Blocks — Alphabetical List” on page 10-3

### **Network Buffer Blocks**

The Ethernet library has a sublibrary, Network Buffers, which contains blocks specific to the management of Ethernet network buffers. The blocks in this sublibrary are core blocks that you might want to use to create other subsystems. However, the top-level Ethernet blocks provide enough functionality for model-based Ethernet communications.

## **Blocks – Alphabetical List**

Buffer  
Buffer Mngmt  
Chain Size  
Compose  
Extract  
Link  
Merge  
Split  
Unlink

# Buffer

---

**Purpose** Network Buffer block

**Library** xPC Target Library for Ethernet

**Description** Use the Buffer block to queue up buffers.

**Block Parameters** **Chain size**  
Specify the queuing (output) behavior of the block as packets are received.

Value	Description
inf	Pass through all packets. No queuing occurs.
0	No packets are passed through.
Positive number, $N$	Pass through the first $N$ packets.
Negative number, $N$	Pass through the last $N$ packets.

**Buffer size**  
Specify the buffering behavior of the block as packets are received.

Value	Description
inf	Buffer all remaining packets. No packets are deleted.
0	Do not buffer any packets. All remaining packets are deleted.
Positive number, $N$	Buffer the remaining first $N$ packets.
Negative number, $N$	Buffer the remaining last $N$ packets.

## Threshold

Enter a minimum threshold before which this block begins to output buffers.

Value	Description
0	Specifies no threshold.
Negative number, less than 0	Delay passing buffer packets until the threshold value is met once.
Positive number, greater than 0	Passes buffer packets if <i>Threshold</i> number of buffer packets are buffered.

# Buffer Mngmt

---

<b>Purpose</b>	Network Buffer Mngmt block
<b>Library</b>	xPC Target Library for Ethernet
<b>Description</b>	Use the Buffer Mngmt block to initialize network buffers.
<b>Block Parameters</b>	<p>This block has two tabs, <b>Main</b> and <b>Advanced</b>.</p> <p><b>Main</b> tab</p> <p><b>Sample time</b> Base sample time or a multiple of the base sample time.</p> <p><b>Advanced</b> tab. You do not normally need to modify the values of the parameters in this tab. Changing the values of these parameters can affect the performance of your system.</p> <p><b>Buffer pool sizes (256, 512, 1024, 2048)</b> Enter a vector of the number of buffers for each pool size (256, 512, 1024, 2048).</p> <p><b>Display tuning information</b> Select this check box to enable a display of statistical data collected during the run of the model.</p>

<b>Purpose</b>	Network Buffer Chain Size block
<b>Library</b>	xPC Target Library for Ethernet
<b>Description</b>	Use the Chain Size block to determine the number of buffers that are on the chain.

# Compose

---

**Purpose** Network Buffer Compose block

**Library** xPC Target Library for Ethernet

**Description** Use the Compose block to create a network buffer. This block creates a pointer to a network buffer.

<b>Purpose</b>	Network Buffer Extract block
<b>Library</b>	xPC Target Library for Ethernet
<b>Description</b>	Use the Extract block to extract network buffer packets.
<b>Block Parameters</b>	<b>Packet size (-1: inherit)</b> Enter the packet size for the network buffer packet to extract. Enter -1 (default) to inherit the packet size.

# Link

---

**Purpose**

Network Buffer Link block

**Library**

xPC Target Library for Ethernet

**Description**

Use the Link block to convert a vector of network buffer signals into a linked list of signals.

<b>Purpose</b>	Network Buffer Merge block
<b>Library</b>	xPC Target Library for Ethernet
<b>Description</b>	Use the Merge block to combine signal pointers to a linked list.
<b>Block Parameters</b>	<b>Number of inputs</b> Enter the number of network buffer signal pointers to combine into a linked list.

# Split

---

<b>Purpose</b>	Network Buffer Split block
<b>Library</b>	xPC Target Library for Ethernet
<b>Description</b>	Use the Split block to separate a linked list of buffer pointers into separate individual pointers.
<b>Block Parameters</b>	<p><b>Number of outputs</b> Enter the number of pointers the input linked list should be separated into.</p> <ul style="list-style-type: none"><li>• If the number of buffers is the same as this value, this block splits them and outputs them in the order they appear in the vector, or in reverse order (depending on the setting of the <b>Split in reverse order</b> parameter).</li><li>• If the number of buffers is less than <b>Number of outputs</b>, the block outputs zeros (0s) for the extra output ports.</li><li>• If the number of buffers is greater than <b>Number of outputs</b>, the block either deletes the extra buffers, or chains the remaining buffers together (depending on the setting of the <b>Allow chaining for last signal</b> parameter).</li></ul> <p><b>Split in reverse order</b> Select this check box to split out the network buffers in the reverse order in which they are received.</p> <p><b>Allow chaining for last signal</b> Select this check box to chain together remaining network buffers. There might be remaining buffers if the incoming linked list contains more buffers than the number in <b>Number of outputs</b>.</p> <p>Clear this check box to delete the remaining buffers.</p>

**Purpose** Network Buffer Unlink block

**Library** xPC Target Library for Ethernet

**Description** Use the Unlink block to convert a linked list of signals into a vector of network buffer signal.

**Block Parameters** **Vector length (-1: inherit)**  
Enter the number of signals in the linked list of signals that you want to separate. Enter -1 (default) to inherit the vector length.

# Unlink

---

# Model-Based EtherCAT Communications Support

---

- “Model-Based EtherCAT Communications” on page 11-2
- “Blocks — Alphabetical List” on page 11-11

## Model-Based EtherCAT Communications

### EtherCAT Protocol

The Ethernet for Control Automation (EtherCAT®) is an open, high-performance Ethernet network protocol for real-time distributed control. You can use an EtherCAT network instead of an Ethernet network because the EtherCAT protocol provides:

- Deterministic and fast cycle times
- Inexpensive hardware cost

EtherCAT networks consist of one master node and several slave nodes. The xPC Target EtherCAT sublibrary supports only the master node of an EtherCAT network. In other words, you cannot emulate slave nodes using these blocks. However, you can prototype multiple EtherCAT controllers, with multiple Ethernet cards, using these blocks.

This topic assumes that you:

- Are familiar with Ethernet and EtherCAT networks
- Are familiar with CAN networks
- Are familiar with CANopen networks
- Have access to and know how to use the Beckhoff® ET9000 EtherCAT configurator software

### Before You Start

- 1 Provide a dedicated Ethernet card on your target computer. A dedicated Ethernet card is to be used only for model-based EtherCAT communications and not for communication between the host computer and target computer. This requirement means that your target computer must have at least two Ethernet cards, one for host computer to target computer communications, and one for model-based EtherCAT communication.

---

**Tip** For a list of supported chip sets, see “EtherCAT Chip Sets” on page 11-6.

---

- 2 Download or purchase the Beckhoff ET9000 EtherCAT configurator software, available at:

[http://www.beckhoff.com/english.asp?ethercat/et9000\\_et9200\\_et9300.htm](http://www.beckhoff.com/english.asp?ethercat/et9000_et9200_et9300.htm)

- 3 Design and configure an EtherCAT network.
- 4 Create an EtherCAT configuration XML file for this network. The xPC Target EtherCAT blocks require this configuration XML file.

## Basic Workflow

This topic describes a basic workflow for using the xPC Target EtherCAT sublibrary. It assumes that you:

- Have an EtherCAT network
- Have access to the EtherCAT configurator software
- Know how to use the EtherCAT configurator software to create a configuration XML file

See “Before You Start” on page 11-2 for more information.

- 1 Connect to the EtherCAT network.
- 2 Use the Beckhoff ET9000 configurator to configure the EtherCAT network.

This step produces a network configuration file, contained in an XML file.

---

**Note** The xPC Target documentation does not describe how to use Beckhoff ET9000 configurator software.

---

- 3 From the Beckhoff ET9000 configurator, export the XML configuration file.

- 4 In the xPC Target model for the EtherCAT master, add an xPC Target EtherCAT Init block.

---

**Tip** The xPC Target EtherCAT blocks are not available in the standard xPC Target library. To access these blocks, type `xpcethercatlib` in the MATLAB Command Window.

---

- 5 Double-click the Init block and enter the configuration file name from step 3 in the **EtherCAT configuration file** parameter. Use single quotes around the configuration file name.
- 6 Configure the rest of the block.
- 7 Add and configure other EtherCAT blocks, such as the EtherCAT Rx Var and EtherCAT Tx Var blocks. At the very least, each model must contain the EtherCAT Init, EtherCAT Rx Frames, and EtherCAT Update blocks. Add an EtherCAT Rx Var or EtherCAT Tx Var block for every EtherCAT device variable that is linked to a variable in an ET9000 task.

When you add EtherCAT blocks to the model, the blocks automatically obtain the list of variables from the XML configuration file specified in their associated EtherCAT Init block. The parameter dialog boxes of these blocks are updated with the configuration information.

- 8 Repeat steps 1 to 6 for each EtherCAT network that you want. Use a different Ethernet card interface for each EtherCAT network.

### Execution Priority

The EtherCAT blocks have the following execution priority, from first to last:

- 1 EtherCAT Init
- 2 EtherCAT Rx Frames
- 3 EtherCAT Rx Var, EtherCAT Tx Var
- 4 EtherCAT Update

## 5 EtherCAT COE Read, EtherCAT COE Write, EtherCAT Get State, EtherCAT Set State

### xPC Target Applications of the Beckhoff ET9000 EtherCAT Configurator

This topic lists the configurator components that have corresponding xPC Target elements. It does not describe how to use the configurator. See the Beckhoff ET9000 EtherCAT Configurator for documentation on how to use the tool.

- The EtherCAT configuration file that you specify in the EtherCAT Init block describes the network, including the device variables of the network. When you specify a device variable from this network, the block variable information corresponds to signal information on the block mask and block parameter dialog box.
- If your model contains the EtherCAT COE Read and EtherCAT COE Write blocks, include in the configuration file at least one cyclic input/output task linked to one of the slave device input/output channels.
- Do not name the configuration file with the name *modelName.xml*.
- Create at least one task for data transfers.
- To perform data transfers at multiple rates, create multiple tasks in the configurator.

The task rate that you use in the configurator must match the greatest common divisor of the rates in the Simulink model. All tasks run at this rate. The greatest common divisor must be a multiple of the fixed **Sample Time** defined in the block parameter dialog box, or the fixed **Sample Time** must be auto.

- You can create more than one task at the same rate.

ET9000 EtherCAT Configurator Component	Corresponding xPC Target Component
Execution rate of EtherCAT task	Sample Time/Task Rate
Scalars and vectors	Dimension

<b>ET9000 EtherCAT Configurator Component</b>	<b>Corresponding xPC Target Component</b>
BitSize	Type Size (see “EtherCAT Data Types” on page 11-7)
Data Type+BitSize	Signal Type (see “EtherCAT Data Types” on page 11-7)
BitOffs – 80	Signal Offset
EtherCAT device variable that is linked to a variable in an ET9000 task	EtherCAT Rx Var <b>Signal Name</b>
Task scheduling	EtherCAT Update <b>Task priorities</b>
Tasks	EtherCAT Update <b>Task Id List</b>
Each device variable in Process Image entity	Depending on purpose, EtherCAT Rx Var or EtherCAT Tx Var block

### **EtherCAT Chip Sets**

The xPC Target EtherCAT communication blocks support the following Intel (Vendor ID 0x8086) chips:

<b>Device ID</b>	<b>Chip Description</b>
0x1013	I82541EI COPPER
0x1078	I82541ER
0x1076	I82541GI COPPER
0x107C	I82541PI DESKTOP
0x1019	I82547GI COPPER
0x1026	I82545GM COPPER
0x10BD	I82566L
0x10A7	I82575 ZOAR
0x1075	I82547EI
0x108C	I82573

<b>Device ID</b>	<b>Chip Description</b>
0x10B9	I82572GI
0x10A4	I82571GB QUAD
0x10D3	I82574L
0x1229	Intel PRO/100 M Desktop Adapter: 82557, 82558, 82559, 82550, 82551
0x1209	Fast Ethernet Controller: 82551xER, 82551IT
0x1059	Fast Ethernet PCI Controller: 82551QM
0x2449	Integrated 10Base-T/100Base-T TX Ethernet Controller: 8x559ER
0x103A	LAN Controller: 82562ET, 82562EZ, 82563ET, 82801DB
0x1039	LAN Controller: 82562ET, 82562EZ, 82562VE, 82562VM
0x1050	Pro/100 VE Network Connection: 8201EB, 8201ER
0x27DC	Intel PRO/100 VE Desktop Adapter: 82562V

## EtherCAT Data Types

The xPC Target EtherCAT blocks support the following EtherCAT data types:

<b>EtherCAT Data Type</b>	<b>Data Type Size (bits)</b>	<b>Converted Simulink Data Type</b>
bit	1	uint8
bit8	8	uint8
bitarr	8 (bit array)	uint8
bitarr16	16 (bit array)	uint16

<b>EtherCAT Data Type</b>	<b>Data Type Size (bits)</b>	<b>Converted Simulink Data Type</b>
bitarr32	32 (bit array)	uint32
BOOL	1	Boolean
int8	8	int8
int16	16	int16
int32	32	int32
int64	64	Not available
uint8	8	uint8
uint16	16	uint16
uint32	32	uint32
uint64	64	Not available
float	32	real32_T
double	64	real_T

### **EtherCAT Error Values**

The xPC Target EtherCAT blocks return the following EtherCAT error values:

<b>Error Value</b>	<b>Description</b>
65537 (0x10001)	Cyclic command: working counter error
65538 (0x10002)	Master init command: working counter error
65539 (0x10003)	Slave init command: working counter error
65540 (0x10004)	EOE mbox receive: working counter error
65541 (0x10005)	COE mbox receive: working counter error

<b>Error Value</b>	<b>Description</b>
65542 (0x10006)	FOE mbox receive: working counter error
65543 (0x10007)	EOE mbox send: working counter error
65544 (0x10008)	COE mbox send: working counter error
65545 (0x10009)	FOE mbox send: working counter error
65546 (0x1000a)	Got no response on a sent Ethernet frame
65547 (0x1000b)	Got no response on a sent ecats init command from slave
65548 (0x1000c)	Got no response on a sent ecats master init command
65549 (0x1000d)	Missing ecats command in received ethernet frame
65550 (0x1000e)	Timeout when waiting for mailbox init command response
65551 (0x1000f)	Not all slave devices are in operational state when receiving cyclic frames
65552 (0x10010)	Ethernet link (cable) not connected
65553 (0x10011)	Obsolete
65554 (0x10012)	Redundancy: line break detected
65555 (0x10013)	At least one slave is in error state when receiving cyclic frames (BRD AL-STATUS)
65556 (0x10014)	Slave error status information
65557 (0x10015)	Fixed station address lost (or slave missing) - FPRD to AL_STATUS failed

<b>Error Value</b>	<b>Description</b>
65558 (0x10016)	SOE mbox receive: working counter error
65559 (0x10017)	SOE mbox send: working counter error
65560 (0x10018)	SOE mbox write responded with an error
65561 (0x10019)	COE mbox SDO abort
65562 (0x1001a)	Client registration dropped, possibly call to ecatConfigureMaster by other thread (RAS)
65563 (0x1001b)	Redundancy: line is repaired
65564 (0x1001c)	FOE mbox abort

## **Blocks – Alphabetical List**

EtherCAT COE Read  
EtherCAT COE Write  
EtherCAT Get State  
EtherCAT Init  
EtherCAT Rx Frames  
EtherCAT Rx Var  
EtherCAT Set State  
EtherCAT Tx Var  
EtherCAT Update

# EtherCAT COE Read

---

**Purpose** EtherCAT COE Read block

**Library** xPC Target for EtherCAT

**Description** The EtherCAT COE Read block reads a CANopen dictionary entry in the specified EtherCAT slave.

---

**Note** If there is an error while opening the block, the dialog box will show the error message instead of the normal interface. Click **Cancel** to exit.

---

## Block Outputs

Name	Description
D	Data received from the Network Device.
Error	0 if there is no error, otherwise a nonzero value. See “EtherCAT Error Values” on page 11-8.

## Block Parameters

### Index

Specify the index of the CANopen dictionary.

### Subindex

Specify the subindex of the CANopen dictionary entry.

### Data Type

From the list, select the data type of the dictionary entry.

### Dimension

Specify the row and column dimension of the dictionary entry. Enter a value of 1 because EtherCAT blocks support only scalars and vectors.

**Device Index**

From the list, select the integer that identifies the Ethernet card for the EtherCAT network from which to read the EtherCAT variable. This value must match the **Device index** value of the EtherCAT Init block of the EtherCAT network you want to use.

**Slave Name**

From the list, select the name of the slave that contains the CANopen data dictionary variable. The block populates this drop-down list with the contents of the configuration file.

**Sample Time**

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

**Timeout**

Enter the number of seconds to wait for a response from the EtherCAT slave.

# EtherCAT COE Write

---

**Purpose** EtherCAT COE Write block

**Library** xPC Target for EtherCAT

**Description** The EtherCAT COE Write block writes to a CANopen dictionary entry in the specified EtherCAT slave.

---

**Note** If there is an error when you open the block, the dialog box will show the error message instead of the normal interface. Click **Cancel** to exit.

---

## Block Inputs and Outputs

### Inputs

Name	Description
D	Input data for writing to the Network Device.

### Outputs

Name	Description
Error	0 if there is no error, otherwise a nonzero value. See “EtherCAT Error Values” on page 11-8.

## Block Parameters

### Index

Specify the index of the CANopen dictionary.

### Subindex

Specify the subindex of the CANopen dictionary entry.

### Data Type

From the list, select the data type of the dictionary entry.

**Dimension**

Specify the row and column dimension of the dictionary entry.  
Enter a value of 1 because EtherCAT blocks support only scalars and vectors.

**Device Index**

From the list, select the integer that identifies the Ethernet card for the EtherCAT network from which to read the EtherCAT variable. This value must match the **Device index** value of the EtherCAT Init block of the EtherCAT network you want to use.

**Slave Name**

From the list, select the name of the slave that contains the CANopen data dictionary variable. The block populates this drop-down list with the contents of the configuration file.

**Sample Time**

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

**Timeout**

Enter the number of seconds to wait for a response from the EtherCAT slave.

# EtherCAT Get State

---

**Purpose** EtherCAT Get State block

**Library** xPC Target for EtherCAT

**Description** The EtherCAT Get State block returns the state of the EtherCAT slave.

State	Value	Description
INIT	1	Initialization state
PREOP	2	Preoperational state
SAFEOP	4	Safe operational state
OP	8	Operational state

## Block Outputs

Name	Description
State	State received from the Network Device.

## Block Parameters

### Device index

Enter an integer that identifies the Ethernet card for the EtherCAT network from which to read the EtherCAT variable. This value must match the **Device index** value of the EtherCAT Init block of the EtherCAT network you want to use.

### Sample Time

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

**Purpose** EtherCAT Init block

**Library** xPC Target for EtherCAT

**Description** The EtherCAT Init block initializes the EtherCAT master stack, and specifies the Ethernet interface cards used in the network.

---

**Note** Before you use this block, create an XML configuration file. This configuration file cannot have the name *modelName.xml* (see “xPC Target Applications of the Beckhoff ET9000 EtherCAT Configurator” on page 11-5).

---

The xPC Target software can support multiple EtherCAT networks. To use multiple networks:

- Use a different Ethernet card interface for each EtherCAT network.
- In the model, use one EtherCAT Init block for each network.
- Place the EtherCAT Init block in only the top-level subsystem of the root model.

## Block Outputs

Name	Description
Error State	0 if there is no error, otherwise a nonzero value. See “EtherCAT Error Values” on page 11-8.

## Block Parameters

### EtherCAT configuration file

Specify the XML configuration file that you exported from the Beckhoff ET9000 EtherCAT configurator. See “EtherCAT Data Types” on page 11-7 for more information.

**Device index**

(Read-only parameter) An integer that identifies the Ethernet card for the EtherCAT network from which to read the EtherCAT variable. When you add this block to your model, the software automatically assigns a unique device number for this value. If other EtherCAT blocks in this model must use this EtherCAT network, use the same **Device index** value for those blocks.

**PCI bus**

Enter the PCI bus number of the Ethernet card that you will use for model-based communications.

**PCI slot**

Enter the PCI slot number of the Ethernet card that you will use for model-based communications.

**Enable DC**

Select this check box to enable distributed clocks to keep the master and slave in sync. Clear this check box to disable distributed clocks.

**Purpose** EtherCAT Rx Frames block

**Library** xPC Target for EtherCAT

**Description** The EtherCAT Rx Frames block updates the EtherCAT message that the EtherCAT Link Layer receives.

---

### Note

- If there is an error when you open the block, the dialog box will show the error message instead of the normal interface. Click **Cancel** to exit.
  - The EtherCAT receive frame block inherits the fastest sample rate specified in the root model.
- 

### Block Outputs

Name	Description
DC Offset	The distributed clock offset of the frame.

### Block Parameters

#### Device index

Enter an integer that identifies the Ethernet card for the EtherCAT network from which to read the EtherCAT variable. This value must match the **Device index** value of the EtherCAT Init block of the EtherCAT network you want to use.

# EtherCAT Rx Var

---

**Purpose** EtherCAT Rx Var block

**Library** xPC Target for EtherCAT

**Description** The EtherCAT Rx Var block receives data from the EtherCAT slave device.

The block parameter dialog box has two sections, parameters and signal information. When you specify an EtherCAT network and device variable name:

- The EtherCAT Rx Var block mask updates with the selected signal name.
- The signal information in the block parameter dialog box update to reflect the device variable.

---

**Note** If there is an error when you open the block, the dialog box will show the error message instead of the normal interface. Click **Cancel** to exit.

---

## Block Outputs

Name	Description
D	Data received from the Network Device.

## Block Parameters

### Device index

From the list, select an integer that identifies the Ethernet card for the EtherCAT network from which to read the EtherCAT variable. This value must match the **Device index** value of the EtherCAT Init block of the EtherCAT network you want to use.

### Signal name

From the list, select the EtherCAT device variable name.

The signal information in the block parameter dialog box updates to reflect the device variable you select.

<b>Signal Information</b>	<b>Description</b>
Signal Offset	Location in the process image from which the data is available after the execution of the EtherCAT Rx Frames block. This value is the EtherCAT configurator BitOffs – 80.
Signal Type	The Simulink data type for the EtherCAT data. See “EtherCAT Data Types” on page 11-7 for a mapping of Simulink data types to the EtherCAT data types.
Type Size (bits)	Size of the EtherCAT data type. See “EtherCAT Data Types” on page 11-7 for a mapping of Simulink data types to the EtherCAT data types and data type sizes.
Signal Dimension	Dimension of the signal. The EtherCAT blocks support scalars and vectors (dimension of 1).
Sample Time	Rate at which this block is executed. This is the execution rate of the EtherCAT task, as specified in the Beckhoff ET9000 EtherCAT configurator.

# EtherCAT Set State

---

**Purpose** EtherCAT Set State block

**Library** xPC Target for EtherCAT

**Description** The EtherCAT Set State block sets the state of the EtherCAT slave. Each state has the corresponding integer:

State	Value	Description
INIT	1	Initialization state
PREOP	2	Preoperational state
SAFEOP	4	Safe operational state
OP	8	Operational state

## Block Inputs and Outputs

### Inputs

Name	Description
New State	New state for transmission to the Network Device.

### Outputs

Name	Description
Prev State	Previous state of the Network Device.
Error	0 if there is no error, otherwise a nonzero value. See “EtherCAT Error Values” on page 11-8.

## Block Parameters

### Device index

Enter an integer that identifies the Ethernet card for the EtherCAT network from which to read the EtherCAT variable.

This value must match the **Device index** value of the EtherCAT Init block of the EtherCAT network you want to use.

### **Timeout**

Enter the number of seconds to wait for a response from the EtherCAT slave.

### **Sample Time**

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

# EtherCAT Tx Var

---

**Purpose** EtherCAT Tx Var block

**Library** xPC Target for EtherCAT

**Description** The EtherCAT Tx Var transmits computed data to a particular variable in the EtherCAT device.

The block parameter dialog box has two sections, parameters and signal information. When you specify an EtherCAT network and device variable name:

- The EtherCAT Rx Var block mask updates with the selected signal name.
- The signal information in the block parameter dialog box updates to reflect the device variable.

---

**Note** If there is an error when you open the block, the dialog box will show the error message instead of the normal interface. Click **Cancel** to exit.

---

## Block Inputs

Name	Description
D	Data to transmit to the Network Device.

## Block Parameters

### Device index

From the list, select an integer that identifies the Ethernet card for the EtherCAT network from which to read the EtherCAT variable. This value must match the **Device index** value of the EtherCAT Init block of the EtherCAT network you want to use.

### Signal name

From the list, select the EtherCAT device variable name.

The signal information in the block parameter dialog box updates to reflect the device variable that you select.

<b>Signal Information</b>	<b>Description</b>
Signal Offset	Location in the process image from which the data is available after the execution of the EtherCAT Rx Frames block. This value is the EtherCAT configurator BitOffs – 80.
Signal Type	The Simulink data type for the EtherCAT data. See “EtherCAT Data Types” on page 11-7 for a mapping of Simulink data types to the EtherCAT data types.
Type Size (bits)	Size of the EtherCAT data type. See “EtherCAT Data Types” on page 11-7 for a mapping of Simulink data types to the EtherCAT data types and data type sizes.
Signal Dimension	Dimension of the signal. The EtherCAT blocks support scalars and vectors (dimension of 1).
Sample Time	Rate at which this block is executed. This is the execution rate of the EtherCAT task, as specified in the Beckhoff ET9000 EtherCAT configurator.

# EtherCAT Update

---

<b>Purpose</b>	EtherCAT Update block
<b>Library</b>	xPC Target for EtherCAT
<b>Description</b>	The EtherCAT Update block transmits the Simulink model output variables to the EtherCAT device target. This block also asynchronously performs administrative tasks.
<b>Block Parameters</b>	<p><b>Device index</b> Enter an integer that identifies the Ethernet card for the EtherCAT network from which to read the EtherCAT variable. This value must match the <b>Device index</b> value of the EtherCAT Init block of the EtherCAT network you want to use.</p> <p><b>Task ID list</b> Enter a vector of the task IDs that you want to execute. The Beckhoff ET9000 EtherCAT configurator specifies these task IDs.</p> <p><b>Task rates</b> Enter a vector of the task rate execution, in milliseconds. The number of members in this vector must equal the number of task IDs in <b>Task ID list</b>.</p> <p><b>Task priorities</b> Enter the task priorities for the task IDs. These values are task offsets expressed as multiples of the base task rate. The block schedules an ET9000 task using the following formulas:</p> $start\_cycle = task\_priority [task\_id]\%task\_rate[1]$ $task\_scheduled\_on\_cycles\_numbers = start\_cycle + task\_rate[task\_id]\%task\_rate[1]$ <p><b>Async rate</b> Enter the rate of the asynchronous task.</p> <p><b>Async priority</b> Enter the priority of the asynchronous task.</p>

# Real-Time UDP Communication Support

---

- “Using Real-Time UDP Communications” on page 12-2
- “Blocks — Alphabetical List” on page 12-3

# Using Real-Time UDP Communications

Consider using the real-time UDP communication blocks for the following:

- Enhanced performance over standard UDP communications — The UDP communication support that uses real-time Ethernet requires a dedicated Ethernet card. , Applications that use these UDP blocks send and receive UDP packets in real time. The standard UDP communication shares bandwidth with the communication between the host computer and target computer.
- Flexible configuration — The real-time UDP communication supports variable lengths and variable numbers of packets. The standard UDP communication also supports variable lengths, but only one packet at a time.

## Before You Start

Verify that you have a dedicated Ethernet card on your target computer. A dedicated Ethernet card is to be used only for real-time UDP communications, and not for communication between the host computer and target computer. This requirement means that your target computer must have at least two Ethernet cards, one for host computer to target computer communications, and one for real-time UDP communications. For a list of supported Ethernet chip families, see [What Is Model-Based Ethernet Communications?](#).

## **Blocks – Alphabetical List**

Network Configuration

Receive

Send

# Network Configuration

---

<b>Purpose</b>	Real-time UDP Network Configuration block
<b>Library</b>	xPC Target for Real-Time UDP
<b>Description</b>	The Network Configuration block configures the network for real-time UDP operation.
<b>Block Parameters</b>	<b>IP Address</b> Enter the IP address for the interface.
	<b>Subnet Mask</b> Enter the subnet mask for the interface.
	<b>Gateway</b> Enter the gateway address for the interface.
	<b>Driver</b> Identifies the driver for each chip family supported. Possible values are Intel 8255X and Intel Gigabit.
	<b>PCI Bus</b> Enter the PCI bus index for the Ethernet card.
	<b>PCI Slot</b> Enter the PCI slot number for the Ethernet card.
<b>See Also</b>	Receive, Send

<b>Purpose</b>	Real-time UDP Receive block
<b>Library</b>	xPC Target for Real-Time UDP
<b>Description</b>	The Receive block receives UDP packets.
<b>Block Parameters</b>	<p><b>IP address to receive from (0.0.0.0 for accepting all)</b> Enter a valid IP address as a dotted decimal string, for example, 10.10.10.3. You can also use a MATLAB expression that returns a valid IP address as a string. With <b>IP port to receive from</b>, this parameter defines the source address.</p> <p>The default address, 0.0.0.0, enables the acceptance of all UDP packets from any computer. If set to a specific IP address, packets arriving from only that IP address are received.</p> <p><b>IP port to receive from</b> Specify the port of the PC or device from which to receive the UDP packets. With <b>IP address to receive from (0.0.0.0 for accepting all)</b>, this parameter defines the source address.</p> <p><b>Output port width</b> Enter the output port width.</p> <p><b>Sample time</b> Enter the base sample time or a multiple of the base sample time.</p>
<b>See Also</b>	Network Configuration, Send

# Send

---

**Purpose** Real-time UDP Send block

**Library** xPC Target for Real-Time UDP

**Description** The Send block sends UDP packets.

**Block Parameters** **IP address to send to (255.255.255.255 for broadcast)**  
Specify the IP address of the PC to which you want to send the UDP packets. To broadcast the packets to all listening PCs or devices, enter 255.255.255.255. With **Remote IP port to send to**, this parameter defines the destination address.

**Remote IP port to send to**  
Specify the PC port to which you want to send the UDP packets. With **IP address to send to (255.255.255.255 for broadcast)**, this parameter defines the destination address.

**Use the following local IP port (-1 for automatic assignment)**  
Specify the target computer port from which you want to send the UDP packets.

Enter -1 to automatically assign a port for the target computer.

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

**See Also** , Network Configuration, Send

# IP Library for Real-Time UDP Communication Support

---

- “Internet Protocol Blocks” on page 13-2
- “Blocks — Alphabetical List” on page 13-3

### Internet Protocol Blocks

The Internet Protocol (IP) library contains blocks specific to the management of the UDP network. The blocks in this sublibrary are core blocks that you might want to use to create other subsystems. However, the top-level real-time UDP blocks provide enough functionality for real-time UDP communications. To access these blocks, type `xpciplib` in the MATLAB Command Window.

## **Blocks – Alphabetical List**

Network Configuration  
UDP Consume  
UDP Produce  
UDP Rx  
UDP Tx

# Network Configuration

---

<b>Purpose</b>	Real-time UDP Network Configuration block
<b>Library</b>	xPC Target for IP
<b>Description</b>	The Network Configuration block configures the network for real-time UDP operation.
<b>Block Parameters</b>	<p><b>IP Address</b> Enter the IP address for the interface.</p> <p><b>Subnet Mask</b> Enter the subnet mask for the interface.</p> <p><b>Gateway</b> Enter the gateway address for the interface.</p> <p><b>Driver</b> Identifies the driver for each chip family supported. Possible values are Intel 8255X and Intel Gigabit.</p> <p><b>PCI Bus</b> Enter the PCI bus index for the Ethernet card.</p> <p><b>PCI Slot</b> Enter the PCI slot number for the Ethernet card.</p>

**Purpose** Real-time UDP Consume block

**Library** xPC Target for IP

**Description** The UDP Consume block outputs a network buffer with raw data that you can output to a Network Buffer library block. To create this output, the block:

- 1** Receives as input a network buffer that contains a UDP header.
- 2** Removes the UDP header.
- 3** Outputs the updated network buffer.

The block has two output ports:

- **Buffers**  
Chain of network buffers.
- **Chain size**  
Number of buffers on the chain.

## **Block Parameters**

### **Output port width**

Enter the width of the port. A value other than 0 creates the following output ports:

- Source IP Address
- Destination IP Address
- Local UDP Port
- Remote UDP Port

### **Sample time**

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

# UDP Produce

---

<b>Purpose</b>	Real-time UDP Produce block
<b>Library</b>	xPC Target for IP
<b>Description</b>	The UDP Produce block receives a network buffer and adds a header to that buffer. It then outputs that updated buffer.
<b>Block Parameters</b>	<p><b>IP address to send to (255.255.255.255 for broadcast)</b> Specify IP address of the PC to which to send the UDP packets. To broadcast the packets to all listening PCs or devices, enter 255.255.255.255. With <b>Remote IP port to send to</b>, this parameter defines the destination address.</p> <p><b>Remote IP port to send to</b> Specify the PC port to which to send the UDP packets. With <b>IP address to send to (255.255.255.255 for broadcast)</b>, this parameter defines the destination address.</p> <p><b>Use the following local IP port (-1 for automatic assignment)</b> Specify the target PC port from which to send the UDP packets.  Enter -1 to automatically assign a port for the target PC.</p> <p><b>Sample time</b> Enter a base sample time or a multiple of the base sample time.</p>

---

<b>Purpose</b>	Real-time UDP Rx block
<b>Library</b>	xPC Target for IP
<b>Description</b>	The UDP Rx block outputs a network buffer with a UDP header.
<b>Block Parameters</b>	<p><b>IP address to receive from (0.0.0.0 for accepting all)</b> Enter a valid IP address as a dotted decimal string. For example, 10.10.10.3. You can also use a MATLAB expression that returns a valid IP address as a string. With <b>IP port to receive from</b>, this parameter defines the source address.</p> <p>The default address, 0.0.0.0, enables the acceptance of all UDP packets from any computer. If set to a specific IP address, only packets arriving from that IP address are received.</p> <p><b>IP port to receive from</b> Specify the port of the PC or device from which to receive the video frames. With <b>IP address to receive from (0.0.0.0 for accepting all)</b>, this parameter defines the source address.</p> <p><b>Sample time</b> Enter a base sample time or a multiple of the base sample time.</p>

# UDP Tx

---

<b>Purpose</b>	Real-time UDP Tx block
<b>Library</b>	xPC Target for IP
<b>Description</b>	The UDP Tx block receives a network buffer with a UDP header and sends it.
<b>Block Parameters</b>	<b>Sample time</b> Enter a base sample time or a multiple of the base sample time.

# UDP I/O Support

---

- “User Datagram Protocol (UDP)” on page 14-2
- “Two-Way Data Exchange Using UDP” on page 14-6
- “UDP Communication Setup” on page 14-12
- “UDP and Variable-Size Signals” on page 14-14
- “Boards and Blocks — Alphabetical List” on page 14-16

## User Datagram Protocol (UDP)

### In this section...

“What Is UDP?” on page 14-2

“Why UDP?” on page 14-4

“Note on UDP Communication” on page 14-4

### What Is UDP?

The xPC Target software supports communication from the target computer 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.

To use UDP for your xPC Target system, be sure to create a TCP/IP boot disk and boot the target computer with that boot disk.

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 Supports Error Detection** — 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 order by the receiver.

Further, timeouts 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** *Reliable* is used in the sense of “not guaranteed to succeed” as opposed to “succeeds all of the time”. *Unreliable* is used in the sense of “may fail (packets may not arrive) without the application being aware that the packet did not arrive” as opposed to “fails most of the time.” In practice, UDP will continue receiving packets as long as the receiving socket is active and is processing data as quickly as it arrives.

---

## Why UDP?

UDP was chosen as the xPC Target transport layer 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 gives the real-time application a good chance of succeeding in real-time execution. Also, the datagram nature of UDP is good for sending samples of data from the application generated by the Simulink Coder software. 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. The xPC Target block library provides blocks for combining various signals into one packet (packing), and then transmitting it. It 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 1450 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 the receipt of further packets after the ones that were lost bridges any gap. .

Note also that because UDP block transfers operate as background tasks, the xPC Target software disables them in polling mode. See [Restrictions Introduced by Polling Mode](#).

## Two-Way Data Exchange Using UDP

This section provides an example of how to set up two-way data exchange between two xPC Target systems, between the xPC Target and Simulink products, or between two Simulink models. When one or both of the systems is running Simulink in nonreal time, be sure to set the sample time.

---

**Note** To use UDP for your xPC Target system, be sure to create a TCP/IP boot disk and boot the target computer 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 Simulink Coder software uses this function at the time of code generation to generate random numbers. 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 `udpreceiveA`.

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** parameter to 192.168.0.11. This is the hypothetical address of the system that will run `udpreceiveA`. Set the **IP Port to send to** parameter to 25000 (picked arbitrarily). The sample time is set to 0.01.

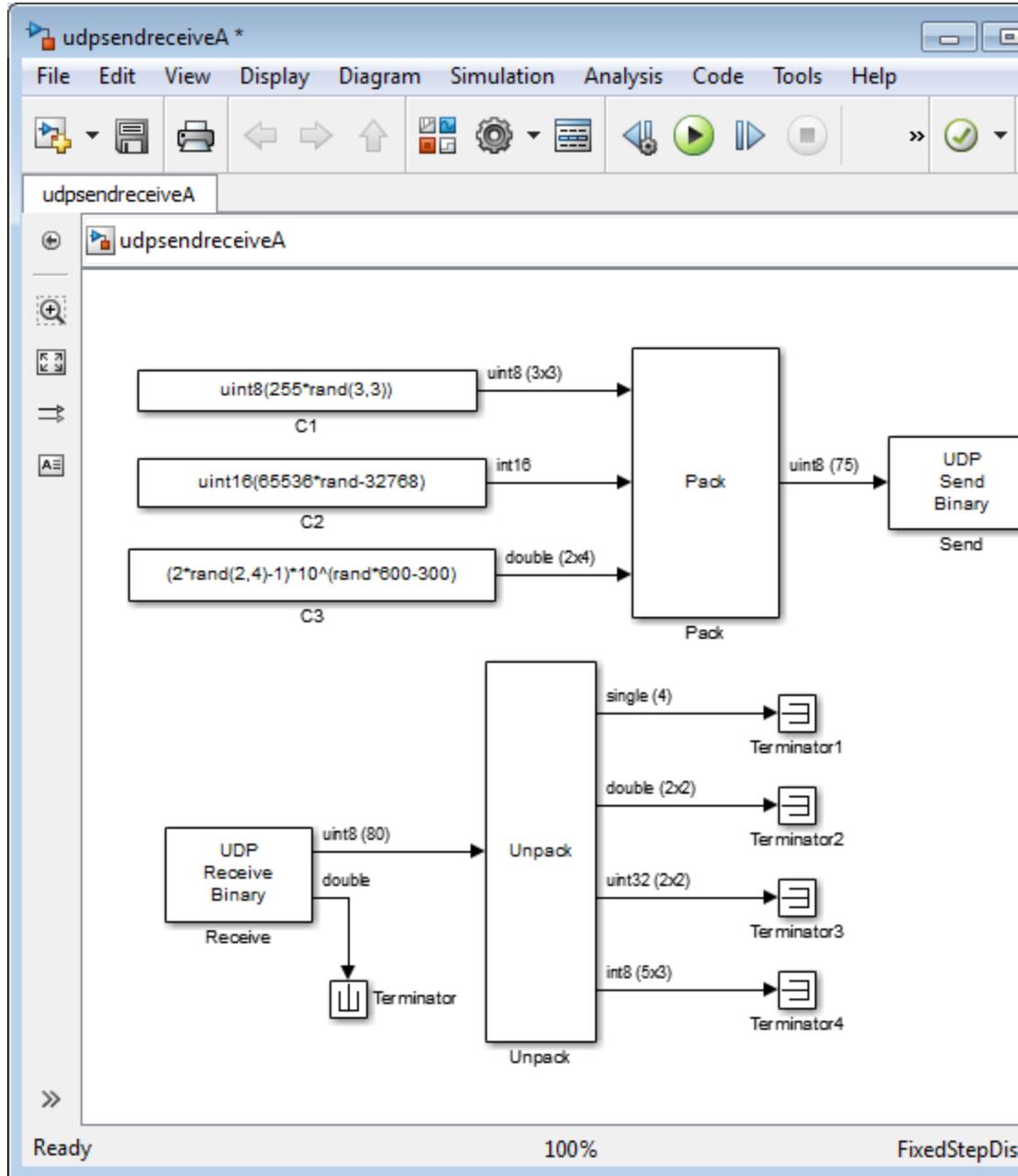
Use this information to construct the receive end of `udpreceiveB`.

For setting up the Receive block, set **IP address to receive from** to 192.168.0.10 (the hypothetical address of the system that will run `udpreceiveB`). The **IP port to receive from** is set to 25000 (the same value as set in the **Remote IP port to send to** parameter of the Send block in `udpreceiveA`). The **Output port width** is set to 75, which is obtained from the output port width of the Pack block in `udpreceiveA`.

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 `udpreceiveA`. The **Output port dimensions** is set to `{[3 3],1,[2 4]}` from the dimensions of the inputs to the Pack block in `udpreceiveA`.

Note that in `udpreceiveB`, 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.

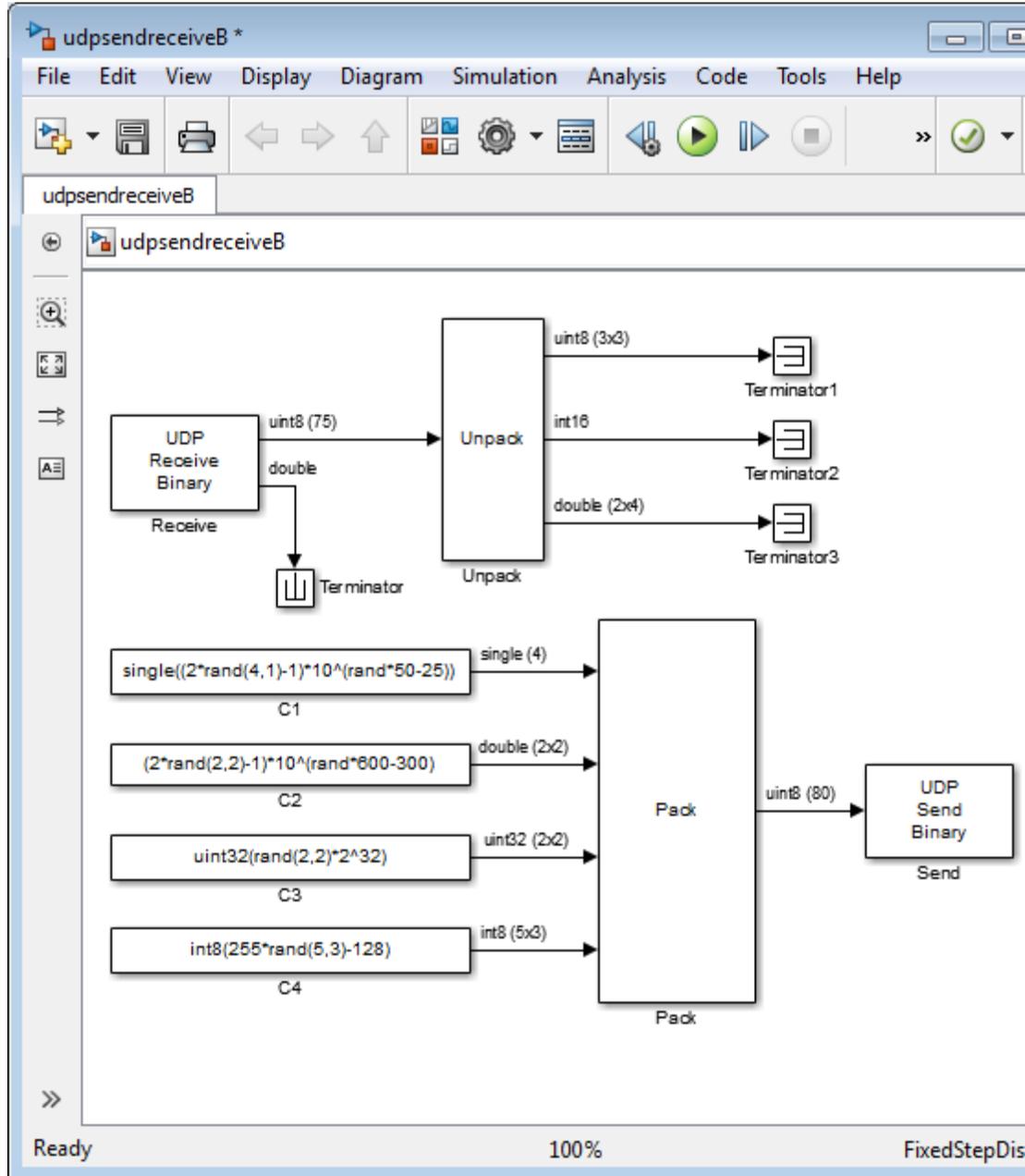
To construct the udpsendreceiveB to udpsendreceiveA side of the communication, follow an analogous procedure. The final udpsendreceiveA is shown below.



The following table lists the parameters in `udpreceiveA`.

<b>Block</b>	<b>Parameter</b>	<b>Value</b>
Receive	IP address	192.168.0.11
	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 udpsendreceiveB 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	192.168.0.10
	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 1450 bytes (i.e., a 1 by 1450 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 `uint8s` 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).

The xPC Target block library 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 the xPC Target or Microsoft® Windows® software.

All the blocks are set up to work both from within the Simulink environment and from an application running under the xPC Target system. 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 nonreal 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 blocks can communicate.

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.

See the UDP Data Streaming example for a example of how to perform UDP data streaming with the xPC Target system.

## UDP and Variable-Size Signals

The xPC Target UDP sublibrary does not directly support variable-size signals. Of particular note, the UDP Send block input port accepts only fixed-size signals. This topic describes a workaround.

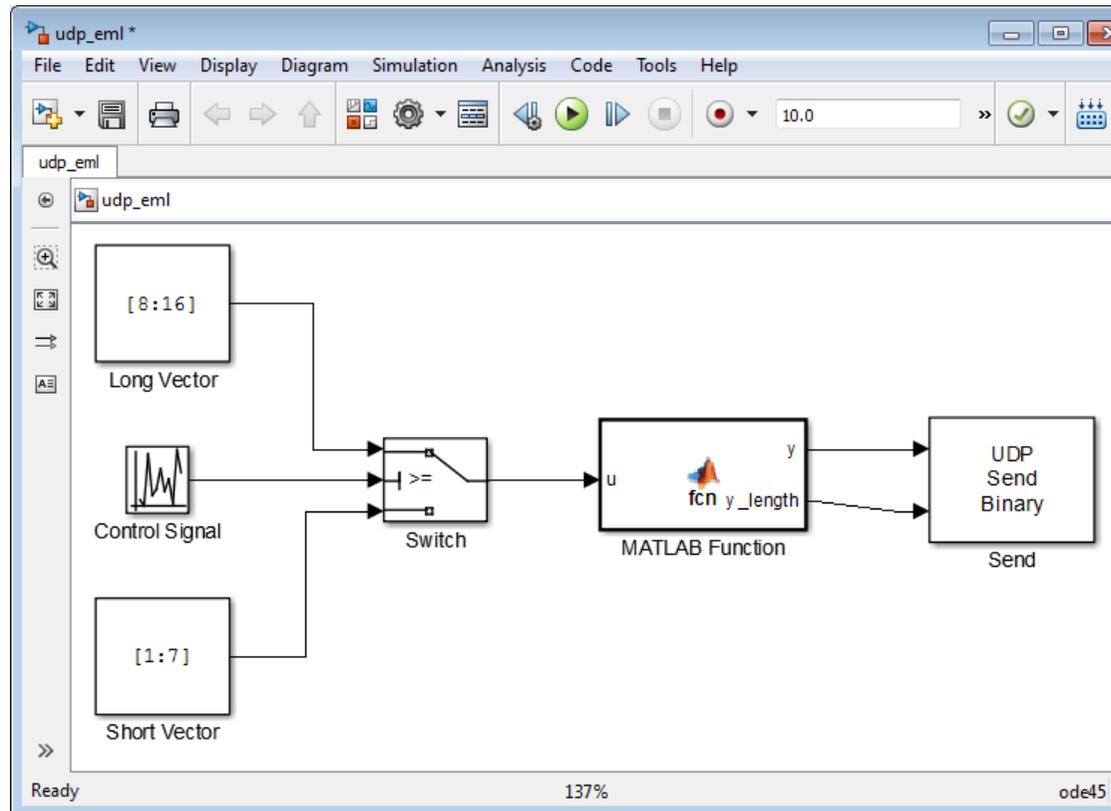
To send variable-size signals through UDP in a target application, select the **Allow variable length packets** parameter of the UDP Send block. This check box creates a second input for the block. With this second input, you can specify the number of elements of the fixed-size input signal to send through UDP.

Consider the following example. This example configures the MATLAB Function block to accept a variable-size signal and maps that signal to a fixed-size output signal. It also outputs the number of relevant elements. You can then output the fixed-size output signal and number of elements to the inputs of the UDP Send block.

- Create an MATLAB Function block to accept a variable-size input signal.
- Enter code like the following in the MATLAB Function block. In this code, the maximum size of the variable-size input signal is 9.

```
function [y,y_length] = fcn(u)
%#codegen
y = uint8(zeros(9,1));
y_length = length(u);
for a = 1:y_length
    y(a) = u(a);
end
```

- In the MATLAB Function Editor, select **Tools > Edit Data/Ports**. In **Ports and Data Manager**, select the data `u`, then select the corresponding **Variable size** check box.
- Select the data `y` and enter the size of the variable-size data input signal in the corresponding **Size** parameter. For this example, the size value is 9.
- Provide a variable-size signal source for the MATLAB Function block.



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

# Byte Reversal/Change Endianness

---

## Purpose

Byte Reversal/Change Endianness block

## Description

You use the Byte Reversal/Change Endianness 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 Endianness block before the Pack block and another just after the Unpack block. The following is the Change Endianness block.

## Block Parameters for Change Endianness

### 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

---

<b>Purpose</b>	UDP Pack block
<b>Library</b>	xPC Target Library for UDP
<b>Description</b>	The Pack block is used to convert one or more Simulink signals of varying data types to a single vector of <code>uint8</code> as required by the Send block.
<b>Block Parameters</b>	<p><b>Input port data types</b></p> <p>Specify the data types for the different signals as part of the block parameters. The supported data types are <code>double</code>, <code>single</code>, <code>int8</code>, <code>uint8</code>, <code>int16</code>, <code>uint16</code>, <code>int32</code>, <code>uint32</code>, and <code>boolean</code>. The block determines the sizes of the signals automatically. This parameter is usually the same as the <b>Output port data types</b> parameter of the corresponding UDP Unpack 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 starts each element in the list of signals on a boundary specified by the alignment relative to the start of the vector.</p>
<b>Example</b>	<p>The data types of each of the signals must be specified as a cell array of strings in the required order. Once this is done, the block automatically converts itself to one with the corresponding number of input ports. There is always one output port. For example, say the Input port data types are specified as</p> <pre>{'uint8', 'uint32', 'single', 'int16', 'double'}</pre> <p>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</p>

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.

---

### **See Also**

UDP Unpack

UDP Send

# 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.

Regardless of the data type, this block returns its data as a byte stream (represented as a vector of `uint8` values). Use the UDP Unpack block to convert from the vector of `uint8` values to the signal data type. See the UDP Data Streaming example for an example of how to use the UDP Unpack block to extract your data.

## Block Parameters

### IP address to receive from

Enter a valid IP address as a dotted decimal string. For example, `10.10.10.3`. You can also use a MATLAB expression that returns a valid IP address as a string.

Can be left with the default value of `0.0.0.0`. This accepts all UDP packets from any computer. If set to a specific IP address, only packets arriving from that IP address are received.

### IP port to receive from

Receiver port that the block accepts data from. The other end of the communication sends data to the port specified here. This value must match the **Remote IP port to send to** parameter of the UDP Send block

### Output port width

Width of the acceptable packets, in bytes. You can obtain this number when designing the other side (send side) of the communication. If you select **Allow variable length packets**, this parameter defines the maximum length of the message.

If **Allow variable length packets** is cleared, the block receives only packets of **Output port width** length. It discards the other packets.

### **Allow variable length packets**

Select this check box to enable the reception of variable length messages. The value of **Output port width** defines the maximum length of the message. If this check box is selected, the first output port sends the actual data. If the port size is less than the size of the actual data, the block sends up to the allowed length of data. It discards the extra data. The UDP Receive block does not return the number of discarded bytes. The behavior of the second output port depends on what environment the block is in:

- If the block is in the xPC Target environment, the second output port sends the length of the original incoming message, *m*.
- If the block is in the Simulink environment, the second output port sends the size of the first output port +1.

If this check box is cleared (default), the first output port contains the actual data, and the second output port contains:

- 1, if the block receives the packet.
- 0, if the block does not receive the packet.

### **Sample time**

You can set this parameter to -1 for an inheritable sample time, but you should set this parameter to some specific (small) value to eliminate chances of dropped packets. This is especially true when you are using a small base sample time.

### **See Also**

UDP Send

UDP Unpack

UDP Data Streaming

# UDP Send

---

**Purpose** UDP Send block

**Library** xPC Target Library for UDP

**Description** The Send block has one input port, which receives the `uint8` vector that is sent as a UDP packet.

Regardless of the data type of the data being sent, you must supply a byte stream (represented as a `uint8` vector) to the Send block. Use the UDP Pack block to convert from the signal data type to a single vector of `uint8` values. See the UDP Data Streaming example for an example of how to use the UDP Pack block to convert your data to a `uint8` vector.

## Block Parameters

### IP address to send to

Specify the IP address to send the packet. Enter a valid IP address as a dotted decimal string. For example, `10.10.10.3`. You can also use a MATLAB expression that returns a valid IP address as a string.

### Remote IP port to send to

Specify the port to which to send the packet. This value must match the **IP port to receive from** parameter of the UDP Receive block

### 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.

### Allow variable length packets

Select this check box to enable the sending of variable length messages. Selecting this check box enables a second input port.

The width of the first signal entering the first input port specifies the maximum length of the input message. If this check box is selected, the first input port receives the actual data, and the second input port receives the actual length of the message.

If this check box is cleared, the block sends all the data each time.

## **Sample time**

You can set this parameter to -1 for an inheritable sample time, but you should set this parameter to some specific (large) value to eliminate chances of dropped packets. This is especially true when you are using a small base sample time.

## **See Also**

UDP Pack

UDP Receive

UDP Data Streaming

# 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 <code>uint8</code> 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> Contains a cell array, with each element the dimension returned by the MATLAB size function 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 data types</b> Specify the data types for the different signals as part of the block parameters. The supported data types are <code>double</code>, <code>single</code>, <code>int8</code>, <code>uint8</code>, <code>int16</code>, <code>uint16</code>, <code>int32</code>, <code>uint32</code>, and <code>boolean</code>. This parameter is usually the same as the <b>Input port data types</b> parameter of the corresponding UDP Pack block.</p> <p><b>Byte alignment</b> 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 starts each element in the list of signals on a boundary specified by the alignment relative to the start of the vector.</p>
<b>Note on Byte Alignment</b>	The byte-alignment feature provided in the Pack and Unpack blocks is primarily intended for interfacing a system running the xPC Target software to another system running neither Simulink nor xPC Target software. 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. You should use a byte-alignment value of 1 (tightly packed) if the source and target data formats permit it. (This is easily

accomplished when UDP I/O is used to exchange data between two xPC Target systems or between xPC Target and Simulink systems.)

Even when communication is between the xPC Target system 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 cases when this is not possible.

# UDP Unpack

---

# Parallel Ports

---

- “Using Parallel Ports” on page 15-2
- “Boards and Blocks — Alphabetical List” on page 15-6

## Using Parallel Ports

In this section...
“Introduction” on page 15-2
“Using the Parallel Port as an Interrupt Source” on page 15-3
“Using Add-On Parallel Port Boards” on page 15-4

### Introduction

Most target PCs have a parallel port that you can use for a variety of devices. The xPC Target block library provides blocks that enable you to use the parallel ports of a target PC for digital input and output, and source interrupts.

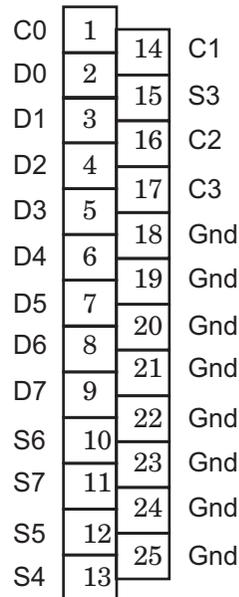
**Warning** The parallel port is part of the motherboard on many PCs. Be careful when configuring the port and connecting external hardware to the port. Incorrect connections to the port might damage your PC.

The xPC Target parallel port blocks assume that the connector to the parallel port has one 25-pin connector whose pins have the following designations:

- Eight data pins
- Five status pins
- Four control pins
- Eight ground pins

Function	Channel	1	2	3	4	5	6	7	8	Additional Pins
	Bit	0	1	2	3	4	5	6	7	
Digital Input		02	03	04	05	06	07	08	09	
Digital Output		02	03	04	05	06	07	08	09	
Digital Input (Status)		15	13	12	10	11				

Function	Channel	1	2	3	4	5	6	7	8	Additional Pins
	Bit	0	1	2	3	4	5	6	7	
Digital Output (Control)		01	14	16	17					
Interrupt										10



## Using the Parallel Port as an Interrupt Source

To use the parallel port as an interrupt source, use pin 10 of the parallel port as the interrupt source. Configure the xPC Target model as follows:

- 1** Select **Simulation > Model Configuration Parameters**.
- 2** Under node **Code Generation**, select node **xPC Target options**.
- 3** In the **Execution options** pane:
  - From **Execution mode**, select Real-Time.

- From **Real-time interrupt source**, select the IRQ level (typically 7).
- From **I/O board generating the interrupt**, select `Parallel_Port`.
- In **PCI slot (-1: autosearch) or ISA base address**, enter the base address of the parallel port (typically `0x378`).

If you want to use the Async IRQ Source block, you do not have to configure the model. Instead, you can set the Async IRQ Source block parameters as follows:

- **IRQ line number** — Select the IRQ level (typically 7).
- **Allow preemption of function-call subsystem** — Set as you wish.
- **I/O board generating the interrupt** — Select `Parallel_Port`.
- **PCI slot** — Enter the base address of the parallel port (typically `0x378`).

## Using Add-On Parallel Port Boards

To use an add-on parallel port board with the parallel port blocks, configure the base address for the board as follows:

- 1** To get the base address of a board, in the MATLAB Command Window, use the `xpctarget.xpc.getxpcpci` function with the 'verbose' option. For example

```
getxpcpci('verbose')
```

- 2** Identify the base address for the add-on parallel port board.
- 3** In your model, open the parallel port block and set the value of the **Base address** parameter to `Other`.

The **Alternate base address** parameter is displayed.

- 4** In the **Alternate base address** parameter, enter the base address you identified in step 2.
- 5** Configure the rest of the block as desired.

---

**Note** You cannot use add-on parallel port boards as interrupt sources. You also cannot trigger the execution of a model with these boards.

---

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

# Parallel Port Digital Input

**Purpose** Parallel Port Digital Input block

**Library** xPC Target Library for Parallel Port

## Scaling Input to Output

Hardware Input	Block Output Data Type	Scaling
TTL	Double ( <b>Format:8</b> 1-bit Channels)	Double: TTL low = 0.0 TTL high = 1.0
	uint8 ( <b>Format:One</b> 8-bit Port)	uint8: TTL low corresponding bit is clear TTL high corresponding bit is set

## Block Parameters

### Base address

Select a parallel port base address. This address depends on the PC BIOS. From the list, select one of the following. If your base address is not one of the supplied standard base addresses, select **Other** and enter your base address in **Alternate base address**.

- 0x3bc
- 0x378
- 0x278
- Other

### Alternate base address

Enter an alternate parallel port base address, in hexadecimal. This parameter appears only if you select **Other** for **Base address**. For example,

0x300

# Parallel Port Digital Input

---

## **Format**

From the list, select one of the following modes to specify how to treat data:

- **8 1 bit Channels**

Treats data as individual bits. Configures block to accept up to eight 1-bit channels.

- **One 8 bit Port**

Treats data as a single byte. Configures block to accept one 8-bit port.

## **Channels**

Enter a vector of numbers between 1 and 8. This parameter appears only if you select **8 1 bit Channels** for **Format**. For example,

[1, 3]

## **Sample time**

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

# Parallel Port Digital Input Status Bits

**Purpose** Parallel Port Digital Input Status Bits block

**Library** xPC Target Library for Parallel Port

<b>Scaling Input to Output</b>	<b>Hardware Input</b>	<b>Block Output Data Type</b>	<b>Scaling</b>
	TTL	Double ( <b>Format:5</b> 1-bit Channels)	Double: TTL low = 0.0 TTL high = 1.0
		uint8 ( <b>Format:One</b> 5-bit Port)	uint8: TTL low corresponding bit is clear TTL high corresponding bit is set

**Block Parameters**

**Base address**  
Select a parallel port base address. This address depends on the PC BIOS. From the list, select one of the following. If your base address is not one of the supplied standard base addresses, select **Other** and enter your base address in **Alternate base address**.

- 0x3bc
- 0x378
- 0x278
- Other

# Parallel Port Digital Input Status Bits

---

## Alternate base address

Enter an alternate parallel port base address, in hexadecimal. This parameter appears only if you select **Other** for **Base address**. For example,

0x300

## Format

From the list, select one of the following modes to specify how to treat data:

- 5 1 bit Channels

Treats data as individual bits. Configures block to accept up to five 1-bit channels.

- One 5 bit Port

Treats data as a single byte. Configures block to accept one 5-bit port.

## Channels

Enter a vector of numbers between 1 and 5. This parameter appears only if you select **5 1 bit Channels** for **Format**. For example,

[1, 3]

## Sample time

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

**Purpose** Parallel Port Digital Output block

**Library** xPC Target Library for Parallel Port

## Scaling Output to Input

Hardware Output	Block Input Data Type	Scaling
TTL	Double ( <b>Format:8</b> 1-bit Channels)	Double: < 0.5 = TTL low > 0.5 = TTL high
	uint8 ( <b>Format:One</b> 8-bit Port)	uint8: Bit clear = TTL low Bit set = TTL high

## Block Parameters

### Base address

Select a parallel port base address. This address depends on the PC BIOS. From the list, select one of the following. If your base address is not one of the supplied standard base addresses, select **Other** and enter your base address in **Alternate base address**.

- 0x3bc
- 0x378
- 0x278
- Other

### Alternate base address

Enter an alternate parallel port base address, in hexadecimal. This parameter appears only if you select **Other** for **Base address**. For example,

0x300

# Parallel Port Digital Output

---

## Format

From the list, select one of the following modes to specify how to treat data:

- 8 1 bit Channels

Treats data as individual bits. Configures block to accept up to eight 1-bit channels.

- One 8 bit Port

Treats data as a single byte. Configures block to accept one 8-bit port.

## Channels

Enter a vector of numbers between 1 and 8. This parameter appears only if you select 5 1 bit Channels for **Format**. For example,

[1, 3]

## 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.

# Parallel Port Digital Output Control Bits

---

**Purpose** Parallel Port Digital Output Control Bits block

**Library** xPC Target Library for Parallel Port

## Scaling Output to Input

Hardware Output	Block Input Data Type	Scaling
TTL	Double ( <b>Format:</b> 4 1-bit Channels)	Double: < 0.5 = TTL low > 0.5 = TTL high
	uint8 ( <b>Format:</b> One 4-bit Port)	uint8: Bit clear = TTL low Bit set = TTL high

## Block Parameters

### Base address

Select a parallel port base address. This address depends on the PC BIOS. From the list, select one of the following. If your base address is not one of the supplied standard base addresses, select Other and enter your base address in **Alternate base address**.

- 0x3bc
- 0x378
- 0x278
- Other

### Alternate base address

Enter an alternate parallel port base address, in hexadecimal. This parameter appears only if you select Other for **Base address**. For example,

0x300

# Parallel Port Digital Output Control Bits

---

## Format

From the list, select one of the following modes to specify how to treat data:

- 4 1 bit Channels

Treats data as individual bits. Configures block to accept up to four 1-bit channels.

- One 4 bit Port

Treats data as a single byte. Configures block to accept one 4-bit port.

## Channels

Enter a vector of numbers between 1 and 4. This parameter appears only if you select 4 1 bit Channels for **Format**. For example,

[1, 3]

## 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.

# Parallel Port Digital Output Control Bits

---

## **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.

# SAE J1939

---

- “SAE J1939 Blocks” on page 16-2
- “Blocks — Alphabetical List” on page 16-3

## **SAE J1939 Blocks**

The xPC Target J1939 blocks enable you to send and receive messages over a CAN network using the SAE J1939 message protocol. Before you start, provide a J1939 database in .dbc format.

Note the following:

- Use the FIFO mode of the Softing CAN hardware with these blocks.

The J1939 I/O - Using Transport Protocol example illustrates the transmission and reception of J1939 data. It uses a loopback connection of two CAN ports in a single target computer. Note, the example requires a J1939 database file, C:\work\J1939\_demo.dbc, which the xPC Target software does not supply. You must provide your own database file, which must contain the requested information outlined in the example.

This topic assumes familiarity with the SAE J1939 specifications.

## **Blocks – Alphabetical List**

# J1939 Controller Application

---

**Purpose** J1939 Controller Application

**Library** xPC Target Library for CAN/J1939

**Description** The J1939 Controller Application block supports address claiming. It enables the dynamic exchange of address information with other nodes in the J1939 bus, resolving conflicts. The effect is that no other block on the CAN network has the same address or the same ID.

Use this block to register your J1939 device on the CAN bus. Associate this block with the J1939 Transmit Message and J1939 Receive Message blocks.

The block has two tab groups, **General** and **NAME**. The **General** tab contains general block information. The **NAME** tab has ten parameters that, when combined, create a unique identification address (NAME) for this J1939 device. Refer to the SAE J1939-81 and base SAE J1939 specifications for details.

## Block Inputs

None

## Block Outputs

This block has the following output port:

Status output (Optional)  
Outputs a status signal.

Current node address output (Optional)  
Outputs current node address.

## Block Parameters

**General** tab:

**CA ID**  
Enter the controller application ID.

**Protocol Stack ID**  
Enter the protocol stack ID.

**Node address**

From a range of 0 to 253, specify the node address of the node for which J1939 communication is to occur. This value is the source address for the J1939 communication. Set this address to the same as that in the J1939 Transmit Message block.

**Show status output**

Select this check box to enable an output status signal.

**Show current node address as output**

Select this check box to enable an output current node address signal.

**NAME** tab:

**Identity Number**

Enter the identity number for the controller application. Use the identify number provided by the ECU manufacturer.

**Manufacturer Code**

Enter the code of the electronic control unit (ECU) manufacturer.

**ECU Instance**

Enter a number to identify the particular ECU associated with **Function**.

If this number identifies the first or only instance, enter 0.

**Function Instance**

Enter a number to identify an instance of the function for the **Vehicle System** in the CAN network.

**Function**

Enter an 8-bit value for the function for this controller application. See Appendix B of the base SAE J1939 specification for a list of allowed function values.

If you enter a value between 0 and 127, the block independently evaluates the function value.

# J1939 Controller Application

---

If you enter a value greater than 127, the block takes into account the value of the **Vehicle System** parameter when evaluating the function value.

## **Reserved**

Reserved. Leave set to 0.

## **Vehicle System**

Enter a 7-bit value for the vehicle system for this controller application. See Appendix B of the base SAE J1939 specification for a list of allowed vehicle system values.

## **Vehicle System Instance**

Enter a number to identify an instance of the vehicle system in the CAN network.

If this number identifies the first or only instance, enter 0.

## **Industry Group**

Enter a 3-bit value for the industry group for this controller application. See Appendix B of the base SAE J1939 specification for a list of allowed industry group values.

## **Arbitrary Address Capable**

Select this check box to allow the controller application to resolve address claim conflicts with arbitrary source addresses.

## **See Also**

SAE J1939-81 specification

# J1939 Database (CANdb) Setup

---

<b>Purpose</b>	J1939 Database Setup
<b>Library</b>	xPC Target Library for CAN/J1939
<b>Description</b>	The J1939 Database Setup block is where you specify the user-supplied database for the J1939 block set. Use one block per model.

## Block Inputs

None

## Block Outputs

None

## Block Parameters

### J1939 database file

Specify the J1939 database location and file name. For example, enter `J1939.dbc` if the file is in the current directory; otherwise enter the full path with the file name, such as `C:\work\J1939.dbc`.

This file defines the J1939 message set and is in a format defined by Vector Informatik GmbH.

---

**Note** This file is a customer-supplied database file. The xPC Target software does not supply this file.

---

### J1939 Database Format

From the list, select:

- **J1939 PG — Full Extended CAN ID**

Specifies that the database format is the new version of CANdb. The block extracts the Parameter Group Number (PGN) from the full extended CAN ID (29-bit).

- **J1939 PG**

# J1939 Database (CANdb) Setup

---

Specifies that the database format is simplified ID (CANDB++ 2.7 SP7 and before).

**Purpose** J1939 Message Triggering

**Library** xPC Target Library for CAN/J1939

**Description** The J1939 Message Trigger block triggers the transmission of a J1939 message under conditions that the block specifies. When using this block in your model, connect its Trigger Out port to the Trigger input port of the J1939 Transmit Message block. The possible triggering conditions are:

- Nonzero input at the Trigger Message input port
- Expiration of the repetition interval
- A detected change in the input signal

### Block Inputs

This block has the following input ports:

Trigger Message

Manually triggers the message transmission when the input value is 1.

Signal For Change Detection

Detects changes in the input.

### Block Outputs

This block has the following output ports:

Trigger Out

Outputs an active signal when a condition for triggering a message occurs. Connect this output to the Trigger input port of the J1939 Transmit Message block.

### Block Parameters

**Repeat on interval**

Select this check box to enable periodic triggering at the interval specified in the **Repetition Interval** parameter.

# J1939 Message Trigger

---

## **Repetition Interval (ms)**

Specify the enabled repetition interval in milliseconds. The repetition interval must be an integer multiple of the model update rate.

## **Send on a change in signal**

Select this check box to enable triggering when both of the following are true:

- Input signal wired to the Signal For Change Detection port input port changes by at least the value in the **Minimum Change Threshold** parameter.
- Specified time in the **Minimum Change Interval** parameter has expired.

## **Minimum Change Threshold (%)**

Specify the threshold change to trigger a message. The trigger event is a change in the input signal relative to the signal value from the previous triggering.

## **Minimum Change Interval (ms)**

Specify the minimum time for the block to wait after a message transmission occurs because of a change in the signal.

<b>Purpose</b>	J1939 Protocol Stack instance
<b>Library</b>	xPC Target Library for CAN/J1939
<b>Description</b>	<p>The J1939 Protocol Stack block defines a J1939 protocol stack instance that you can associate with the CAN boards. This block is useful for the transmission and reception of CAN data to or from the bus. For each selected CAN board, you must have a corresponding CAN FIFO Setup block in the model. The protocol stack handles both the regular J1939 communication and the (optional) Transport Protocol functionality. Use this block with the J1939 Transmit Message and/or J1939 Receive Message blocks.</p>
<b>Block Inputs</b>	None
<b>Block Outputs</b>	None
<b>Block Parameters</b>	<p>General tab:</p> <p><b>Protocol Stack ID</b> Enter the protocol stack ID.</p> <p><b>Max CAN Message Receive (Per Sample Time)</b> Specify maximum number of CAN messages that the block can receive (process) in a single sample time.</p> <p><b>Max CAN Message Transmit (Per Sample Time)</b> Specify maximum number of CAN messages the block can send in a single sample time.</p> <p><b>Sample time</b> Specify the sample rate for the protocol stack.</p> <p>Transport Protocol tab:</p>

# J1939 Protocol Stack

---

## **Enable Transport Protocol**

Select this check box to enable the transport protocol. Enabling the transport protocol allows for the sending and receiving of J1939 data, whose size is greater than 8 bytes, over the CAN bus.

## **Maximum Concurrent Sessions**

Specify maximum transport protocol sessions that can be active at a given time.

## **Network Management tab**

### **Enable Address Claiming**

Select this check box to enable address claiming. Selecting this check box enables J1939 Controller Application blocks to negotiate node addresses with other nodes on the bus.

<b>Purpose</b>	J1939 Receive Message
<b>Library</b>	xPC Target Library for CAN/J1939
<b>Description</b>	The J1939 Receive Message block receives a J1939 message. The J1939 database file defines the message type. You specify the J1939 database with the J1939 Database (CANdb) Setup block.
<b>Block Inputs</b>	None
<b>Block Outputs</b>	<p>Signal Output(s)</p> <p>Depending on the J1939 message defined in the J1939 database file, the block can have multiple output signal ports. If the bit length of a signal exceeds 32, the output is a byte array; otherwise, the block output data type is double.</p> <p>New Message Received (Optional)</p> <p>Outputs 1 when a new message is received from the CAN bus; otherwise, outputs 0.</p>
<b>Block Parameters</b>	<p><b>PGN</b></p> <p>From the list, select the parameter group number. The contents of this list vary depending on the messages that the J1939 database file specifies.</p> <p><b>CA ID</b></p> <p>Specify the ID of the controller application that this block maps to. The ID must match the <b>CA ID</b> value of the corresponding J1939 Controller Application block.</p> <p><b>Source Address Filter (255:all)</b></p> <p>Specify the source node address from which the block is to expect messages. Type 255 to receive messages from any node.</p> <p><b>Destination Address Filter</b></p> <p>From the list, select the node destination of the expected message:</p>

# J1939 Receive Message

---

- global and specific  
Receive all messages for all types of destination nodes.
- global only  
Receive only broadcast messages.
- CA specific only  
Receive only messages sent to this node.

**Show 'New Message Received' output port**

Select this check box to create a New Message Received output port.

<b>Purpose</b>	J1939 Transmit
<b>Library</b>	xPC Target Library for CAN/J1939
<b>Description</b>	The J1939 Transmit Message block transmits a J1939 message. The J1939 database file defines the message type. You specify the J1939 database with the J1939 Database (CANdb) Setup block.
<b>Block Inputs</b>	<ul style="list-style-type: none"><li>• <b>Trigger</b> Enables the transmission of the message for that sample (a value of 1 indicates send, a value of 0 indicates do not send).</li><li>• <b>Signal Input(s)</b> Depending on the J1939 message defined in the J1939 database file, the block can have multiple input signal ports. If the bit length of a signal exceeds 32, the input is a byte array; otherwise, the block input data type is double.</li></ul>
<b>Block Outputs</b>	None
<b>Block Parameters</b>	<p><b>PGN</b> From the list, select the parameter group number. The contents of this list vary depending on the messages that the J1939 database file specifies.</p> <p><b>CA ID</b> Identifies the controller application that this block maps to. The ID must match the <b>CA ID</b> value of the corresponding J1939 Controller Application block.</p> <p><b>Priority (0-7; highest to lowest)</b> From a range of 0 to 7, specify the priority for the message transmission. 0 is the highest, 7 is the lowest.</p>

# J1939 Transmit Message

---

## **Destination Address (0–253 or 255)**

Specify the node address of the destination node. Type 255 for broadcast.

---

**Note** For message-oriented PGNs, this block ignores this parameter.

---

# Shared Memory Support

---

This topic describes implementations of reflective (shared) memory by various manufacturers.

- “GE Fanuc Embedded Systems Shared Memory” on page 17-2
- “GE Intelligent Platforms Shared Memory” on page 17-13
- “Curtiss-Wright Electronic Systems Shared Memory” on page 17-23
- “Boards and Blocks — Alphabetical List” on page 17-37

## GE Fanuc Embedded Systems Shared Memory

xPC Target supports reflective (shared) memory boards manufactured by GE Fanuc Embedded Systems (<http://www.gefanucembedded.com/>) – formerly VMIC.

In this section...
“Before You Start” on page 17-2
“Shared Memory Structure Reference” on page 17-5

### Before You Start

- “Create Shared Memory Partitions” on page 17-2
- “Initialize Shared Memory Nodes” on page 17-4

### Create Shared Memory Partitions

The xPC Target software uses a model for reflective memory (also known as shared memory) that includes Simulink blocks for shared memory driver functions and MATLAB structures to define node initialization and shared memory partitions. This topic describes the xPC Target support of the GE Fanuc Embedded Systems VMIPCI-5565 (formerly from VMIC) and PCI-5565PIORC boards. Both the VMIPCI-5565 and PCI-5565PIORC boards are fully supported and will be collectively referenced as PCI-5565 in the documentation. The xPC Target library supports these boards with the same set of blocks.

To use the xPC Target PCI-5565 shared memory blocks, you must define shared memory partition structures. A partition structure describes how you want to allocate (partition) the shared memory. The xPC Target software allocates shared memory as segments of data that are packed into memory regions or partitions. Along with the Shared Memory Pack and Shared Memory Unpack blocks, the following PCI-5565 blocks use shared memory partition structures:

- GE Fanuc 5565 read
- GE Fanuc 5565 write

After defining the shared memory partitions, you can add PCI-5565 shared memory driver blocks to your Simulink model. See “Shared Memory Partition Structure” on page 17-5 for the complete list of fields in a partition.

The following description refers to the `completepartitionstruct` command.  
Type

```
help completepartitionstruct
```

for more information.

- Create a partition structure in one of the following ways. Using the `completepartitionstruct` command at the MATLAB Command Window, create a default partition structure. For example, type

```
Partition = completepartitionstruct([], '5565')
```

```
Partition =
```

```
    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 a script file, partially define a structure, load that script 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 function extrapolates segment addresses from the first segment definition. If you have or require fragmented memory, use multiple partitions.
- The **Type** and **Size** fields are required for all segments 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 the preceding partition definition, the third segment (Partition(3)) has an alignment of '8'.
- The base addresses of subsequent segments are defined by the data type, size, and alignment of the preceding segment.
- Call the `completepartitionstruct()` command to fully populate the partition structure.

```
Partition = completepartitionstruct(Partition, '5565');
```

## **Initialize Shared Memory Nodes**

In addition to shared memory partitions, you must also define a node initialization structure before using the PCI-5565 shared memory blocks. A node initialization structure describes the shared memory partitions (see “Create Shared Memory Partitions” on page 17-2) and the PCI-5565 board configuration, including any interrupt settings if used. The following PCI-5565 block requires a shared memory node initialization structure.

- GE Fanuc 5565 init
- GE Fanuc 5565 read

After defining the node initialization structure, you can add PCI-5565 shared memory driver blocks to your Simulink model. See “Shared Memory Node Initialization Structure” on page 17-7 for the complete list of fields in a node initialization structure.

The following description refers to the `completenodeconstruct` command. Type

```
help completenodeconstruct
```

for more information.

- Create a node initialization structure in one of the following ways. 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]
```

- Now modify the structure fields to meet your requirements. For example:

```
node.Interface.NodeID = '128';
node.Partitions = Partition;
```

- A user-defined node structure, created with MATLAB code or from the MATLAB Command Window and supplement the resulting structure with a call to the `completenodeconstruct` function. For example:

```
node.Interface.NodeID = '128';
node.Partitions = Partition;
node.completenodeconstruct(node, '5565');
```

## Shared Memory Structure Reference

- “Shared Memory Partition Structure” on page 17-5
- “Shared Memory Node Initialization Structure” on page 17-7

## Shared Memory Partition Structure

You do not need to use all the fields of a partition initialization structure. However, knowing the possible structure fields will be helpful when you are setting up to use shared memory.

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>	<p>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.</p> <p>Align partition addresses on 32-bit word boundaries (for example, 0x0, 0x4, 0x8, and so forth).</p>
<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>Partition Fields</b>	<b>Description</b>
<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 <b>Size</b> entry as the specification of the length of a non-oriented array or vector</li> <li>• [m,n] — Treats the <b>Size</b> 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

<b>Node Structure Fields</b>	<b>Description</b>
<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>• <b>Mode</b> — Configures board registers (see “Board Mode” on page 17-8)</li> </ul>

Node Structure Fields	Description
	<ul style="list-style-type: none"> <li>• <b>Interrupts</b> — Enables the board to generate PCI interrupts from network events that have been broadcast from other nodes, or in response to error conditions (see “Board Interrupts” on page 17-9)</li> <li>• <b>NodeID</b> — Specifies the node ID for the board (see “Board Node ID” on page 17-11)</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 17-2)

**Board Mode.** The PCI-5565 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 PCI-5565 board setting of the LSR1 (Local Control and Status Register 1) and LIER (Local Interrupt Enable Register) registers. Refer to the PCI-5565 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 GE Fanuc 5565 read or GE Fanuc 5565 write block.

Of particular note are the following modes:

<b>Board Modes</b>	<b>Description</b>
<b>StatusLEDOff</b>	Turns the PCI-5565 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 software 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 PCI-5565 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, in the format 'sizeMByte'.The PCI-5565 driver checks this value against the memory size of the PCI-5565 board. If you enter a size in this field that is larger than the actual PCI-5565 board memory size, the driver will return an error.

This table lists the values for MemoryOffset:

<b>Value</b>	<b>Offset Produced</b>
'0'	0
'1'	0x4000000
'2'	0x8000000
'3'	0xC000000

**Board Interrupts.** The PCI-5565 board can generate PCI interrupts in response to network events that have been broadcast from other nodes, or in response to 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 PCI-5565 board. Each bit enables the specified interrupt source on the PCI-5565 board. Refer to the PCI-5565 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 PCI-5565 board. See **GE Fanuc 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 the xPC Target software knows about.

- 2 Find the IRQ specified for the PCI-5565 board.

This is the interrupt source number you need to specify in the **Real-Time interrupt source** field of the **xPC target options** pane.

- 3 Edit your script 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 Select **Simulation > Model Configuration Parameters**

- 6 Select node **Code Generation**.

- 7 Select node **xPC Target options**.

- 8 Set the **Execution mode** field to Real-Time.

- 9 Click the **Real-Time interrupt source** list.

- 10 Select the interrupt source number to which the board is set.

- 11 Click the **I/O board generating the interrupt** list and select GE\_Fanuc (VMIC)\_PCI-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 PCI-5565 board specify the board node ID. Correspondingly, you can also configure the PCI-5565 block with the board node ID using the `Interface.NodeID` field. Enter values according to the following:

<b>NodeID Value</b>	<b>Description</b>
'any'	Allows the PCI-5565 driver to work with any PCI-5565 node regardless of the PCI-5565 board node ID jumper setting
value from '0' to '255'	Specifies the particular PCI-5565 node that the driver must look for. If this value does not match the jumpered value on the PCI-5565 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 PCI-5565 boards in your system and you want to identify the driver for a particular node.

## GE Intelligent Platforms Shared Memory

xPC Target supports high-speed fiber optic reflective memory boards manufactured by GE Intelligent Platforms (<http://defense.ge-ip.com/products/family/avionics>) – formerly SBS Technologies.

In this section...
“Broadcast Memory” on page 17-13
“Shared Memory Structure Reference” on page 17-18

### Broadcast Memory

- “Introduction” on page 17-13
- “Create Shared Memory Partitions” on page 17-13
- “Initialize Shared Memory Nodes” on page 17-16

### Introduction

The xPC Target software 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.

---

**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 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. The xPC Target software

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 17-18 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 a script file, partially define a structure, load that script 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 17-13) 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 17-20 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 a script file, partially define a structure, load that script 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**.

## Shared Memory Structure Reference

- “Shared Memory Partition Structure” on page 17-18
- “Shared Memory Node Initialization Structure” on page 17-20

### Shared Memory Partition Structure

You do not need to use all the fields of a partition initialization structure. However, knowing the possible structure fields will be helpful when you are creating any of the structures.

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>Type</b>	Specifies the data type of the memory segment. Specify one of the following types: <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> </ul>

<b>Partition Fields</b>	<b>Description</b>
	<ul style="list-style-type: none"> <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 <b>Size</b> entry as the specification of the length of a non-oriented array or vector</li> <li>• [m,n] — Treats the <b>Size</b> 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 the first address of this segment.</p>
<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

You do not need to use all the fields of a node initialization structure. However, knowing the possible structure fields will be helpful when you are creating any of the structures.

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>	Specifies if the xPC Target software can operate without having the shared memory network fully operational. <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 the xPC Target software to operate without having the shared memory network fully operational.</li> </ul>
<b>Memory Size</b>	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

Board Modes	Description
	<p>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 the xPC Target software 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 **Real-Time interrupt source** field of the **xPC target options** pane.

**4** From the MATLAB Command Window, type the name of your Simulink model.

The Simulink model appears.

**5** Select **Simulation > Model Configuration Parameters**

**6** Select node **Code Generation**.

**7** Select node **xPC Target options**.

**8** Set the **Execution mode** field to **Real-Time**.

**9** Click the **Real-Time interrupt source** list.

**10** Select the interrupt source number to which the board is set.

**11** Click the **I/O board generating the interrupt** list and select the SBS board ID associated with the IRQ:

```
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 (-1: autosearch) or ISA base address** parameter, specify the PCI slot that contains the board you are configuring for interrupts.

**13** Click **OK**.

# Curtiss-Wright Electronic Systems Shared Memory

xPC Target supports reflective (shared) memory boards manufactured by Curtiss-Wright Electronic Systems ([http://www.cwcelectronicssystem.com/data\\_communications.html](http://www.cwcelectronicssystem.com/data_communications.html)) – formerly Systran.

## In this section...

“Before You Start” on page 17-23

“Systran Shared Memory Structure Reference” on page 17-26

## Before You Start

- “Create Shared Memory Partitions” on page 17-23
- “Initialize Shared Memory Nodes” on page 17-25

## Create Shared Memory Partitions

The xPC Target software uses a model for reflective memory (also known as shared memory) that includes Simulink blocks for shared memory driver functions and MATLAB structures to define node initialization and shared memory partitions. This topic describes xPC Target support of the Systran SCRAMNet+ SC150 board.

To use the xPC Target Systran SCRAMNet+ SC150 shared memory blocks, you must define shared memory partition structures. A partition structure describes how you want to allocate (partition) the shared memory. The xPC Target software allocates shared memory as segments of data that are packed into memory regions or partitions. Along with the Shared Memory Pack and Shared Memory Unpack blocks, the following SCRAMNet+ SC150 blocks use shared memory partition structures:

- Systran SC150 read
- Systran SC150 write
- Systran SC150 rearm

After defining the shared memory partitions, you can add SCRAMNet+ SC150 shared memory driver blocks to your Simulink model. See “Shared Memory Partition Structure” on page 17-26 for the complete list of fields in a partition structure.

The following description refers to the `completepartitionstruct` command. Type

```
help completepartitionstruct
```

for more information.

- Create a partition structure in one of the following ways. Using the `completepartitionstruct` command at the MATLAB Command Window, create a default partition structure. For example, type

```
Partition = completepartitionstruct([], 'scramnet')
```

```
Partition =  
    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 a script file, partially define a structure, load that script 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 function extrapolates segment addresses from the first segment definition. If you have or require fragmented memory, use multiple partitions.
- The **Type** and **Size** fields are required for all segments in the partition structure.
- The **Alignment** value is optional. It is '4' by default. This value forces segments that do not have alignment specifications to start on 4 byte (32-bit) boundaries.
- The base addresses of subsequent segments are defined by the data type, size, and alignment of the preceding segment.
- Call the `completestruct()` command to fully populate the partition structure.

```
Partition = completestruct(Partition, 'scramnet');
```

## Initialize Shared Memory Nodes

In addition to shared memory partitions, you must also define a node initialization structure before using the SCRAMNet+ SC150 shared memory blocks. A node initialization structure describes the shared memory partitions (see “Create Shared Memory Partitions” on page 17-23) and the SCRAMNet+ SC150 board configuration, including any interrupt settings if used. The Systran SC150 init block requires a shared memory node initialization structure.

After defining the node initialization structure, you can add SCRAMNet+ SC150 shared memory driver blocks to your Simulink model. See “Shared Memory Node Initialization Structure” on page 17-32 for the complete list of fields in a node initialization.

The following description refers to the `completenodeconstruct` command. Type

```
help completenodeconstruct
```

for more information.

- Create a node initialization structure in one of the following ways. Using the `completenodeconstruct` 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]
```

- Now modify the structure fields to meet your requirements. For example:

```
node.interface.NodeID = '128';  
node.Partitions = Partition;
```

- A user-defined node structure, created with MATLAB code or from the MATLAB Command Window and supplement the resulting structure with a call to the `completenodeconstruct` function. For example:

```
node.Interface.NodeID = '128';  
node.Partitions = Partition;  
node = completenodeconstruct(node, 'scramnet');
```

## **Systran Shared Memory Structure Reference**

- “Shared Memory Partition Structure” on page 17-26
- “Shared Memory Node Initialization Structure” on page 17-32

### **Shared Memory Partition Structure**

A shared memory partition structure has the following fields. 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 setting up to use shared memory.

```

Address: '0x0'
Type: 'uint32'
Size: '1'
Alignment: '4'
RIE: 'off'
TIE: 'off'
ExtTrigger1: 'off'
ExtTrigger2: 'off'
HIPRO: 'off'
Internal: [1x1 struct]

```

where

<b>Partition Fields</b>	<b>Description</b>
<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>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>Partition Fields</b>	<b>Description</b>
<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>Specifies the byte alignment of the next partition (if one is defined). Enter alignment value in bytes: 1, 2, 3, 4. The default value is '4', forcing a double word boundary alignment for all elements. See “Alignment Examples” on page 17-30.</p>
<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>Partition Fields</b>	<b>Description</b>
<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>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>Partition Fields</b>	<b>Description</b>
<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>
<b>HIPRO</b>	<p>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'.</p> <p>'off' — Prevents the partition from transmitting the elements as one message</p> <p>'on' — Allows the partition to transmit the elements as one message</p>
<b>Internal</b>	Reserved for internal use.

**Alignment Examples.** This example shows the shared memory map with default alignment values.

```
Partition1(1).Type='int32';
Partition1(1).Size='1';
```

```
Partition1(2).Type='boolean';
Partition1(2).Size='1';
```

```
Partition1(3).Type='uint32';
```

```
Partition1(3).Size='1';
Partition1 = completepartitionstruct(Partition1,'scramnet');
```



This example shows the shared memory map with alignment value changed from 4 to 1 in the second partition.

```
Partition1(1).Type='int32';
Partition1(1).Size='1';
Partition1(1).Alignment='4';
```

```
Partition1(2).Type='boolean';
Partition1(2).Size='1';
Partition1(2).Alignment='1';
```

```
Partition1(3).Type='uint32';
Partition1(3).Size='1';
Partition1 = completepartitionstruct(Partition1,'scramnet');
```



This example shows the shared memory map with alignment value changed from 4 to 2 in the second partition.

```
Partition1(1).Type='int32';
Partition1(1).Size='1';
Partition1(1).Alignment='4';
```

```
Partition1(2).Type='boolean';
Partition1(2).Size='1';
Partition1(2).Alignment='2';
```

```
Partition1(3).Type='uint32';
Partition1(3).Size='1';
Partition1 = completepartitionstruct(Partition1,'scramnet');
```

1	1	1	1	2		3	3	3	3				
---	---	---	---	---	--	---	---	---	---	--	--	--	--

### 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>	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. <ul style="list-style-type: none"> <li>• <b>Mode</b> — Configures board modes (see “Board Mode” on page 17-33)</li> <li>• <b>Timeout</b> — Enables the board to set the timeout value (see “Board Timeout” on page 17-34)</li> <li>• <b>DataFilter</b> — Controls the data filtering operation (see “Board Data Filter” on page 17-34)</li> <li>• <b>VirtualPaging</b> — Controls the board virtual paging operation (see “Virtual Paging” on page 17-35)</li> <li>• <b>Interrupts</b> — Enables the board to generate and receive interrupts from the network (see “Board Interrupts” on page 17-35)</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 17-23)

**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]

Field	Values	Default	CSR
<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
Enable TransmitData Filter	'off', 'on'	'off'	CSR0[10]
EnableLower4KBytesFor DataFilter	'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
VirtualPagingEnable	'off', 'on'	'off'	CSR12[0]
VirtualPage Number	'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:

<b>Field</b>	<b>Values</b>	<b>Default</b>	<b>CSR</b>
<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

GE Fanuc 5565 broadcast	GE Fanuc PCI-5565 broadcast interrupt block
GE Fanuc 5565 init	GE Fanuc PCI-5565 initialization block
GE Fanuc 5565 read	GE Fanuc PCI-5565 read block
GE Fanuc 5565 write	GE Fanuc PCI-5565 write block
SBS25x0 init	SBS25x0 init block
SBS25x0 read	SBS25x0 read block
SBS25x0 write	SBS25x0 write block
Systran SC150 init	SystranSC150 init block
Systran SC150 init	SystranSC150 init block
Systran SC150 read	Systran SC150 read block
Systran SC150 read	Systran SC150 read block
Systran SC150 rearm	Systran SC150 rearm block
Systran SC150 rearm	Systran SC150 rearm block
Systran SC150 write	Systran SC150 write block
Systran SC150 write	Systran SC150 write block



# Video Image Processing

---

- “Processing Video Images Using xPC Target” on page 18-2
- “Acquiring Images from Camera Link Cameras” on page 18-5
- “Boards and Blocks — Alphabetical List” on page 18-15

## Processing Video Images Using xPC Target

In this section...
“xPC Target Video Basics” on page 18-2
“Before You Start” on page 18-2
“Basic Workflow” on page 18-3

### xPC Target Video Basics

The xPC Target software supports webcams compliant with the USB Video Class (UVC) standard and cameras compliant with the Automated Imaging Association Camera Link® standard.

---

**Note** UVC compliant cameras are often referred to as "driverless webcams". For more information, see your camera documentation.

---

The xPC Target Video library contains blocks that support the following real-time video image applications:

- Acquire real-time video frames from cameras connected to the target computer.
- Select a region of interest in the real-time image.
- Compress video frames acquired on the target computer.
- Stream video frames acquired on the target computer to the host computer.
- Decompress video frames on the host computer.

You can then process and display video frames on the host computer using blocks from the Computer Vision System Toolbox™ or Image Processing Toolbox™.

### Before You Start

To acquire video frames, buy one of the following camera types:

- USB Video Class (UVC) compliant webcam — Use the From USB Video Device block, available in the Video/USB library.

---

**Tip** Enable USB general support through the BIOS.

---

- Camera Link compliant camera — Use the NEON BitFlow Image Input block, available in the xPC Target for BitFlow library.

## Basic Workflow

On the target computer, you can do the following:

- Acquire images.
- Process or reduce images.
- Compress images.
- Transmit images to the host computer.

On the host computer, you can do the following:

- Receive images from the target computer.
- Decompress images.
- Process or reduce images.
- Display images.

For example, the following workflow acquires and compresses video frames on the target computer, transmits them to the host computer, and decompresses and displays them:

- 1 Connect a USB webcam or Camera Link camera to the target computer.
- 2 Add the image input block (From USB Video Device or NEON BitFlow Image Input) for the camera to the target-side model. Configure the block as required.

---

**Note** If you add a From USB Video Device block, you can optionally add the USB Video Device List block to query the available configurations for the attached camera.

---

- 3 Add the JPEG Compression block to the target-side model. Connect this block to the output of the image input block.

---

**Note** Some USB cameras support on-chip JPEG compression. For these cameras, you do not need to use an additional JPEG Compression block.

---

- 4 Add the Image Transmit block to the target-side model. Connect this block to the output of the compression block. Configure the block to transmit frames to the host computer.
- 5 Add the Image Receive block to the host-side model. Configure the block to receive frames from the target computer.
- 6 Add the JPEG Decompression block to the host-side model. Connect this block to the output of the Image Receive block.
- 7 Add the To Video Display block from the Computer Vision System Toolbox to the host-side model.
- 8 Build and download the target-side model to the target computer.
- 9 Run the host-side model on the host computer.

# Acquiring Images from Camera Link Cameras

## Introduction

You can use the NEON BitFlow Image Input block to acquire images from cameras that support the Automated Imaging Association Camera Link interface standard. This block supports the BitFlow Neon-CLB board (see <http://www.bitflow.com>).

This topic assumes that you are familiar with the following:

- Video capture procedures
- Camera Link interface standard
- Manufacturer's instructions for installing the BitFlow Neon-CLB board on target computer
- Manufacturer's instructions for installing the BitFlow SDK on host computer
- Serial command codes for your particular camera model

For more information on image processing in xPC Target, see “Basic Workflow” on page 18-3.

## Before You Start

To acquire images from cameras that support the Camera Link standard:

- Install the BitFlow Neon-CLB board in a PCI Express x4 or wider slot in the target computer.
- Install the BitFlow SDK on the host computer.
- Acquire a monochrome digital camera with:
  - One or two taps
  - Free-running or triggered configuration file in BitFlow SDK
- Use the Image Transmit and Image Receive blocks to transfer images from the target computer to the host computer for display or processing.

---

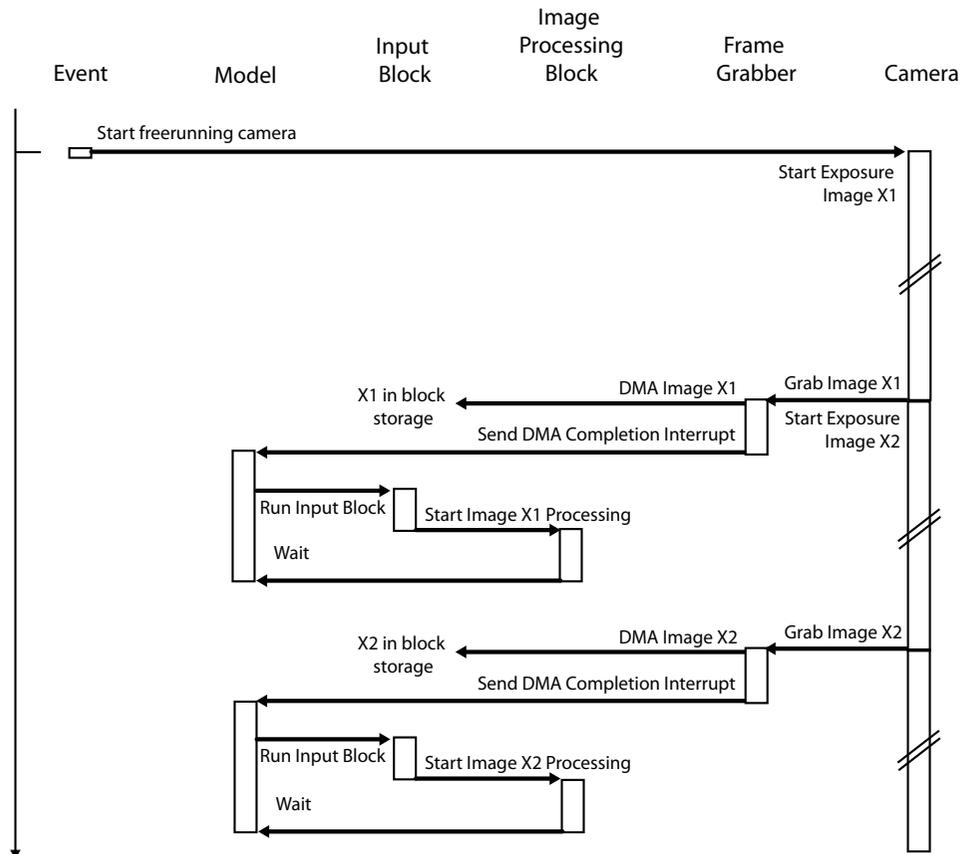
### **Note**

- The BitFlow Neon-CLB board comes in two hardware configurations, one with differential triggers and one with TTL/Opto triggers. You must decide on a trigger method before acquiring the board.
  - The NEON BitFlow Image Input block requires the BitFlow SDK to configure the BitFlow Neon-CLB board. If the SDK does not contain a configuration file for your specific camera, contact your BitFlow representative to acquire one. You cannot proceed without this file.
-

## Camera Triggering

The NEON BitFlow Image Input block interacts with the frame grabber and camera using four trigger modes set in the camera configuration file:

- **Free Running.** The camera and frame grabber run continuously. When the camera finishes an exposure, the frame grabber board copies the image to target memory using DMA and signals a frame completion interrupt. Upon receiving the interrupt, the model runs and performs image processing.



---

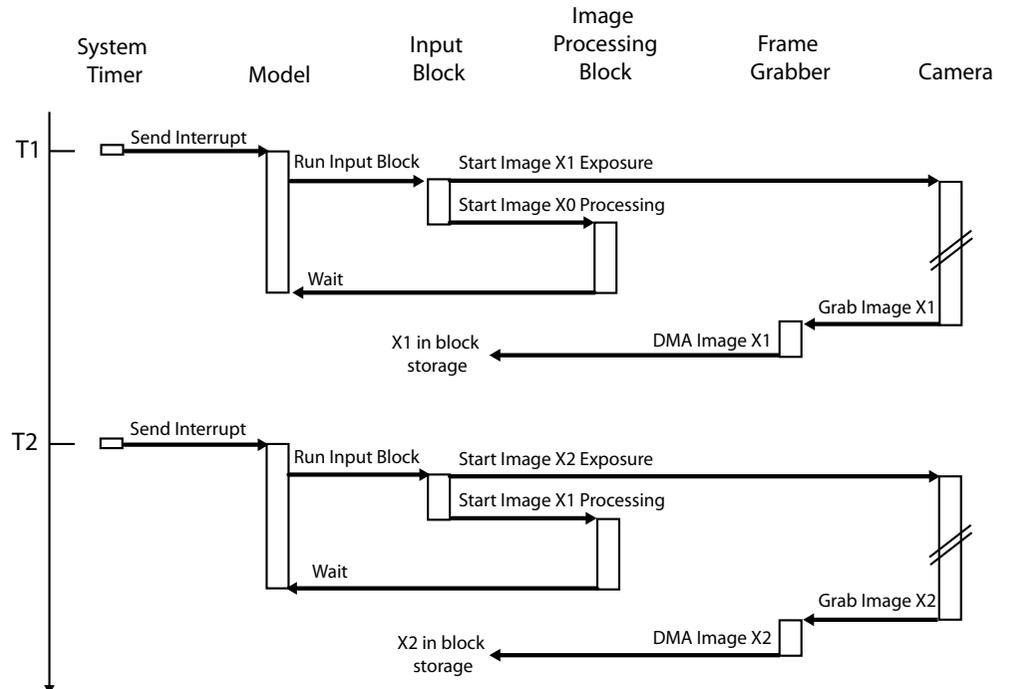
**Note** The model must complete image processing before the next DMA completion interrupt. Otherwise, the CPU overloads.

---

If heavy image processing is required, create a multirate, multitasking model that samples every  $N$ th image. Place the image input block at the top level with a sample time of  $T_s$ . Construct an atomic subsystem for image processing and assign it a sample time of  $T_s * 2$  (or whatever factor  $N$  allows the subsystem to sample every  $N$ th image). Connect the output of the image input block to the input port of the subsystem through a rate transition block.

The example Image Capture with Camera Link® and Bitflow™ Neon-CLB Frame Grabber shows such a model, with the subsystem running at  $T_s * 3$ , or sampling every 3rd image. The model is configured for a large format camera where the JPEG compression and image send block take more than  $T_s$  to complete.

- **Timer Triggered.** The model starts from the target computer system timer. When the model runs, it simultaneously starts a camera exposure and begins to process the image captured during the previous time step. When the camera finishes the exposure, the frame grabber copies the image to target memory using DMA for processing in the next time step.

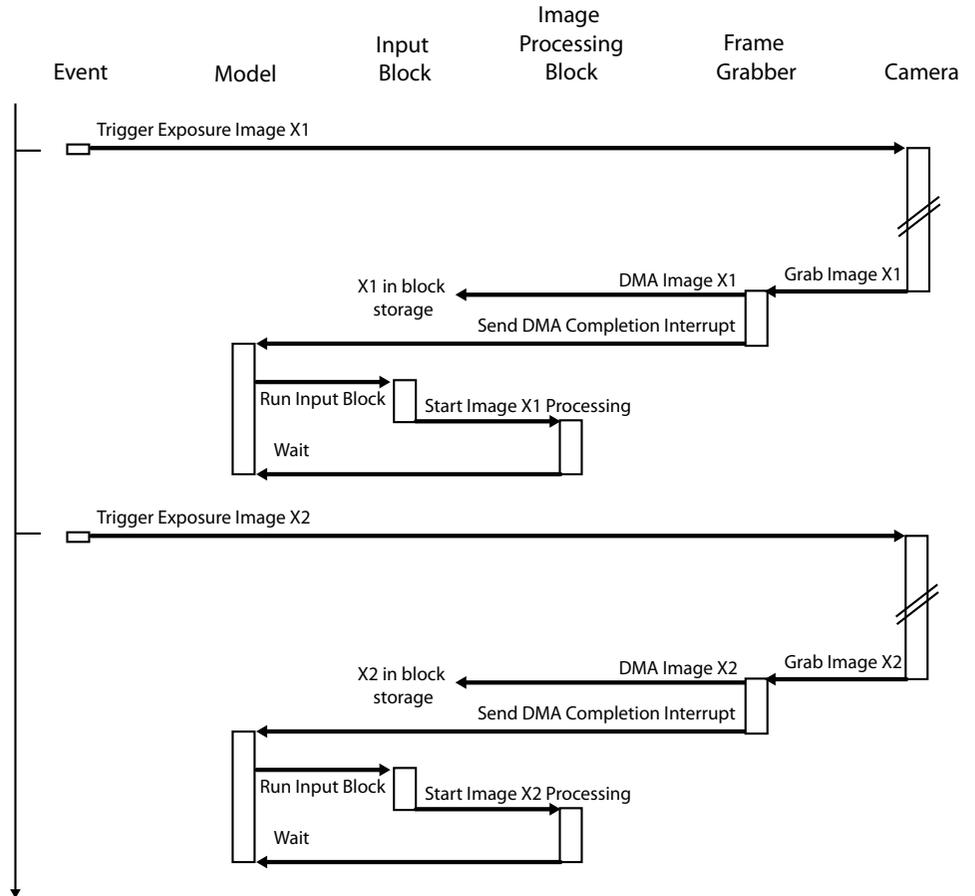


- **Differential Trigger.** The camera triggers from a rising edge on the differential trigger input. When the camera finishes the exposure, the frame grabber copies the image to target memory using DMA and signals a frame completion interrupt. Upon receiving the interrupt, the model runs and performs image processing.

---

**Tip** All Camera Link cameras respond to one of four camera control lines (CC1 to CC4) coming from the frame grabber. The required line is given in the triggered-mode configuration file for the camera. Connect the external trigger to the frame grabber trigger input.

---



- **TTL Trigger.** The camera triggers from a rising edge on the TTL trigger input. The model runs from the frame completion interrupt, as with Differential Trigger.
- **Opto Trigger.** The camera triggers from a rising edge on the Opto trigger input. The model runs from the frame completion interrupt, as with Differential Trigger.

---

**Note**

- The camera configuration file puts the frame grabber into triggered mode, but does not put the camera itself into that mode. You must send a command to the camera to set it to triggered mode. See “Serial Camera Configuration” on page 18-12.
  - In some cases, you can use the block mask to change the trigger mode and trigger source selected by the camera configuration file. In most cases, however, you cannot use the block mask to turn triggering on or off. If you need a mode that cannot be set from the block mask, contact BitFlow for a camera configuration file for the mode you want.
- 

**Serial Camera Configuration**

To enter a serial command for the camera, enter it without quotation marks in the **Camera configuration serial command** text box.

---

**Tip** Enter multiple commands as one line without white space.

---

Each camera manufacturer has a serial command set documented in the manual for each camera or family of cameras. Each command set is completely different from any other. For example, to put a camera from the given series into triggered mode and set the shutter time to 1 ms, you might use these mask settings and serial command strings:

Camera Series	Mask Settings	Serial Setup
CIS VCC G21	9600 baud 8 bits No parity	000000W004001\r 000000W002005\r
Pulnix TM1400	9600 baud 8 bits No parity	:SA5\r

Camera Series	Mask Settings	Serial Setup
Sony XCL	38400 baud 8 bits No parity	SHUTTER 7\r TRG_MODE 1\r
Cohu 7800	9600 baud 8 bits No parity	\002\0377cE2,10chk\003 \002\0377cM2chk\003 \002\0377T0,1chk\003 where <i>chk</i> is a checksum for the given portion of the message.

---

**Note** You must find the required command codes in the manufacturer documentation for your specific camera. If the codes you need are not in the manual, contact the camera manufacturer.

---

Some cameras return a success or failure value in response to the command. The image input block dialog box includes a check box to allow the software to display such values. Some responses include non-printable control characters, displayed as C escape sequences. For example:

Escape Sequence	Control Character	Definition
\r	CR	Carriage Return
\002	STX	Start of Text
\003	ETX	End of Text
\006	ACK	Acknowledge
\025	NACK	Negative Acknowledge

## Offline Camera Configuration

Many camera manufacturers offer Windows utilities that can be used to send configuration strings to their cameras. Using such a utility, you can configure

a camera on Windows, save the settings on the camera, then connect the camera to the target computer and use it.

Other camera manufacturers support a special serial cable that connects the camera's serial communication line to a separate DB9 connector. You can connect the DB9 to a serial port on the host computer, configure the camera via the serial cable, and capture images via the BitFlow Neon-CLB board on the target computer.

---

**Note** If you use a separate serial cable to send configuration strings to the camera, settings in the **Camera configuration serial command** section have no effect.

---

### **BitFlow Neon-CLB Example**

The xPC Target product contains the following example that illustrates how to acquire images from a camera connected to the BitFlow Neon-CLB block:

Image Capture with Camera Link and Bitflow Neon-CLB Frame Grabber

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

# BitFlow Neon-CLB

---

**Purpose** Support for BitFlow Neon-CLB camera interface board

**Board** BitFlow Neon-CLB

**General Description** The BitFlow Neon-CLB board has:

- One camera port that supports one Camera Link based camera with power over Camera Link (PoCL)
- SafePower, which protects against Camera Link power line faults

The BitFlow Neon-CLB board supports one-tap and two-tap cameras and requires a PCI Express x4 or wider slot. The xPC Target block library supports this board with the NEON BitFlow Image Input block.

---

**Note** The BitFlow Neon-CLB board comes in two hardware configurations, one with differential trigger input and one with TTL/Opto trigger inputs. You must decide on a trigger method before acquiring the board.

---

## Board Characteristics

Board name	BitFlow Neon-CLB
Manufacturer	BitFlow
Bus type	PCI Express
Multiple block instance support	No
Multiple board support	No

**Purpose**

Support for NEON BitFlow Image Input block

**Library**

xPC Target for BitFlow

**Block Parameters****Camera config file**

Click the **Browse** button to select a configuration from the BitFlow SDK installation folder.

The browser window displays a list of camera configuration files for BitFlow R64-type camera boards. The BitFlow Neon-CLB supports 1-tap and 2-tap cameras from this list. Scroll and open the camera configuration file specified by you and your BitFlow representative— for example, `CIS-VCC-G21V31CL-E1-FreeRun.r64`. Alternatively, start typing the manufacturer name, and an abbreviated list of camera configuration files appears for that manufacturer. Open the BitFlow camera configuration file for your camera model.

When you open the configuration file, it assigns values to the noneditable parameters:

**Manufacturer**

Displays camera manufacturer name.

**Model**

Displays the model of the camera that the configuration file configures and works with.

**Mode**

Describes the camera operation mode.

**Bit Depth**

Displays the number of bits per pixel that the camera uses.

**Acquisition Timeout**

Displays the number of milliseconds before the board sends an interrupt. For example, the board might emit an interrupt if the camera is not on.

# NEON BitFlow Image Input

---

## **Full Width**

Displays the width of the incoming image, in pixels.

## **Full Height**

Displays the height of the incoming image, in lines.

## **Trigger Mode**

From the Trigger Mode list, select one of:

- **Free Running.** The camera and frame grabber run continuously. When the camera finishes an exposure, the frame grabber board copies the image to target memory using DMA and signals a frame completion interrupt. Upon receiving the interrupt, the model runs and performs image processing.
- **Timer Triggered.** The model starts from the target computer system timer. When the model runs, it simultaneously starts a camera exposure and begins to process the image captured during the previous time step. When the camera finishes the exposure, the frame grabber copies the image to target memory using DMA for processing in the next time step.
- **Differential Trigger.** The camera triggers from a rising edge on the differential trigger input. When the camera finishes the exposure, the frame grabber copies the image to target memory using DMA and signals a frame completion interrupt. Upon receiving the interrupt, the model runs and performs image processing.
- **TTL Trigger.** The camera triggers from a rising edge on the TTL trigger input. The model runs from the frame completion interrupt, as with **Differential Trigger**.
- **Opto Trigger.** The camera triggers from a rising edge on the Opto trigger input. The model runs from the frame completion interrupt, as with **Differential Trigger**.

---

## Note

- For more on triggering, see “Camera Triggering” on page 18-7.
  - The BitFlow Neon-CLB board comes in two hardware configurations, one with differential triggers and one with TTL/Opto triggers. You must decide on a trigger method before acquiring the board.
  - In some cases, you can use the block mask to change the trigger mode and trigger source selected by the camera configuration file. However, in most cases you cannot use the block mask to turn triggering on or off. If you need a mode that cannot be set from the block mask, contact BitFlow for a camera configuration file for the mode you want.
- 

## Acquisition Region of Interest

Select a rectangular region of the full image to output from the block. When you select a new camera configuration file, these values will be set to the full size of the image specified in the configuration file.

Column 0, row 0 is in the upper left corner of the image.

### Starting Column

Column numbers are 0 based.

### Starting Row

Row numbers are 0 based.

### Columns

The number of columns plus the starting column must be less than the full image width.

### Rows

The number of rows plus the starting row must be less than the full image height.

# NEON BitFlow Image Input

---

## **Camera configuration serial command**

Select the serial port configuration and specify a command to be sent to the camera when the model starts executing.

### **Baud Rate**

Select one of 9600, 19200, 38400, 57600, 115200, 230400, or 460800 baud.

### **Bits per Byte**

Select one of 5, 6, 7, or 8. Most camera configurations will use 8 bits per byte.

### **Parity**

Select one of None, Even, Odd, Mark, or Space.

### **Stop Bits**

Select 1 or 2.

### **Command String**

Enter a command string to send to your camera. The exact format of the string depends on the camera.

### **Display camera response string**

Select this check box to display the camera's response to the command string. Nonprintable control codes are printed as C escape sequences.

---

**Note** For more on camera configuration, see "Serial Camera Configuration" on page 18-12.

---

## **Camera Configuration comments**

Displays comments from BitFlow, Inc., about this configuration.

## **Sample time**

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

If you configure the target application to use the board interrupt, the application adds the sample time to the displayed execution time.

## **PCI slot (-1: autosearch)**

If only one board of this type is in the target computer, enter -1 to automatically locate the board.

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

```
getxpcpci
```

# From USB Video Device

---

**Purpose** From USB Video Device block

**Library** xPC Target Library for USB Camera

**Description** The From USB Video Device block enables you to acquire real-time video frames or still images from a USB Video Class (UVC) webcam attached to a USB port on the target computer. Perform the following on the acquired image signal:

- Stream captured frames to the host computer display (for example, with the Video Display block from the Computer Vision System Toolbox).
- Analyze the image signals on the host computer.
- Compress or decompress the input signal with the JPEG Compression or JPEG Decompression blocks.

When you add this block, also add the USB Video Device List block to help configure the webcam.

## Block Parameters

### Configuration

From the list, select a configuration that you specified in the USB Video Device List block. When you click the **Reload Device List** button on the USB Video Device List block, this configuration list is updated.

### Port address (-1 for any)

Specify the port to which the webcam is attached. Enter -1 for any USB port.

### Image width

Enter the width of the image input from the USB port, in pixels.

### Image height

Enter the height of the image input from the USB port, in pixels.

## Frame interval

From the list, select the desired sample time between frame transfers:

- 1/60
- 1/30
- 1/25
- 1/20
- 1/15
- 1/10
- 1/7.5
- 1/5

## Frame format

From the list, specify if the incoming frames are to be compressed:

- Uncompressed

Do not compress frames.

- MJPEG

Compress frames using Motion JPEG format, where each frame is individually compressed as a JPEG image. Selecting this option disables the **Color format** and **Image signal** parameters.

## Color format

From the list, specify the color format for the incoming frames:

- RGB24 (8:8:8)

Output frames using RGB24 color encoding.

- YCbCr (4:2:2)

Output frames using YCbCr color encoding.

# From USB Video Device

---

## **Image signal**

From the list, select the type of image signal the block outputs:

- One multidimensional signal

One signal where each dimension contains color information. Selecting this option creates one output port, Image.

- Separate color signals

Multiple color signals where each signal contains the information for one color. Selecting this option creates three output ports: R, G, and B for the red, green, and blue signals (or Y, Cb, or Cr for YCbCr colorspace).

## **Show trigger input**

Select this check box to display an input port, Trigger, for the block.

## **Show length output**

Select this check box to display an output port, Length, for the block.

## **See Also**

USB Video Device List

<b>Purpose</b>	Image Receive block
<b>Library</b>	xPC Target Library for Video Utilities
<b>Description</b>	The Image Receive block receives the video from the target computer located at the IP address and port pair.
<b>Block Parameters</b>	<p><b>IP address to receive from (0.0.0.0 for accepting all)</b> Enter a valid IP address as a dotted decimal string for the source address, for example, 10.10.10.3. You can also use a MATLAB expression that returns a valid IP address as a string.</p> <p>The default address, 0.0.0.0, enables the acceptance of all frames from any computer. If set to a specific IP address, packets arriving from only that IP address are received.</p> <p>The <b>IP port to receive from</b> parameter specifies the port for the source.</p> <p><b>IP port to receive from</b> Specify the port of the PC from which to receive the video frames. The <b>IP address to receive from (0.0.0.0 for accepting all)</b> parameter specifies the IP address for the source.</p> <p><b>Use the following local IP port (-1 for automatic port assignment)</b> Specify the port of the PC receiving the video frames.</p> <p>Enter -1 to automatically assign a port for the PC.</p> <p><b>Allow variable length packets</b> Select this check box to enable the reception of variable-length frames. For example, use this option for compressed frames, because the length of each frame varies.</p> <p>If this check box is selected, the port sends the actual data. If the port size is less than the size of the actual data, the block sends</p>

# Image Receive

---

up to the allowed length of data. Selecting this block creates a Length input port.

If this check box is cleared (default), the block receives only fixed-length packets.

## **Sample time**

Enter the base sample time or a multiple of the base sample time

## **See Also**

Image Transmit

<b>Purpose</b>	Image Transmit block
<b>Library</b>	xPC Target Library for Video Utilities
<b>Description</b>	The Image Transmit block sends the video frame from the target computer to the host computer located at the IP address and port pair.
<b>Block Parameters</b>	<p><b>IP address sent to (255.255.255.255 for broadcast)</b> Specify the IP address of the PC to which you send the video frames. To broadcast the video frames to all listening computers, enter 255.255.255.255. The <b>Remote IP port to send to</b> parameter specifies the port for the destination.</p> <p><b>Remote IP port to send to</b> Specify the PC port to which you send the video frames. The <b>IP address sent to (255.255.255.255 for broadcast)</b> parameter specifies the IP address for the destination.</p> <p><b>Use the following local IP port (-1 for automatic port assignment)</b> Specify the target computer port from which to send the video frames.  Enter -1 to automatically assign a port for the target computer.</p> <p><b>Allow variable length packets</b> Select this check box to enable the transmission of variable-length frames. For example, use this option for compressed frames, because the length of each frame varies.  If this check box is selected, the port sends the actual data. If the port size is less than the size of the actual data, the block sends up to the allowed length of data. Selecting this block creates a Length input port.  If this check box is cleared (default), the block sends only fixed-length packets.</p>

# Image Transmit

---

## Sample time

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

## See Also

Image Receive

<b>Purpose</b>	JPEG Compression block
<b>Library</b>	xPC Target Library for Video Utilities
<b>Description</b>	The JPEG Compression block compresses the video frame received by the target computer.
<b>Block Parameters</b>	<p><b>Compression quality (-1:default)</b> Enter a value between 0 and 100 to specify how much to compress the incoming video frame. The lower the value, the less the compression quality.</p> <p>Enter -1 to use the default compression quality for the video frame.</p> <p><b>Input colorspace</b> From the list, select the color space scheme for the input video frame.</p> <ul style="list-style-type: none"><li>• Grayscale Compress the video frame using a grayscale color space scheme.</li><li>• YCbCr 4:4:4 Compress the video frame using the YCbCr color space scheme.</li><li>• RGB Compress the video frame using the red, blue, green (RGB) color space scheme.</li></ul> <p><b>Image signal</b> From the list, select the type of frame signal that the block outputs:</p> <ul style="list-style-type: none"><li>• One multidimensional signal One signal where each dimension contains color information. Selecting this option creates one output port, Image.</li><li>• Separate color signals</li></ul>

# JPEG Compression

---

Multiple color signals where each signal contains the information for one color. Selecting this option creates three output ports: R, G, and B for the red, green, and blue signals (or Y, CB, or Cr for YCbCr colorspace).

**Max output image size (bytes)**

Enter the maximum output size for the compressed video frame, in bytes. Use format *height \* width*.

**Show output image length**

Select this check box to output the video frame length. Selecting this check box displays the Length port.

**See Also**

JPEG Decompression

<b>Purpose</b>	JPEG Decompression block
<b>Library</b>	xPC Target Library for Video Utilities
<b>Description</b>	The JPEG Decompression block decompresses the video frame received by the target computer.
<b>Block Parameters</b>	<p><b>Image width</b> Specify the width, in bytes, of the video frame to be decompressed.</p> <p><b>Image height</b> Specify the height, in bytes, of the video frame to be decompressed.</p> <p><b>Output colorspace</b> From the list, select the color space scheme for the output video frame.</p> <ul style="list-style-type: none"><li>• Grayscale Compress the image using a grayscale color space scheme.</li><li>• YCbCr 4:4:4 Compress the image using the YCbCr color space scheme.</li><li>• RGB Compress the image using the red, blue, green (RGB) color space scheme.</li></ul> <p><b>Image signal</b> From the list, select the type of image signal that the block outputs:</p> <ul style="list-style-type: none"><li>• One multidimensional signal One signal where each dimension contains color information. Selecting this option creates one output port, Image.</li><li>• Separate color signals Multiple color signals where each signal contains the information for one color. Selecting this option creates three</li></ul>

# JPEG Decompression

---

output ports: R, G, and B for the red, green, and blue signals  
(or Y, CB, or Cr for YCbCr colorspace).

## **Show trigger input**

Select this check box to display an input port, Trigger, for the block.

## **See Also**

JPEG Compression

<b>Purpose</b>	USB Video Device List block
<b>Library</b>	xPC Target Library for USB Camera
<b>Description</b>	<p>When you connect a USB Video Class (UVC) webcam to the target computer, the USB Video Device List block probes the device and displays the manufacturer information of the webcam. Based on this information, the block configures its parameters with supported options. You can select the configuration parameters required by your USB webcam, then name the configuration for future use.</p> <p>Add this block when you add the From USB Video Device block to your model. You need the USB Video Device List block to configure the webcam.</p>
<b>Block Parameters</b>	<p><b>Manufacturer</b> From the list, select the manufacturer for the installed webcam.</p> <p><b>Format</b> From the list, select an image format that the connected webcam supports, for example, MJPEG.</p> <p><b>Resolution</b> From the list, select a supported resolution for the image.</p> <p><b>Interval</b> From the list, select the sample time for the video frame.</p> <p><b>Configurations</b> Use this parameter to store and name a particular configuration. A configuration consists of the settings you select for a particular webcam.</p> <p>After you select the options in the other parameters:</p> <ol style="list-style-type: none"><li>1 In the edit field, enter a name for the configuration.</li><li>2 Click the <b>Add</b> button to add the configuration.</li></ol> <p>To remove the configuration, click the <b>Remove</b> button.</p>

# USB Video Device List

---

## **Reload Device List**

Click this button to refresh the list of webcam information.  
Clicking this button also updates the mask and parameters for the From USB Video Device block.

## **See Also**

From USB Video Device

# XCP Over CAN Support

---

- “XCP Over CAN” on page 19-2
- “Blocks — Alphabetical List” on page 19-4

## XCP Over CAN

### Introduction

The Universal Measurement and Calibration Protocol (XCP) interconnects engineering control units (ECUs) in a CAN network. The xPC Target software supports this capability with the XCP sublibrary. A model that contains the blocks from this library can act as an XCP master that communicates with CAN network slave or ECU devices. The xPC Target XCP blocks do not work with UDP networks. Using the XCP sublibrary, you can:

- Use the XCP Protocol Stack block to synchronize one or more CAN network slave or ECU devices.
- Initialize an XCP slave server running in an ECU on a CAN network.
- Parse A2L (ASAP2 database) files.
- Acquire real-time measurement data when specific events occur.
- Support data stimulation.

### Using the XCP Blocks

When creating models to work as XCP masters with ECU slaves:

- Use the FIFO mode of the Softing CAN hardware with the XCP blocks. Do not use the object mode of the Softing CAN hardware with the XCP blocks. (See “FIFO Mode” on page 8-2.)
- Use one XCP Protocol Stack block for each CAN channel.
- Provide two A2L format files:
  - One A2L (ASAP2) file that the model can load into the CAN software database. The generated file contains signal and parameter access information for the target application. It describes the slave ECUs.
  - One A2L (ASAP2) file that the model can load into the CAN software database. The generated file contains signal and parameter access information for the XCP specific elements of the network.

---

**Note** If you have one A2L file that contains this information, separate the information into two A2L files as previously described.

---

## **Blocks – Alphabetical List**

XCP Protocol Stack

XCP DAQSTIM

<b>Purpose</b>	XCP Protocol Stack block
<b>Library</b>	xPC Target for XCP
<b>Description</b>	The XCP Protocol Stack block implements the transportation layer for the communication between the target PC and the CAN network. It also initializes ECUs on CAN networks. Use one XCP Protocol Stack block for each CAN channel.
<b>Block Parameters</b>	<p><b>Protocol Stack ID</b> Enter the identifier for the XCP protocol stack. This value must match the value of the <b>Stack ID</b> parameter in the corresponding XCP DAQSTIM block.</p> <p><b>CAN TX Port</b> From the list, select the index of the CAN transmit port.</p> <p><b>CAN Rx Port</b> From the list, select the index of the CAN receive port.</p> <p><b>Max CAN Frame Tx (Per sample time)</b> Enter the maximum number of frames the target application can transmit per sample time.</p> <p><b>Max CAN Frame Rx (Per Sample time)</b> Enter the maximum number of frames the target application can receive per sample time.</p> <p><b>Sample time</b> Enter the base sample time or a multiple of the base sample time.</p>
<b>See Also</b>	XCP DAQSTIM

# XCP DAQSTIM

---

**Purpose** XCP DAQSTIM block

**Library** xPC Target for XCP

**Description** The XCP DAQSTIM block manages the synchronous data communication with an XCP slave running on the CAN network. This block supports:

- Data acquisition — Data flows from the ECU that supports the XCP slave to the master.
- Data stimulation — Data flows from the master to the ECU that supports the XCP slave.

The block dialog box supports configuration of the dynamic data acquisition list with a selection tool. As you configure the block, the block inputs and outputs change correspondingly.

- 1** Specify an ECU configuration A2L file (ASAP2 database) in the **Main A2L File** parameter.
- 2** Specify an XCP slave configuration A2L file (ASAP2 database) in the **XCP A2L File** parameter.
- 3** Observe that the block populates the **Events** and **Signals** parameters with the contents of the two A2L files.
- 4** Use the add and remove buttons on the selection tool to move event channels into a configuration. Assign signals to those channels for the configuration.

## **Block Parameters**

### **Events**

The software populates this field with event channels based on the contents of the files you specify in **Main A2L File** and **XCP A2L File**. Select the event channel to associate with the periodic data acquisition and stimulation of signals. You can configure

multiple event channels to obtain different rates of triggering for the provided signal acquisition or stimulation events.

The event channel names clearly indicate:

- Data flow direction (data acquisition or data stimulation)
- Slave-side event type, such as the data flow rate

### Signals

The software populates this field based on the contents of the files you specify in **Main A2L File** and **XCP A2L File**.

### Configuration

Use the add and remove buttons to move signals and events into configurations for the slave ECU.

### Main A2L File

Specify the A2L file that describes the available measurement and characteristics. The block maps the selected measurements as signals.

### XCP A2L File

Specify the A2L file that describes a slave that supports XCP. The file contains the IF\_DATA settings of the A2L database.

### Slave ID

Enter the identifier for the slave ECU you are configuring.

### Stack ID

Enter the stack identifier for the slave ECU you are configuring. This value must match the value of the **Protocol Stack ID** parameter in the corresponding XCP Protocol Stack block.

## See Also

XCP Protocol Stack

# XCP DAQSTIM

---

# ACCES I/O

---

This topic describes the ACCES I/O board supported by the xPC Target product (<http://www.accesio.com>).

## ACCES I/O WDG-CSM

The WDG-CSM is a watchdog timer board used to detect computer failure.

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

ACCES I/O WDG-CSM

ACCES I/O WDG-CSM Watchdog Timer

**Purpose** Support for ACCES I/O WDG-CSM watchdog timer

**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.

The xPC Target block library supports this board with one driver block:

- ACCES I/O WDG-CSM Watchdog Timer

<b>Board Characteristics</b>	Board name	WDG-CSM
	Manufacturer	ACCES IO
	Bus type	ISA
	Access method	I/O mapped
	Multiple block instance support	No
	Multiple board support	Yes

# ACCES I/O WDG-CSM Watchdog Timer

---

**Purpose** WDG-CSM Watchdog Timer block

**Library** xPC Target Library for Access

**Block Parameters** **Watchdog Time [s] (20us-4800s)**  
Enter a timeout value in seconds.

**Show reset port**

Select this check box to enable an input port on the driver block.  
A signal connected to this port resets the watchdog.

**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

# ADDI-DATA

---

This topic describes I/O boards supported by the xPC Target product (<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 21-2

Description of block parameters for ADDI-DATA driver blocks.

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

**Purpose** Support for APCI-1710 general purpose counting board

**Board** APCI-1710

**General Description** The APCI-1710 is a general-purpose counting board with four function modules.

The xPC Target block library supports this board with this driver block:

- ADDI-DATA APCI-1710 Incremental Encoder

Note, the xPC Target software 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 required 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  $0$  to  $2\pi$  or  $-\pi$  to  $+\pi$ , 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 APCI-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 APCI-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
```

**Purpose** Support for the PI1700 counter board

**Board** The PA1700 is a counter board with three 24-bit counters for connecting three incremental encoders.

The xPC Target block library supports this board with this driver block:

- ADDI-DATA PA-1700 Incremental Encoder

<b>Board Characteristics</b>	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.

# **ADDI-DATA PA-1700 Incremental Encoder**

---

# Adlink

---

This topic describes the Adlink boards supported by the xPC Target product (<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 22-2

Description of block parameters for Adlink driver blocks.

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

**Purpose** Support for Adlink PCI-8133 encoder counter and PWM output board

**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.

The xPC Target block library 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, connect the OENA pin (pin 34 of connector CN1) 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 required 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
```

**Purpose** Support for the Adlink PCI-6208 digital-to-analog converter board

**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.

The xPC Target block library 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

## Scaling Output to Input

Hardware Output	Block Input Data Type	Scaling
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 milliamperes (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:

- This block is scaled for the current output ports, not the voltage output ports.
- 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
```

**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
```

# Adlink PCI-6208A Digital Output

---

# Advantech

---

This topic describes I/O boards supported by the xPC Target product (<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 23-3	Description of block parameters for Advantech® driver blocks.

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

# Advantech PCL-1800

---

**Purpose** Advantech PCL-1800 board

**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.

The xPC Target block library 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

<b>Board Characteristics</b>	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)

---

<b>Input Range (V)</b>	<b>Range Code</b>	<b>Input Range (V)</b>	<b>Range Code</b>
-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

<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

**Purpose** Advantech PCL-711B board

**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.

The xPC Target block library 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

<b>Board Characteristics</b>	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

## Scaling of Input to Output

Hardware Input	Block Output Data Type	Scaling
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.

Input Range (V)	Range Code
-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
-0.3125 to +0.3125	-0.3125

For example, if the first channel is -5 to +5 volts and the second channel is -2.5 to +2.5 volts, enter

# Advantech PCL-711B Analog Input (A/D)

---

[-5, -2.5]

## **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 JP1 jumper 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-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

## 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

**Purpose** Advantech PCL-726 board

**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.

The xPC Target block library supports this board with these driver blocks:

- Advantech PCL-726 Analog Output (D/A)
- Advantech PCL-726 Digital Input
- Advantech PCL-726 Digital Output

**Board Characteristics**

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

---

**Purpose** Advantech PCL-727 board

**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.

The xPC Target block library supports this board with these driver blocks:

- Advantech PCL-727 Analog Output (D/A)
- Advantech PCL-727 Digital Input
- Advantech PCL-727 Digital Output

## Board Characteristics

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

<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 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.

<b>Input Range (V)</b>	<b>Range Code</b>
-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

$\geq 0.5$  = 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

**Purpose** Advantech PCL-728 board

**Board** Advantech PCL-728

**General Description** The PCL-728 is an I/O board with two independent analog output D/A channels (12-bit).

The xPC Target block library supports this board with this driver block:

- Advantech PCL-728 Analog Output (D/A)

<b>Board Characteristics</b>	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

---

**Purpose** Advantech PCL-812 board

**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.

The xPC Target block library 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

<b>Board Characteristics</b>	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

## 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

**Purpose** Advantech PCL-812PG board

**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.

The xPC Target block library 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

<b>Board Characteristics</b>	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

---

**Purpose** Advantech PCL-818 board

**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.

The xPC Target block library 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

<b>Board Characteristics</b>	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.

<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

# 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

**Purpose** Advantech PCL-818H board

**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.

The xPC Target block library 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

**Purpose** Advantech PCL-818HD board

**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.

The xPC Target block library 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

<b>Board Characteristics</b>	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

<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-818HD Digital Output

---

**Purpose** PCL-818HD 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

**Purpose** Advantech PCL-818HG

**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.

The xPC Target block library 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

<b>Board Characteristics</b>	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

**Purpose** Advantech PCL-818L Board

**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.

The xPC Target block library 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

<b>Board Characteristics</b>	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 matches the output voltage to 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

## 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 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

<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 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
```

# Advantech PCL-818L Digital Output

---

# Analogic

---

This topic describes the Analogic I/O boards supported by the xPC Target product (<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 24-2	Description of block parameters for Analogic driver blocks.

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

**Purpose** Analogic AIM board

**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 required 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 reads the hardware configuration and adjusts the integer to float conversion accordingly.

The xPC Target block library supports this board with these driver blocks:

- Analogic AIM12 Analog Input (A/D)
- Analogic AIM12 Digital Input
- Analogic AIM12 Digital Output

# Analogic AIM

---

## **Board Characteristics**

Board name	AIM12
Manufacturer	Analogic
Bus type	PC/104
Access method	I/O mapped
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.

**Purpose** Analogic AIM16 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 required 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 reads the hardware configuration and adjusts the integer to float conversion accordingly.

The xPC Target block library 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

# Analogic 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 topic describes the BittWare boards supported by the xPC Target product (<http://www.bittware.com>). Note that you can no longer buy this board from BittWare. MathWorks supplies and supports the drivers and documentation for the customers who already own this board.

“Running Models with BittWare Blocks” on page 25-2

Usage notes to help use the BittWare blocks in your model.

“Boards and Blocks — Alphabetical List” on page 25-6

Description of block parameters for BittWare driver blocks

## Running Models with BittWare Blocks

### In this section...

“Model Notes” on page 25-2

“Frame Size, Sample Rate, and Sample Time Parameter Notes” on page 25-4

### Model Notes

This topic describes notes on working with models that use the BittWare blocks.

- “Model Execution Timing” on page 25-2
- “Example Models” on page 25-3
- “Model Execution Limitations” on page 25-4

### 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 the xPC Target software knows about.

- 2 Find the IRQ specified for this board. This is the interrupt source number you need to specify in the **Real-time interrupt source** field of the **xPC target options** node.

Set the interrupt vector number:

- 1 From the MATLAB Command Window, type the name of your Simulink model.

The Simulink model appears

- 2 Click **Simulation > Model Configuration Parameters**.

The Configuration Parameter dialog box is displayed for the model.

- 3 Select the **xPC Target options** node under the **Code Generation** node.
- 4 Set the **Execution mode** field to Real-Time.
- 5 Click the **Real-time interrupt source** list.
- 6 Select the interrupt source number to which the board is set.
- 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. To operate these, set the required interrupt source in the simulation parameters dialog.

`xpc8audiochannels` shows the use of all 8 input and 8 output channels at the same time. A block from DSP System Toolbox™ is used to convert from frame to sample based signals to drive the xPC Target scopes.

`xpcaudiospectrum` uses the xPC Target Spectrum Scope subsystem from the `xpcdsp spectrum` sample model with the Audio-PMC+ as the source. This model requires that **Frame Size** be set to 1 because of the way 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 DSP System Toolbox. If the frame size is 1, then the blocks revert to sample based signals. The blocks in DSP System Toolbox expect to see signals that are frames.

## Frame Size, Sample Rate, and Sample Time 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 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 motherboard
- Main memory speed
- General motherboard architecture

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

**Purpose** Support for BittWare Audio PMC+ board

**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. The xPC Target block library 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 DSP System Toolbox 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** Remove the Audio-PMC+ from the carrier board and look for 6 pairs of pins along one edge. Check if any of the pairs are jumpered.
- 2** If any pairs are jumpered, remove each jumper and reinstall it with only one pin connected. Leave the other side hanging. Leaving the jumpers like this will leave them available 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.

---

**Note** You can only set this parameter for the analog input block. The analog output block determines the data type from the connection and sets itself to the required type.

---

**Frame Size, Sample Rate, and Sampletime**

These three parameters are not independent but are related by

$$FrameSize = Sampletime * SampleRate.$$

After you specify two of the parameters, the equation determines the third parameter.

# BittWare Audio-PMC+ Analog Input

---

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.

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 computer, enter

-1

to automatically locate the board.

If two or more boards of this type are in the target computer, enter the bus number and the PCI slot number of 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 25-2

# 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

$$FrameSize = Sampletime * SampleRate.$$

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.

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 computer, enter

-1

# BittWare Audio-PMC+ Analog Output

---

to automatically locate the board.

If two or more boards of this type are in the target computer, enter the bus number and the PCI slot number of 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 25-2

# BVM

---

This topic describes the BVM I/O board supported by the xPC Target product (<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 – Alphabetical List**

BVM PMCDIO64

BVM PMCDIO64 Digital Input

BVM PMCDIO64 Digital Output

**Purpose** Support for 64-bit digital I/O

**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.

The xPC Target block library supports this board with these driver blocks:

- BVM PMCDIO64 Digital Input
- BVM PMCDIO64 Digital Output

<b>Board Characteristics</b>	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  $\geq 0.5$ .

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
```

# Contec

---

This topic describes the Contec I/O boards supported by the xPC Target product (<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)	Opto-isolated 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 27-3	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.

The xPC Target block library 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

The xPC Target software 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 (differential-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.)

# Contec AD12-16(PCI) Analog Input (A/D)

---

## 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:

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 (differential-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
```

# Contec AD12-16(PCI) Analog Input (A/D)

---

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

<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.

## 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.

## **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)E

---

## 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.

The xPC Target block library 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)

The xPC Target software 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 (differential-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.

**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.

The xPC Target block library 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)

The xPC Target software 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 (differential-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.

# Contec AD12-64(PCI)

---

**Purpose** Contec AD12-64(PCI) block

**General 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.

The xPC Target block library 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

The xPC Target software 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 (differential-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 (differential-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.

## 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.

The xPC Target block library 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)

The xPC Target software 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 (differential-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.

# Contec ADI12-16(PCI)

---

## 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.

The xPC Target block library supports this board with these driver block:

- Contec ADI12-16(PCI) Analog Input (A/D)

## Board Characteristics

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 (differential-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.

## 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.

The xPC Target block library supports this board with one driver block:

- Contec CNT24-4D(PCI) Incremental Encoder

The xPC Target software 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) 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.

**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. The xPC Target block library supports this board with this driver block:

- Contec CNT32-8M(PCI) Incremental Encoder

**Board Characteristics**

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)

<b>Hardware Output</b>	<b>Block Output Data Type</b>	<b>Scaling</b>
TTL	Double	TTL low = < 0.5 TTL high = > 0.5

## **Block Parameters: Main Tab**

The **Main** tab contains parameters that define the 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

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**.

# Contec CNT32-8M(PCI) Incremental Encoder

---

## 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.

## 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:

- **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.

# Contec CNT32-8M(PCI) Incremental Encoder

---

- 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.

## **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.

## Block Parameters: Signals Tab

### 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.

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.

### 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.

# Contec CNT32-8M(PCI) Incremental Encoder

---

## 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.

## 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.

# Contec DA12-16(PCI)

---

**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).

The xPC Target block library supports this board with this driver block:

- Contec DA12-16(PCI) Analog Output (D/A)

The xPC Target software 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.

### Range

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 analog output channels, or a vector the same length as the channel vector.

The range vector entries must be range codes selected from the following table:

<b>Input Range (V)</b>	<b>Range Code</b>
-10 to +10	-10
-5 to +5	-5
-2.5 to 2.5	-2

# Contec DA12-16(PCI) Analog Output (D/A)

---

For example, if the first and second channel is -5 to +5 volts, enter [-5,-5]

## **Reset vector**

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

## **Initial value vector**

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

## **Sample time**

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

## **PCI Slot (-1:autosearch)**

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

-1

to automatically locate the board.

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of 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.

**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).

The xPC Target block library supports this board with this driver block:

- Contec DA12-4(PCI) Analog Output (D/A)

The xPC Target software does not support the timer, external trigger, or interrupt functionality of this board.

**Board Characteristics**

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.

# Contec PI-64L(PCI)H

---

## 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).

The xPC Target block library supports this board with this driver block:

- Contec PI-64L(PCI)H Digital Input

## Board Characteristics

Board name	PI-64L(PCD)H
Manufacturer	Contec
Bus type	PCI
Multiple block instance support	No
Multiple board support	Yes

**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:32 1-bit Channels</b> )	Double: TTL low = 0.0 TTL high = 1.0
	uint32 ( <b>Format:Single 32-bit Port</b> )	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 the required data frames.

# Contec PI-64L(PCI)H Digital Input

---

## 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.

**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
```

# Contec PIO-32/32F(PCI)

---

## Board

Contec PIO-32/32F(PCI)

## General Description

The Contec PIO-32/32F(PCI) is an opto-isolated high-speed I/O board with 32 digital input channels and 32 output channels.

The xPC Target block library supports this board with these driver blocks:

- Contec PIO-32/32T(PCI) Digital Input
- Contec PIO-32/32T(PCI) Digital Output

The xPC Target software does not support the timer, external trigger, or interrupt functionality of this board.

## Board Characteristics

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.

# Contec PIO-32/32F(PCI) Digital Output

---

## 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('fffffff')]`.

## 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.

## 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).

The xPC Target block library supports this board with these driver blocks:

- Contec PIO-32/32L(PCI)H Digital Input
- Contec PIO-32/32L(PCI)H Digital Output

The xPC Target software 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 the required 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.

## **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

## Scaling Input to Output

Hardware Output	Block Input Data Type	Scaling
TTL	Double ( <b>Format:32 1-bit Channels</b> )	Double: < 0.5 = TTL low >0.5 = TTL high
	uint32 ( <b>Format:Single 32-bit Port</b> )	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 the required data frames.

# Contec PIO-32/32L(PCI)H Digital Output

---

## 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.

## PCI Slot (-1:autosearch)

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

-1

to automatically locate the board.

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of 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-32/32T(PCI)

---

## 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.

The xPC Target block library supports this board with these driver blocks:

- Contec PIO-32/32T(PCI) Digital Input
- Contec PIO-32/32T(PCI) Digital Output

The xPC Target software does not support the timer, external trigger, or interrupt functionality of this board.

## Board Characteristics

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('ffffffff')]`.

## **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.

## 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).

The xPC Target block library supports this board with this driver block:

- Contec PO-64L(PCI)H Digital Output

## Board Characteristics

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

<b>Scaling Input to Output</b>	<b>Hardware Output</b>	<b>Block Input Data Type</b>	<b>Scaling</b>
	TTL	Double ( <b>Format:32</b> 1-bit Channels)  uint32 ( <b>Format:Single</b> 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.

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.

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 the required 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

# Contec PO-64L(PCI)H Digital Output

---

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
```

# Curtiss-Wright Electronic Systems

---

xPC Target supports reflective (shared) memory boards manufactured by Curtiss-Wright Electronic Systems (formerly Systran). The use of these boards is described in “Curtiss-Wright Electronic Systems Shared Memory” on page 17-23.

Systran SCRAMNet+  
SC150 PCI

A real-time reflective PCI memory board that can generate/broadcast interrupts.

## **Board and Blocks – Alphabetical List**

Systran SCRAMNet+ SC150 PCI  
Systran SC150 init  
Systran SC150 read  
Systran SC150 rearm  
Systran SC150 write  
Systran SCRAMNet+ SC150 PCI  
Systran SC150 init  
Systran SC150 read  
Systran SC150 rearm  
Systran SC150 write

**Purpose** Support for Systran SCRAMNet+ SC150 PCI real-time reflective memory board

**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. The xPC Target software uses this board as part of the shared memory network that you can use to exchange data between computer nodes.

The Systran block library supports this board with these driver blocks:

- Systran SC150 init
- Systran SC150 read
- Systran SC150 write
- Systran SC150 rearm

For information about the Change endianness block, see Byte Reversal/Change Endianness.

<b>Board Characteristics</b>	Board name	SCRAMNet+ SC150
	Manufacturer	Curtiss-Wright Electronic Systems (formerly Systran)
	Bus type	PCI
	Access method	Memory mapped
	Multiple block instance support	Yes
	Multiple board support	Yes

# Systran SC150 init

---

**Purpose** SystranSC150 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 17-25.

**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 computer, enter

-1

to automatically locate the board.

If two or more boards of this type are in the target computer, enter the bus number and the PCI slot number of 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 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 17-23.

## **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 computer, enter

-1

to automatically locate the board.

If two or more boards of this type are in the target computer, enter the bus number and the PCI slot number of 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 SC150 rearm

---

**Purpose** Systran SC150 rearm 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 rearm 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 17-23.

## **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 computer, enter

-1

to automatically locate the board.

If two or more boards of this type are in the target computer, enter the bus number and the PCI slot number of 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 SC150 write block

**Library** Systran 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 17-23.

## **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 computer, enter

-1

to automatically locate the board.

If two or more boards of this type are in the target computer, enter the bus number and the PCI slot number of 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 SC150 write

---

# Data Translation

---

This topic describes Data Translation® I/O boards supported by the xPC Target product (<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 29-4

Description of block parameters for Data Translation driver blocks.

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

**Purpose** Data Translation DT2821 board

**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.

The xPC Target block library 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.

## 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

### 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

---

**Purpose** Data Translation DT2821-F-8DI board

**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.

The xPC Target block library 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.

## 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-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

**Purpose** Data Translation DT2821-G-8DI board

**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.

The xPC Target block library 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

---

**Purpose** Data Translation DT2821-F-16SE board

**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.

The xPC Target block library 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.

<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-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 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 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.

<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-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

**Purpose** Data Translation DT2821-G-16SE board

**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.

The xPC Target block library 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.

## 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-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

---

**Purpose** Data Translation DT2823 board

**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.

The xPC Target block library 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.

<b>Input Range (V)</b>	<b>Range Code</b>
-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.

## 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 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

---

**Purpose** Data Translation DT2824-PGH board

**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.

The xPC Target block library 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.

## 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-PGH 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-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.

<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-PGH 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

**Purpose** Data Translation DT2824-PGL board

**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.

The xPC Target block library 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

<b>Board Characteristics</b>	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

---

**Purpose** Data Translation DT2825 board

**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.

The xPC Target block library 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 +- 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.

<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 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 correspond 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 correspond to the DIP switch settings on the board. For example, if the base address is 300 (hexadecimal), enter

0x300

**Purpose** Data Translation DT2827 board

**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.

The xPC Target block library 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

<b>Board Characteristics</b>	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.

<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

# 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 correspond 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 correspond to the DIP switch settings on the board. For example, if the base address is 300 (hexadecimal), enter

0x300

**Purpose** Data Translation DT2828 board

**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.

The xPC Target block library 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

## 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
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

# Data Translation DT2828 Digital Output

---

# Diamond

---

This topic describes Diamond Systems I/O boards supported by the xPC Target product (<http://www.diamondsystems.com>). To read about the Diamond Systems serial communications boards supported by xPC Target, see “RS-232/422/485 Drivers (Composite)” on page 4-6.

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, Athena, and Athena II

- Diamond Prometheus — 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.

The xPC Target product 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 — 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.

	<ul style="list-style-type: none"> <li>• Diamond Athena II — Single board computer (SBC). Pentium III class VIA Mark CoreFusion™ 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 10/100 Mbps Ethernet connector, and a 16-bit data acquisition circuit.</li> </ul>
Diamond Quartz-MM-5	8 digital input lines and 8 digital output lines.
Diamond Quartz-MM-10	8 digital input line and 8 digital output lines.
Diamond Ruby-MM	PC/104 I/O board with 4 or 8 single analog output (D/A) channels, unipolar and bipolar operation, $\pm 10V$ , $\pm 5V$ , 0-10V, 0-5V fixed ranges, $\pm 2.5V$ , 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 30-4	Description of block parameters for Diamond Systems driver blocks.

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

**Purpose** Diamond Diamond-MM board

**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.

The xPC Target block library 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.

<b>Input Range (V)</b>	<b>Range Code</b>	<b>Input Range (V)</b>	<b>Range Code</b>
-10 to +10	-10	0 to +10	10
-5 to +5	-5	0 to +5	5

# Diamond MM Analog Input (A/D)

Input Range (V)	Range Code	Input Range (V)	Range Code
-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 set for the required 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.

## 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 set as required 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

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 range defined by 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 range defined by 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

**Purpose** Diamond Diamond-MM-16-AT board

**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.

The xPC Target block library 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

The xPC Target software does not support the counters/timers on this board.

<b>Board Characteristics</b>	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 range defined by 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 range defined by 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

**Purpose** Diamond Diamond-MM-32-AT board

**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.

The xPC Target block library supports this board with these driver blocks. These blocks also support the Diamond-MM-32X-AT board.

- Diamond MM-32-AT Analog Input (A/D)
- 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

<b>Board Characteristics</b>	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 > Model Configuration Parameters**.
- 2** Select the **Code Generation** node.
- 3** In the **Target selection** section, from the **System target file** list, browse to and select `xpctarget.tlc`.

# Diamond MM-32-AT Frame Analog Input (A/D)

---

- 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.
- 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 (-1: autosearch) or 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 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)

# Diamond MM-32-AT Frame Analog Input (A/D)

---

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

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

## Output Signal Type

From the list, select either Vector or Frame:

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

## Range

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

## First channel number

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

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

## Number of channels

Enter the number of input channels you want to use. The maximum number of channels varies between 1 and 32 and depends on **Channel configuration** and the **First channel number**. Note that hardware limitations require that either the **Number of channels** or **Number of scans per frame** value be

# Diamond MM-32-AT Frame Analog Input (A/D)

---

even. If your application requires that both quantities be odd, add 1 to one of them and ignore the resulting additional data.

## **Number of scans per frame**

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

## **Interval between conversions within a scan**

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

## **Interval between scans**

Enter the interval, in seconds, between successive scans.

## **Base address**

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

0x300

# Diamond MM-32-AT Analog Output (D/A)

---

**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 set as required by 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.

## 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 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. One block is required for each port 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

# 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. One block is required for each port 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 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

---

**Purpose** Diamond Garnet-MM board

**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.

The xPC Target block library supports this board with these driver blocks:

- Diamond Garnet-MM Digital Input
- Diamond Garnet-MM Digital Output

<b>Board Characteristics</b>	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.

<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. 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

**Purpose** Diamond Onyx-MM board

**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. The xPC Target software does not support the Counter/Timer functionality of this board.

The xPC Target block library 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.

**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

---

**Purpose** Diamond Onyx-MM-DIO board

**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. The xPC Target block library supports this board with these driver blocks:

- Diamond Onyx-MM Digital Input
- Diamond Onyx-MM Digital Output

## Board Characteristics

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.

## 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 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

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 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, Athena, and Athena II

---

## **Purpose**

Diamond Prometheus, Athena, and Athena II boards

## **Board**

Diamond Prometheus, Athena, Athena II

## **General Description**

The xPC Target product supports the Diamond Prometheus, Athena, and Athena II boards.

- Diamond Prometheus — 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.

The xPC Target product 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 — 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.
- Diamond Athena II — Single board computer (SBC). Pentium III class VIA Mark CoreFusion 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 10/100 Mbps Ethernet connector, and a 16-bit data acquisition circuit.

The xPC Target block library supports the Prometheus Model PR-Z32-EA and Athena with four driver blocks:

- Diamond Prometheus/Athena/Athena II Analog Input (A/D)
- Diamond Prometheus/Athena/Athena II Analog Output (D/A)
- Diamond Prometheus/Athena/Athena II Digital Input
- Diamond Prometheus/Athena/Athena II Digital Output

# Diamond Prometheus, Athena, and Athena II

---

The xPC Target software does not support the counter/timer on this board.

## **Board Characteristics**

Board names	Prometheus, Athena, and Athena II
Manufacturer	Diamond Systems
Bus type	PC/104
Multiple block instance support	A/D: No, D/A: No, Digital I/O: Yes
Multiple board support	No

# Diamond Prometheus/Athena/Athena II 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/Athena II 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/Athena II 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/Athena II 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 range defined by 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/Athena II 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.

## 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 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/Athena II 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/Athena II 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/Athena II 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

**Purpose** Diamond Quartz-MM-10 board

**Board** Diamond Quartz-MM-10

**General Description** The Quartz-MM 10 has eight digital input line and eight digital output lines.

The xPC Target block library supports this board with these driver blocks:

- Diamond Quartz-MM-10 Digital Input
- Diamond Quartz-MM-10 Digital Output

**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-5

---

**Purpose** Diamond Quartz-MM-5 board

**Board** Diamond Quartz-MM-5

**General Description** The Quartz-MM 5 has 8 digital input lines, 8 digital output lines. The xPC Target block library supports this board with these driver blocks:

- Diamond Quartz-MM-5 Digital Input
- Diamond Quartz-MM-5 Digital Output

<b>Board Characteristics</b>	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

**Purpose** Diamond Ruby-MM board

**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,  $\pm 10V$ ,  $\pm 5V$ , 0-10V, 0-5V fixed ranges,  $\pm 2.5V$ , 0-2.5V user-adjustable ranges, 24 digital input and output lines.

The xPC Target block library supports this board with these driver blocks:

- Diamond Ruby-MM Analog Output (D/A)
- Diamond Ruby-MM Digital Input
- Diamond Ruby-MM Digital Output

<b>Board Characteristics</b>	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 unexpected 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 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. One block is required for each port 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 in hexadecimal (such as 0x300) as selected with the hardware jumpers on the board.

# Diamond Ruby-MM Digital Input

---

---

**Note** Please consult the 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. One block is required for each port 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 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 Diamond Ruby-MM hardware manuals for more information on jumper settings and I/O connections.

---

# Diamond Ruby-MM-416

---

**Purpose** Diamond Ruby-MM-416 board

**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.

The xPC Target block library 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

## Board Characteristics

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

**Purpose** Diamond Ruby-MM-1612 board

**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.

The xPC Target block library 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.

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)

B1	B2	Configuration
±10 V	±10 V	C11
0-10 V	±10 V	C41
±5 V	±5 V	C22
±2.5	±5 V	C32
0-5 V	±5 V	C52
0-2.5 V	±5 V	C62
±5 V	±2.5 V	C23
±2.5 V	±2.5 V	C33
0-5 V	±2.5 V	C53
0-2.5 V	±2.5 V	C63
±10 V	0-10 V	C14
0-10 V	0-10 V	C44
±5 V	0-5 V	C25
±2.5 V	0-5 V	C35
0-5 V	0-5 V	C55
0-2.5 V	0-5 V	C65
±5 V	0-2.5 V	C26
±2.5 V	0-2.5 V	C36
0-5 V	0-2.5 V	C56
0-2.5 V	0-2.5 V	C66

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.

## 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-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

# GE Intelligent Platforms

---

xPC Target supports boards manufactured by GE Intelligent Platforms (<http://defense.ge-ip.com/products/family/avionics>). This company, formerly known as GE Fanuc, acquired Condor Engineering, SBS Technologies, and VMIC with all their products. For more about these boards and blocks, see:

- Condor Engineering
  - “ARINC 429 Support” on page 31-2
  - “MIL-STD-1553 Support” on page 31-3
- SBS Technologies
  - “GE Intelligent Platforms Shared Memory” on page 17-13
  - “SBS Technologies Support” on page 31-17
- VMIC
  - “GE Fanuc Embedded Systems Shared Memory” on page 17-2
- “Boards and Blocks — Alphabetical List” on page 31-19

## ARINC 429 Support

xPC Target supports the ARINC 429 bus via the CEI-X20 series boards manufactured by GE Intelligent Platforms (<http://defense.ge-ip.com/products/family/avionics>) – formerly Condor Engineering.

Condor CEI-520/520A	Support for ARINC 429 CEI-520/5201A board.
Condor CEI-x20 Initialize	Condor CEI-x20 Initialize block
Condor CEI-x20 Receive	Condor CEI-x20 Receive block
Condor CEI-x20 Send	Condor CEI-x20 Send block
Condor Decode ARINC 429 Words from Receive	Decode ARINC 429 Words from Receive block
Condor Encode ARINC 429 Words for Send	Encode ARINC 429 Words for Send block

## MIL-STD-1553 Support

xPC Target supports the MIL-STD-1553 bus via boards manufactured by GE Intelligent Platforms (<http://defense.ge-ip.com/products/family/avionics>) – formerly Condor Engineering.

### In this section...

“Before You Start” on page 31-3

“Remote Terminal Operation” on page 31-5

“Bus Controller Operation” on page 31-7

“Remote Terminal and Bus Controller Operation” on page 31-10

“Bus Monitor Operation” on page 31-12

“Boards and Blocks” on page 31-14

### Before You Start

The xPC Target software interfaces the target computer to a 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 GE Fanuc (formerly Condor Engineering) PCI-1553, QPCI-1553 and Q104 series boards (GE Intelligent Platforms).

The xPC Target I/O block library supports the MIL-STD-1553 bus with the MIL-STD-1553 sublibrary. 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
- Q104-1553 — Use these blocks to communicate with the Q104-1553 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

Regardless of the operation, always initialize your board with the Initialize block for the board type you are using. (Be sure to use the board block grouping required by your board type — PCI, QPCI, Q104.) The examples in this topic use the QPCI blocks.

---

**Note** The QPCI and Q104 boards have almost the same functionality, except that the Q104 does not support loopback connections and is addressed by PC104 base address instead of by PCI slot.

---

The QPCI Initialize block allows you to specify the board operation: Remote Terminal, Bus Controller, or Bus Monitor. The mask dialog box for this block changes depending on the operation you select.

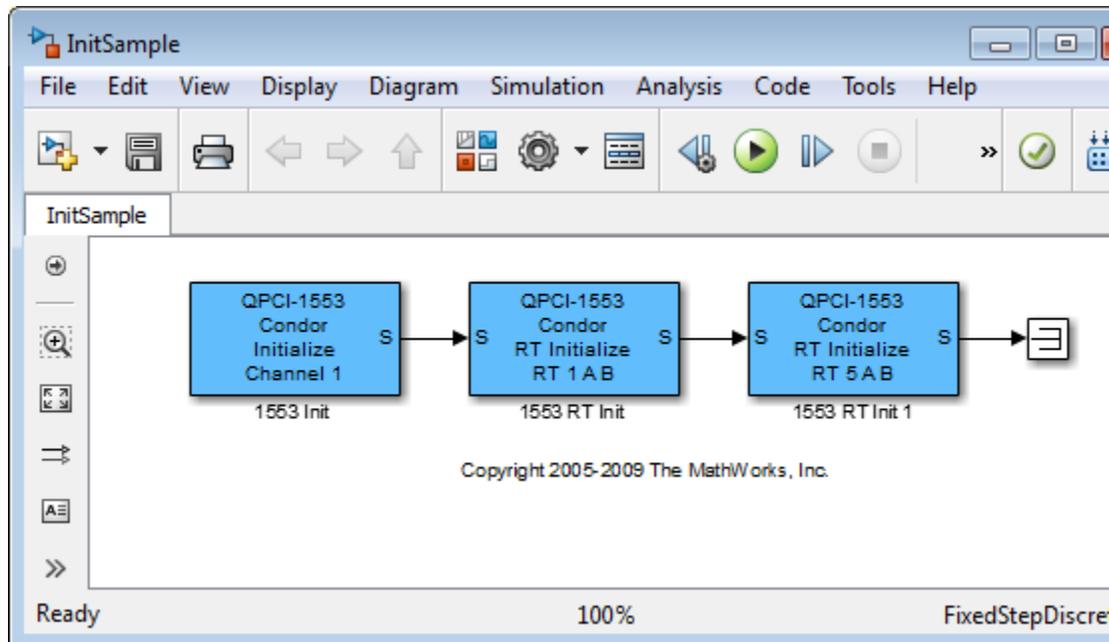
- Select the **Initialize for Bus Controller operation** check box for the Bus Controller
- Select the **Initialize for Bus Monitor operation** check box for the Bus Monitor
- Select the **Initialize for Remote Terminal operation** check box for the Remote Terminal

By default, these check boxes are selected. If you deselect an operation check box, the block grays out the associated parameters.

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 and other examples are located in the `xpcdemos` directory. 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.

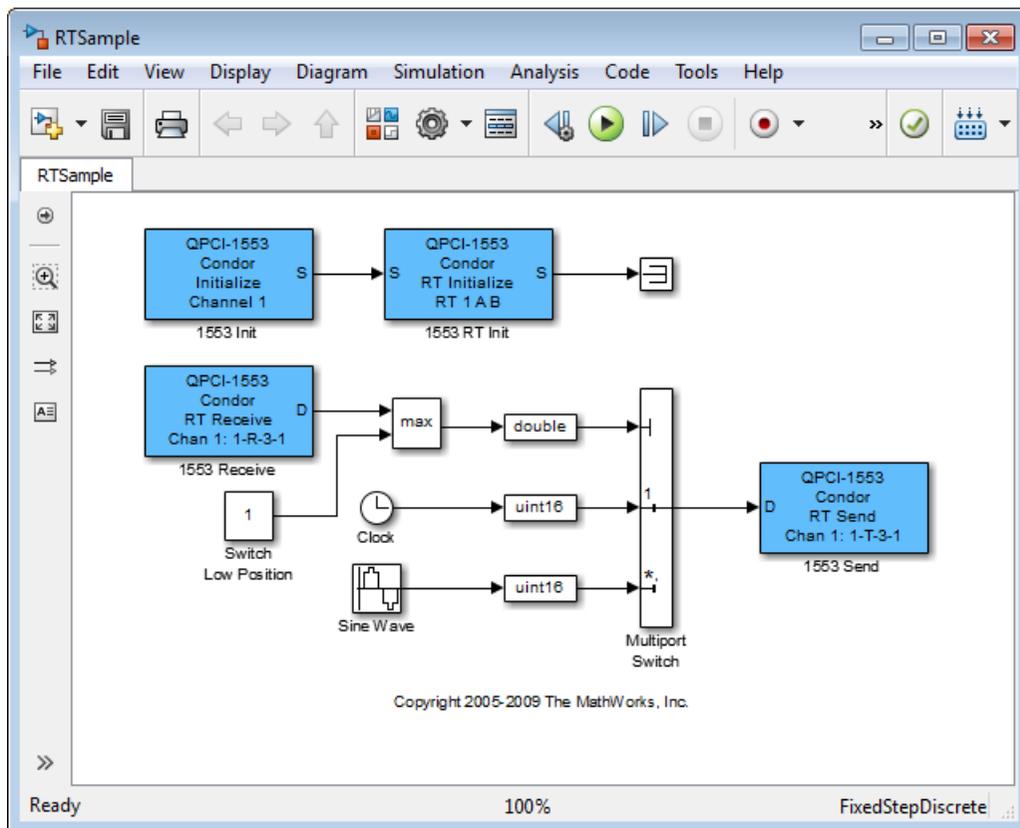
You can configure each subaddress for transmit, receive, or both. Configure all the subaddresses you plan to use.



## 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 computer that has a QPCI-1553 board. Replace the hardware specific blocks with the blocks for your hardware configuration.



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 QPCI-1553 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.

---

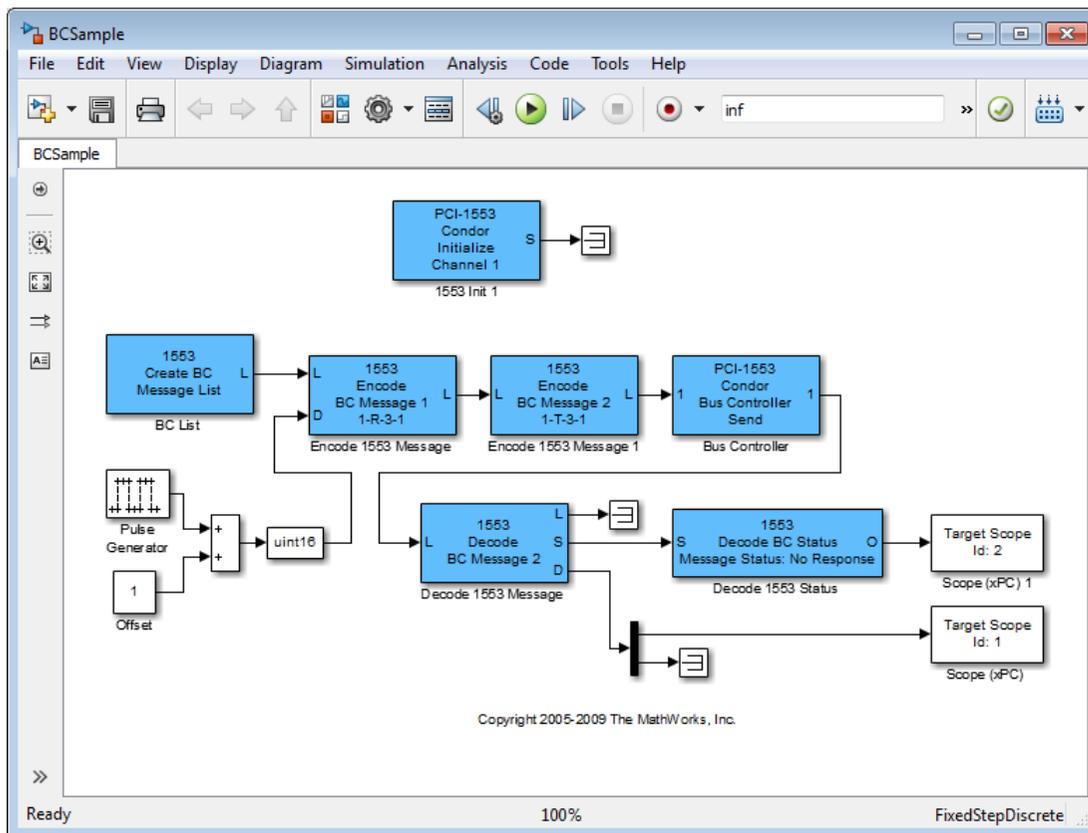
**Note** When the board receives a transmit or receive message, it must respond quickly because the default timeout is 14 microseconds. This timeout is much quicker than the ability of the xPC Target software to reliably execute a model. This situation might cause the board to transmit data from the 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 computer that has a PCI-1553 board. Replace the hardware specific blocks with the blocks for your hardware configuration.



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 time step during which the message should be

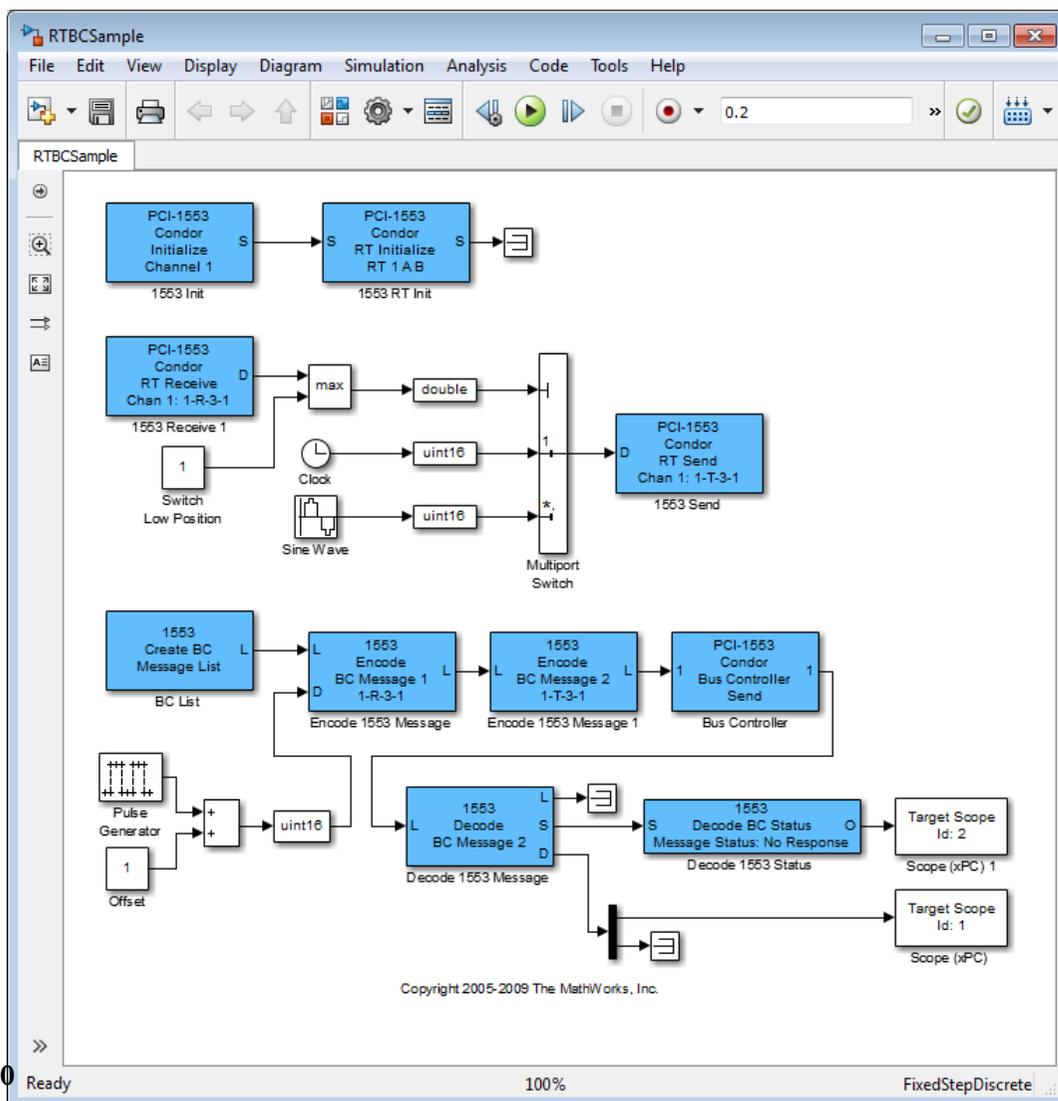
sent. You can also use the Encode BC Message block in an enabled subsystem to only send that message when desired.

The L signal is the message list, a custom data type consisting of a pointer, message length, and special marker. The model passes the L signal through the Encode BC Message blocks and from there to the Bus Controller Send block, executing the Encode blocks before the Send block.

## 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, RTSample and BCSample.

RTBCSample runs on a target computer 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 computer with two different PCI-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. 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.

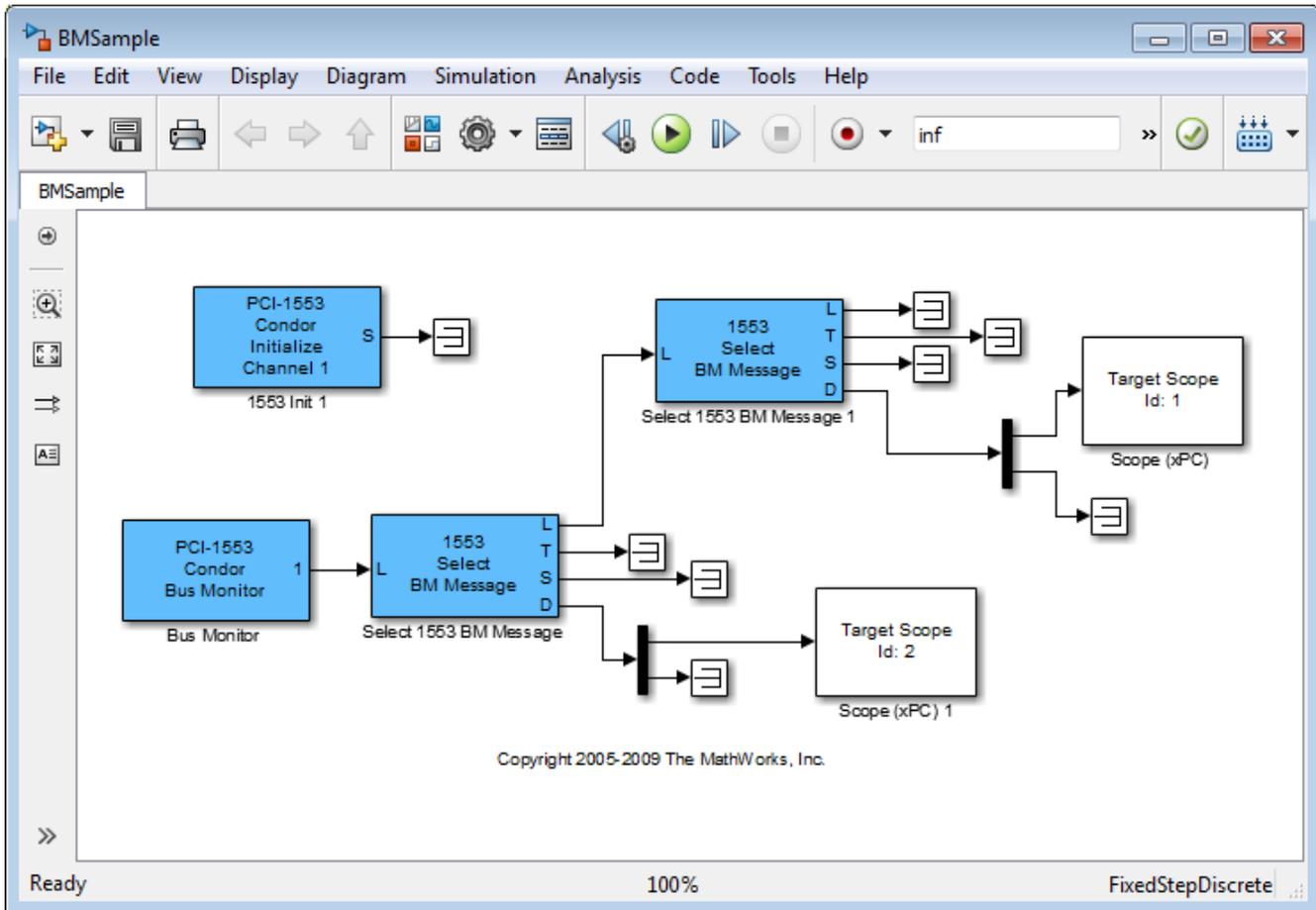
- **L** — Message list passed to other Decode BC Message blocks. If this is the only Decode BC Message block in your model, connect the signal to a terminator or ground.
- **S** — Status information. The Decode BC Status block extracts individual status bits from this status. To get multiple status bits, 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, BCSample, and RTBCSample).

Run this example on a target computer that has a PCI-1553 board. Replace the hardware specific blocks with the blocks for your hardware configuration.



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.

The 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 this is the only Select BM Message block in your model, 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

Condor 1553 Create BC Message List	Condor Create Bus Controller (BC) Message List block
Condor 1553 Decode BC Message	Condor 1553 Decode Bus Controller (BC) Message block
Condor 1553 Decode BC Status	Condor 1553 Decode Bus Controller (BC) Status block
Condor 1553 Encode BC Message	Condor 1553 Encode Bus Controller (BC) Message block
Condor 1553 Select BM Message	Condor Select 1553 Bus Monitor (BC) Message block

Condor PCI-1553 Bus Controller Send	Condor PCI-1553 Bus Controller (BC) Send block
Condor PCI-1553 Bus Monitor	Condor PCI-1553 Bus Monitor block
Condor PCI-1553 Initialize	Condor PCI-1553 Initialize block
Condor PCI-1553 RT Initialize	Condor PCI-1553 RT Initialize block
Condor PCI-1553 RT Receive	Condor PCI-1553 Remote Terminal (RT) Receive
Condor PCI-1553 RT Send	Condor PCI-1553 Remote Terminal (RT) Send block
Condor PCI/QPCI-1553	Support for MIL-STD-1553 PCI and QPCI boards.
Condor Q104-1553	Support for MIL-STD-1553 PC104 boards.
Condor Q104-1553 Bus Controller Send	Condor Q104-1553 Bus Controller (BC) Send block
Condor Q104-1553 Bus Monitor	Condor Q104-1553 Bus Monitor block
Condor Q104-1553 Initialize	Condor Q104-1553 Initialize block
Condor Q104-1553 RT Initialize	Condor Q104-1553 RT Initialize block
Condor Q104-1553 RT Receive	Condor Q104-1553 Remote Terminal (RT) Receive block
Condor Q104-1553 RT Send	Condor Q104-1553 Remote Terminal Send (RT) block
Condor QPCI-1553 Bus Controller Send	Condor QPCI-1553 Bus Controller (BC) Send block
Condor QPCI-1553 Bus Monitor	Condor QPCI-1553 Bus Monitor block
Condor QPCI-1553 Initialize	Condor QPCI-1553 Initialize block
Condor QPCI-1553 RT Initialize	Condor QPCI-1553 RT Initialize block

Condor QPCI-1553 RT Receive

Condor QPCI-1553 Remote Terminal  
(RT) Receive block

Condor QPCI-1553 RT Send

Condor QPCI-1553 Remote Terminal  
Send (RT) block

## SBS Technologies Support

xPC Target supports IP carrier, analog and digital I/O, synchro/resolver, and shared memory boards manufactured by GE Intelligent Platforms (<http://defense.ge-ip.com/products/family/avionics>) – formerly SBS Technologies.

SBS 2510/2500 Broadcast Memory	Support for the SBS 2510/2500 Broadcast Memory board
SBS Flex-104A	Flex-104A block
SBS Flex/104A PC/104 IP Carrier Board	Support for the SBS Flex/104A PC/104 IP Carrier board
SBS IP-16ADC	Support for the SBS IP-16ADC board
SBS IP-16ADC Analog Input (A/D)	IP-16ADC Analog Input block
SBS IP-16DAC	Support for the SBS IP-16DAC I/O board
SBS IP-16DAC Analog Output (D/A)	IP-16DAC Analog Output block
SBS IP-DAC	Support for the SBS IP-DAC I/O board
SBS IP-DAC Analog Output (D/A)	IP-DAC Analog Output block
SBS IP-Digital 24	Support for the SBS IP-Digital 24 I/O board
SBS IP-Digital 24 Digital Input	IP-Digital 24 Digital Input block
SBS IP-Digital 24 Digital Output	IP-Digital 24 Digital Output block
SBS IP-HiADC	Support for the SBS IP-HiADC board
SBS IP-HiADC Analog Input (A/D)	IP-HiADC Analog Input block
SBS IP-Synchro	Support for the SBS IP-Synchro board
SBS IP-Synchro/Resolver	IP-Synchro/Resolver block
SBS IP-UD-IHV-16I80	Support for the SBS IP-UD-IHV-16I80 board

SBS IP-UD-IHV-16I80 Digital Input	SBS IP-UD-IHV-16I80 Digital Input block
SBS IP-UD-IHV-16I80 Digital Output	SBS IP-UD-IHV-16I80 Digital Output block
SBS IP-UD-IHV-8I160	Support for the SBS IP-UD-IHV-8I160 board
SBS IP-UD-IHV-8I160 Digital Input	SBS IP-UD-IHV-8I160 Digital Input block
SBS IP-UD-IHV-8I160 Digital Output	SBS IP-UD-IHV-8I160 Digital Output block
SBS IP-Unidig-E-48 Digital Input	IP-Unidig-E-48 Digital Input block
SBS IP-Unidig-E-48 Digital Output	IP-Unidig-E-48 Digital Output block
SBS P-Unidig-E-48	Support for the SBS P-Unidig-E-48 board
SBS PCI-40A	PCI-40A block
SBS PCI-40A Carrier Board	Support for the SBS PCI-40A Carrier board
SBS PCI-40B	PCI-40B block
SBS PCI-40B Carrier Board	Support for the SBS PCI-40B Carrier board
SBS25x0 init	SBS25x0 init block
SBS25x0 read	SBS25x0 read block
SBS25x0 write	SBS25x0 write block

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

# Condor CEI-520/520A

---

**Purpose** Support for ARINC 429 CEI-520/5201A board.

**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 topic assumes that you are familiar with the ARINC 429 standard.

The xPC Target software supports the ARINC 429 protocol with the following boards. These board interface a target computer to an ARINC 429 data bus. These boards support the PCI bus.

- CEI-520
- CEI-520A

The xPC Target block library 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

## Board Characteristics

Board name	CEI-520, CEI-520A
Manufacturer	GE Fanuc (formerly Condor Engineering)
Bus type	PCI
Access method	Memory mapped
Multiple block instance support	No
Multiple board support	Yes

# Condor CEI-x20 Initialize

---

**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 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 computer, enter

-1

to automatically locate the board.

If two or more boards of this type are in the target computer, enter the bus number and the PCI slot number of 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 CEI-x20 Send

---

**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

---

**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.

# Condor CEI-x20 Receive

---

**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.

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 accepts data as a double. 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) — Scale the input by dividing it by the scale vector in the <b>Resolution (BCD) or Scale (BNR) vector</b> parameter. This input value is restricted to the range $[-Scale, Scale]$ , resulting in a scaled value in the range $[-1, 1]$ .

# Condor Encode ARINC 429 Words for Send

Type	Interpretation
	<p>The driver performs the following on the scaled value:</p> <ol style="list-style-type: none"> <li><b>1</b> Multiplies the scaled value by <math>2^{18}</math>.</li> <li><b>2</b> Truncates the value to a 19-bit fixed-point integer, then masks it to the number of high-order bits specified by the <b>Bits to send</b> parameter.</li> <li><b>3</b> Shifts the result up 10 bits and inserts them into the 32-bit integer along with the <b>SSM</b>, <b>SDI</b>, and <b>Label</b> parameter values.</li> </ol> <p>To place discrete bits into unused positions, construct the 32-bit word and use it with RAW mode instead of the BNR mode.</p>
2	<p>BCD (binary coded decimal) — Cast the input as a signed integer, limit it to the range representable by an ARINC five-character BCD value, and pack it into an ARINC word. and set the required <b>SSM</b>, <b>SDI</b>, and <b>Label</b> parameter values.</p>
3	<p>Discretes — Cast the input as an unsigned 32-bit integer, pack the low order 19 bits of the result into an ARINC word, and set the required <b>SSM</b>, <b>SDI</b>, and <b>Label</b> parameters.</p>

## Resolution (BCD) or Scale (BNR) vector

Enter a vector or scalar value as the resolution vector. This value 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 interpretation of this value depends on the data type. The block works with the data types as follows.

## Condor Encode ARINC 429 Words for Send

---

Type	Effect
Raw	The block ignores any resolution value. However, you must still include an associated value in the resolution vector.
BNR	<p>The block uses this value as a scale factor. The block divides the input value by the scale. Doing so limits the valid input range to [-Scale, Scale]. Values outside that range will be limited to -Scale.</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 keeps it within the range.</p>
BCD	<p>The resolution vector specifies, in the same units as the input signal, the value of the least significant digit of the BCD data field to be encoded and sent. For example, if the associated resolution is .01 and the input signal contains the value 3.1415, the output ARINC word will contain the number 314 in its data field, encoded in BCD. Use the same resolution on the receive side to reconstruct 3.14. Resolution is typically a power of 10, but this is not a restriction.</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 limits it to be within the range.</p>
Discretes	The block ignores any resolution value. However, you must still include an associated value in the resolution vector.

# Condor Encode ARINC 429 Words for Send

## Bits to send (BNR)

Specifies the number of high order bits of the 19-bit scaled input to keep when the value is inserted into the encoded word. The block uses this value for a BNR format input; other formats ignore it. You must include a place in the array of Bits values for each entry in the data type vector array; however, the block uses the value only for BNR.

## 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 the SDI value. However, you must still include an associated value in the resolution vector.
BNR, BCD, Discretes	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.

## SSM vector

Enter a vector or scalar value as the SSM 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. The meaning of a given SSM value differs depending on the data type of the ARINC word.

## Condor Encode ARINC 429 Words for Send

---

Type	Effect
Raw	The block ignores the SSM value. However, you must still include an associated value in the resolution vector.
BNR, BCD, Discretes	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.

# 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.

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.

## 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.

# Condor Decode ARINC 429 Words from Receive

The elements of the vector determine how the input double value is converted to a corresponding double output value, as follows

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 a double output in the range [ -Scale, Scale] using the <b>Resolution (BCD) or Scale (BNR) vector</b> scale factor specified for this value.
2	BCD — For each word, convert bits 10–28 from BCD format to double, using the sign data in the SSM.
3	Disretes — 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

# Condor Decode ARINC 429 Words from Receive

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.

- The Decode block buffers its input internally. It updates its output port only during sample times when it has assembled at least one complete message of length  $n$  without detecting an error.

More than one complete message might be assembled during one sample time. The Decode block overwrites these messages such that only the most recent message is on the output port.

## Resolution (BCD) or Scale (BNR) vector

Enter a vector or scalar value as the resolution vector. This value 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 interpretation of this value depends on the data type. 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 block uses this value as a scale factor. The block converts the binary value in bits 10–28 back to a double in the range <math>[-1, 1]</math>. The block then multiplies that value by the scale value to recover the original value.</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 Decode ARINC 429 Words from Receive

Type	Effect
BCD	<p>The resolution vector specifies, in the same units as the input signal, the value of the least significant digit of the BCD data field to be encoded and sent. For example, if the associated resolution is .01 and the input signal contains the value 3.1415, the output ARINC word will contain the number 314 in its data field, encoded in BCD. Use same resolution on the receive side to reconstruct 3.14. Resolution is typically a power of 10, but this is not a restriction.</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 limits it to within the range.</p>
Discretes	<p>The block ignores any resolution value. However, you must still include an associated value in the resolution vector.</p>

## 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 1100000000 in binary. This value selects the SDI bits (bits 9 and 10) as the sync bits. This functionality works 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

# Condor Decode ARINC 429 Words from Receive

---

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 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 and the required 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.

# Condor Decode ARINC 429 Words from Receive

---

## 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.

# 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>	<p><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.</p> <p><b>Sample time</b> Enter the base sample time or a multiple of the base sample time.</p>
<b>See Also</b>	Condor 1553 Encode BC Message “Bus Controller Operation” on page 31-7

# Condor 1553 Decode BC Message

---

**Purpose** Condor 1553 Decode Bus Controller (BC) Message block

**Library** xPC Target Library for MIL-STD-1553

**Description** Each instance of this block extracts information out of one message in the receive list. This block has three outputs.

L — Specifies the input and output message list passed to other Decode BC Message blocks. If this is the only Decode BC Message block in your model, connect the signal to a terminator or ground.

S — The S output is a vector of six elements from the received message, all bit packed with information:

`control1, command1, status1, command2, status2, overall status`

These elements derive from the `API_BC_MBUF` structure, which is contained in the xPC Target file `matlab\toolbox\rtw\targets\xpc\target\build\xpcblocks\include\busapi.h`. See the GE Fanuc (formerly Condor Engineering) 1553 software user documentation for details on this structure.

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.

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.

## Block Parameters

**Message number**  
Enter the number of the message to decode.

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

# 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.
<b>Block Input and Outputs</b>	<p>This block has the following inputs and outputs:</p> <p>L — The passed through message list</p> <p>T — Time in microseconds at which this message was received</p> <p>S — Status (seven component vector with contents)</p> <ul style="list-style-type: none"><li>• 1 message number</li><li>• 2 cmd1</li><li>• 3 resp1</li><li>• 4 status 1</li><li>• 5 cmd 2</li><li>• 6 resp 2</li><li>• 7 status 2</li></ul> <p>Elements cmd1 and cmd2 contain address information in bit fields in the 16-bit integer.</p> <p>cmdN=&lt;RRRRR&gt;T&lt;SSSSS&gt;&lt;CCCCC&gt;</p> <p>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.</p> <p>Elements 5, 6, and 7 are only nonzero for the RT-&gt;RT messages.</p> <p>D — 32 uint16 vector data. cmd 1 has the length of real data in the low order 5 bits.</p>

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

**Purpose**

Support for MIL-STD-1553 PCI and QPCI boards.

**Board**

Condor PCI/QPCI-1553

**General Description**

The MIL-STD-1553 driver library allows xPC Target applications to connect to a MIL-STD-1553 bus network to send and receive messages of up to 32 16-bit words. This topic assumes that you are familiar with the MIL-STD-1553 standard.

The xPC Target block library supports the MIL-STD-1553 protocol with the following boards. These boards interface a target computer to a MIL-STD-1553 data bus via the PCI and high-density PCI (QPCI) buses, respectively.

- **PCI-1553** — This is available with 1 or 2 channels as either a single function or multifunction configuration. The multifunction 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.
- **QPCI-1553** — This board is available with 1, 2, or 4 channels as either a single function or multifunction configuration. The multifunction 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.

The block library 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/QPCI-1553

---

- 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	GE Fanuc (formerly Condor Engineering)
Bus type	PCI
Multiple board support	Yes

# 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. Daisy chain the Encode BC Message block L signals so that the Encode BC blocks execute before the Bus Controller Send block.

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). This value should not exceed the sample time. Exceeding the sample time triggers 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 computer, enter

-1

to automatically locate the board.

If two or more boards of this type are in the target computer, enter the bus number and the PCI slot number of 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 computer, enter

-1

to automatically locate the board.

If two or more boards of this type are in the target computer, enter the bus number and the PCI slot number of 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

---

<b>Purpose</b>	Condor PCI-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 or 2. This is the channel to initialize on the board.</p> <p><b>Coupling mode</b> From the list, select from Direct Coupling and Transformer Coupling. This defines the mode of coupling a terminal device to the bus.</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 parameters:</p> <p><b>Number of Bus Controller buffers to allocate</b> 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 is given to process.</p> <p><b>Enable retries</b> Select this check box to enable automatic retries when a message times out or has an error.</p> <p><b>No response timeout (microseconds)</b> 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.</p> <p><b>Late response timeout (microseconds)</b> 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,</p>

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.

# Condor PCI-1553 Initialize

---

## PCI Slot (-1:autosearch)

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

-1

to automatically locate the board.

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

```
getxpcpci
```

<b>Purpose</b>	Condor PCI-1553 RT Initialize block
<b>Library</b>	xPC Target Library for MIL-STD-1553
<b>Description</b>	<p>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</p> <ul style="list-style-type: none"><li>• Main Condor PCI-1553 Initialize block for this board</li><li>• Another Condor PCI-1553 RT Initialize block that is connected to the main Condor PCI-1553 Initialize block</li></ul> <p>This connection carries the channel number and PCI slot information.</p>
<b>Block Parameters</b>	<p><b>Remote Terminal</b> From the list, choose a Remote Terminal from 1 to 31. Choose the terminal number that this initialization block is configuring.</p> <p><b>Connect to bus A</b> 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.</p> <p><b>Connect to bus B</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.</p> <p><b>Initial status</b> 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.</p> <p><b>Initial BIT word</b> 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.</p>

# Condor PCI-1553 RT Initialize

---

## **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 accepts a message only 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. Construct this as follows:

- Each element of the vector corresponds to the corresponding element of the **Transmit sub addresses** vector.
- Each element of the transmit message length vector corresponds to the corresponding element of the **Legal receive message lengths**. Failure to do this prevents the subaddress on this Remote Terminal from responding.

A message can have a length of from 1 to 32 words, selected by setting one or more bits in a 32-bit mask. For instance:

Bit Position	Mask Value (hex)	Word Length (words)
0	0x00000001	32
1	0x00000002	1
2	0x00000004	2
...		
30	0x40000000	30
31	0x80000000	31

---

## Note

- Use the `hex2dec` function to specify the bit mask in hexadecimal.
  - The message length is expressed by bit position in the mask, not by mask value. The value `0xffffffff` enables all message lengths from 1 to 32.
  - 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 accepts a message only 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.

# Condor PCI-1553 RT Initialize

---

## 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.

Construct this vector as follows:

- Each element of the vector corresponds to the corresponding element of the **Transmit sub addresses** vector.
- Each element of the vector corresponds to the corresponding element of the **Receive sub addresses** vector.
- Each element of the receive message lengths vector corresponds to the corresponding element of the **Legal transmit message lengths**. Failure to do this prevents the subaddress on this Remote Terminal from responding.

A message can have a length of from 1 to 32 words, selected by setting one or more bits in a 32-bit mask. For instance:

Bit Position	Mask Value (hex)	Word Length (words)
0	0x00000001	32
1	0x00000002	1
2	0x00000004	2
. . .		
30	0x40000000	30
31	0x80000000	31

---

**Note**

- Use the `hex2dec` function to specify the bit mask in hexadecimal.
  - The message length is expressed by bit position in the mask, not by mask value. The value `0xffffffff` enables all message lengths from 1 to 32.
  - 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 channel number and address parameters. The address 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.

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 computer, enter

-1

to automatically locate the board.

If two or more boards of this type are in the target computer, enter the bus number and the PCI slot number of 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 sends only 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 channel number and address parameters. The address 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. Initialize 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 computer, enter

-1

to automatically locate the board.

If two or more boards of this type are in the target computer, enter the bus number and the PCI slot number of 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 Q104-1553

---

**Purpose** Support for MIL-STD-1553 PC104 boards.

**Board** Condor Q104-1553

**General Description** The MIL-STD-1553 driver library allows xPC Target applications to connect to a MIL-STD-1553 bus network to send and receive messages of up to 32 16-bit words. This topic assumes that you are familiar with the MIL-STD-1553 standard.

The xPC Target block library supports the MIL-STD-1553 protocol via the PC104 bus. The Q104-1553 board is available with 1, 2, or 4 channels as either a single function or multifunction configuration. The multifunction 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.

The block library provides the following driver blocks to support these boards:

- Condor Q104-1553 Bus Controller Send
- Condor Q104-1553 Bus Monitor
- Condor Q104-1553 Initialize
- Condor Q104-1553 RT Initialize
- Condor Q104-1553 RT Receive
- Condor Q104-1553 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	Q104-1553
Manufacturer	GE Fanuc (formerly Condor Engineering)
Bus type	PC104
Multiple board support	Yes

# Condor Q104-1553 Bus Controller Send

---

**Purpose** Condor Q104-1553 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. Daisy chain the Encode BC Message block L signals so the Encode BC blocks execute before the Bus Controller Send block.

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, 2, 3, or 4. This is the channel for this command stream.

**Response mode**  
Choose from

# Condor Q104-1553 Bus Controller Send

---

- **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). This value should not exceed the sample time. Exceeding the sample time triggers an execution overload.

## **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

# Condor Q104-1553 Bus Monitor

---

<b>Purpose</b>	Condor Q104-1553 Bus Monitor block
<b>Library</b>	xPC Target Library for MIL-STD-1553
<b>Block Parameters</b>	<b>Channel</b> From the list, choose 1, 2, 3, or 4.
	<b>Maximum number of messages to receive</b> Enter the maximum number of messages this block should read from the board.
	<b>Sample time</b> Enter the base sample time or a multiple of the base sample time.
	<b>Base address</b> 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**

Condor Q104-1553 Initialize block

**Library**

xPC Target Library for MIL-STD-1553

**Block Parameters****Channel**

From the list, select from 1, 2, 3, or 4. 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 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 is 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

# Condor Q104-1553 Initialize

---

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.

## **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

<b>Purpose</b>	Condor Q104-1553 RT Initialize block
<b>Library</b>	xPC Target Library for MIL-STD-1553
<b>Description</b>	<p>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</p> <ul style="list-style-type: none"><li>• Main Condor Q104-1553 Initialize block for this board</li><li>• Another Condor Q104-1553 RT Initialize block that is connected to the main Condor Q104-1553 Initialize block</li></ul> <p>This connection carries the channel number and PC104 base address.</p>
<b>Block Parameters</b>	<p><b>Remote Terminal</b> From the list, choose a Remote Terminal from 1 to 31. Choose the terminal number that this initialization block is configuring.</p> <p><b>Connect to bus A</b> 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.</p> <p><b>Connect to bus B</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.</p> <p><b>Initial status</b> 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.</p> <p><b>Initial BIT word</b> 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.</p>

# Condor Q104-1553 RT Initialize

---

## **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 accepts a message only 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. Construct this vector as follows:

- Each element of the vector corresponds to an element of the **Transmit sub addresses** vector.
- Each element of the vector of transmit message lengths corresponds to an element of the **Legal receive message lengths** vector. Failure to do this prevents the subaddress on this Remote Terminal from responding.

A message can have a length of from 1 to 32 words, selected by setting one or more bits in a 32-bit mask. For instance:

Bit Position	Mask Value (hex)	Word Length (words)
0	0x00000001	32
1	0x00000002	1
2	0x00000004	2
...		
30	0x40000000	30
31	0x80000000	31

---

## Note

- Use the `hex2dec` function to specify the bit mask in hexadecimal.
  - The message length is expressed by bit position in the mask, not by mask value. The value `0xffffffff` enables all message lengths from 1 to 32.
  - 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 accepts a message only 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.

# Condor Q104-1553 RT Initialize

---

## 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. Construct this vector as follows:

- Each element of the vector corresponds to the corresponding element of the **Transmit sub addresses** vector.
- Each element of the vector corresponds to the corresponding element of the **Receive sub addresses** vector.
- Each element of the receive message lengths vector corresponds to the corresponding element of the **Legal transmit message lengths**. Failure to do this prevents the subaddress on this Remote Terminal from responding.

A message can have a length of from 1 to 32 words, selected by setting one or more bits in a 32-bit mask. For instance:

Bit Position	Mask Value (hex)	Word Length (words)
0	0x00000001	32
1	0x00000002	1
2	0x00000004	2
...		
30	0x40000000	30
31	0x80000000	31

---

**Note**

- Use the `hex2dec` function to specify the bit mask in hexadecimal.
  - The message length is expressed by bit position in the mask, not by mask value. The value `0xffffffff` enables all message lengths from 1 to 32.
  - 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 Q104-1553 RT Receive

---

**Purpose** Condor Q104-1553 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 in the following 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.

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.

**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

# Condor Q104-1553 RT Send

---

**Purpose** Condor Q104-1553 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 sends only 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 in the following 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. Initialize this channel with the Q104-1553 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.

**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

# Condor QPCI-1553 Bus Controller Send

---

**Purpose** Condor QPCI-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. Daisy chain the Encode BC Message block L signals so the Encode BC blocks execute before the Bus Controller Send block.

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, 2, 3, or 4. This is the channel for this command stream.

## 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). This value should not exceed the sample time. Exceeding the sample time triggers 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 computer, enter

-1

to automatically locate the board.

If two or more boards of this type are in the target computer, enter the bus number and the PCI slot number of 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 Monitor

---

<b>Purpose</b>	Condor QPCI-1553 Bus Monitor block
<b>Library</b>	xPC Target Library for MIL-STD-1553
<b>Block Parameters</b>	<b>Channel</b> From the list, choose 1, 2, 3, or 4.
	<b>Maximum number of messages to receive</b> Enter the maximum number of messages this block should read from the board.
	<b>Sample time</b> Enter the base sample time or a multiple of the base sample time.
	<b>PCI Slot (-1:autosearch)</b> If only one board of this type is in the target computer, enter  -1  to automatically locate the board.  If two or more boards of this type are in the target computer, enter the bus number and the PCI slot number 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>

<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 test bus 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>Coupling mode</b> From the list, select from Direct Coupling and Transformer Coupling. This defines the mode of coupling a terminal device to the bus.</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 parameters:</p> <p><b>Number of Bus Controller buffers to allocate</b> 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 is given to process.</p> <p><b>Enable retries</b> Select this check box to enable automatic retries when a message times out or has an error.</p> <p><b>No response timeout (microseconds)</b> 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.</p>

# Condor QPCI-1553 Initialize

---

## **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 computer, enter

-1

to automatically locate the board.

If two or more boards of this type are in the target computer, enter the bus number and the PCI slot number of 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 accepts a message only 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. Construct this vector as follows:

- Each element of the vector corresponds to an element of the **Transmit sub addresses** vector.
- Each element of the vector of transmit message lengths corresponds to an element of the **Legal receive message lengths** vector. Failure to do this prevents the subaddress on this Remote Terminal from responding.

A message can have a length of from 1 to 32 words, selected by setting one or more bits in a 32-bit mask. For instance:

# Condor QPCI-1553 RT Initialize

---

Bit Position	Mask Value (hex)	Word Length (words)
0	0x00000001	32
1	0x00000002	1
2	0x00000004	2
...		
30	0x40000000	30
31	0x80000000	31

---

## Note

- Use the hex2dec function to specify the bit mask in hexadecimal.
  - The message length is expressed by bit position in the mask, not by mask value. The value 0xffffffff enables all message lengths from 1 to 32.
  - 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 accepts a message only 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. Construct this vector as follows:

- Each element of the vector corresponds to the corresponding element of the **Transmit sub addresses** vector.
- Each element of the vector corresponds to the corresponding element of the **Receive sub addresses** vector.
- Each element of the receive message lengths vector corresponds to the corresponding element of the **Legal transmit message lengths**. Failure to do this prevents the subaddress on this Remote Terminal from responding.

A message can have a length of from 1 to 32 words, selected by setting one or more bits in a 32-bit mask. For instance:

Bit Position	Mask Value (hex)	Word Length (words)
0	0x00000001	32
1	0x00000002	1
2	0x00000004	2
. . .		
30	0x40000000	30
31	0x80000000	31

# Condor QPCI-1553 RT Initialize

---

---

## Note

- Use the `hex2dec` function to specify the bit mask in hexadecimal.
  - The message length is expressed by bit position in the mask, not by mask value. The value `0xffffffff` enables all message lengths from 1 to 32.
  - 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.

**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 channel number and address parameters. The address 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.

# Condor QPCI-1553 RT Receive

---

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 computer, enter

-1

to automatically locate the board.

If two or more boards of this type are in the target computer, enter the bus number and the PCI slot number of 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 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 sends only 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 channel number and address parameters. The address 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. Initialize 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.

# Condor QPCI-1553 RT Send

---

## **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 computer, enter

-1

to automatically locate the board.

If two or more boards of this type are in the target computer, enter the bus number and the PCI slot number of 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** Support for GE Fanuc PCI-5565 high-speed fiber optic reflective memory boards.

**Board** GE Fanuc PCI-5565

**General Description** The VMIPCI-5565 and PCI-5565PIORC are high-speed fiber optic reflective memory boards. They can also generate/broadcast interrupts. The xPC Target software uses these board as part of the shared memory network that you can use to exchange data between computer nodes.

The xPC Target block library supports these boards with these driver blocks:

- GE Fanuc 5565 broadcast
- GE Fanuc 5565 init
- GE Fanuc 5565 read
- GE Fanuc 5565 write

For information about the Change endianness block, see Byte Reversal/Change Endianness.

<b>Board Characteristics</b>	Board name	VMIPCI-5565, PCI-5565PIORC
	Manufacturer	GE Fanuc
	Bus type	PCI
	Access method	Memory mapped
	Multiple block instance support	Yes
	Multiple board support	Yes

# GE Fanuc 5565 broadcast

---

**Purpose** GE Fanuc PCI-5565 broadcast interrupt block

**Library** xPC Target Library for GE Fanuc

**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** The xPC Target software 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. You cannot broadcast the interrupt to any other subset of nodes.

### Interrupt Number

Enter the shared IRQ number. This value is the special interrupt number that two boards use between each other for synchronization (see the LIER register bit descriptions in the PCI-5565 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 17-9).

node.Interface.Interrupts.PendingIntX

## **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 computer, enter

-1

to automatically locate the board.

If two or more boards of this type are in the target computer, enter the bus number and the PCI slot number 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

# GE Fanuc 5565 init

---

**Purpose** GE Fanuc PCI-5565 initialization block

**Library** xPC Target Library for GE Fanuc

**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 17-4.

Each model that uses shared memory must have one 5565 init block for every PCI-5565 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 computer, enter

-1

to automatically locate the board.

If two or more boards of this type are in the target computer, enter the bus number and the PCI slot number of 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** GE Fanuc PCI-5565 read block

**Library** xPC Target Library for GE Fanuc

**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 17-2.

## **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 computer, enter

-1

to automatically locate the board.

If two or more boards of this type are in the target computer, enter the bus number and the PCI slot number of 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 PCI-5565 board LISR register modes.

# GE Fanuc 5565 write

---

**Purpose** GE Fanuc PCI-5565 write block

**Library** xPC Target Library for GE Fanuc

**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 17-2.

## **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 computer, enter

-1

to automatically locate the board.

If two or more boards of this type are in the target computer, enter the bus number and the PCI slot number of 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 PCI-5565 board LISR register modes.

# SBS 2510/2500 Broadcast Memory

---

**Purpose** Support for the SBS 2510/2500 Broadcast Memory board

**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. The xPC Target software uses this board as part of the shared memory network that you can use to exchange data between computer nodes.

The xPC Target block library 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.

<b>Board Characteristics</b>	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 17-16.

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 computer, enter

-1

to automatically locate the board.

If two or more boards of this type are in the target computer, enter the bus number and the PCI slot number of 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 17-13.

**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 computer, enter

-1

to automatically locate the board.

If two or more boards of this type are in the target computer, enter the bus number and the PCI slot number of 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 17-13.

**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 computer, enter

-1

to automatically locate the board.

If two or more boards of this type are in the target computer, enter the bus number and the PCI slot number of 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

---

**Purpose** Support for the 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. If your model contains multiple carriers, be sure to use a unique ID for each carrier. 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.

The xPC Target block library supports this board with this driver block:

- SBS Flex-104A

<b>Board Characteristics</b>	Board name	Flex-104A PC/104 carrier board
	Manufacturer	GE Intelligent Platforms (formerly 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 Parameters** **Carrier ID**

Enter a number to uniquely identify this IP carrier board within the model. If your model contains multiple carriers, be sure to use a unique ID for each carrier.

**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

<b>Expected Pin</b>	<b>Actual Pin</b>	<b>Expected Pin</b>	<b>Actual Pin</b>	<b>Expected Pin</b>	<b>Actual Pin</b>
13	25	30	10	47	44
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

---

**Purpose** Support for the SBS IP-16ADC board

**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.

The xPC Target block library supports this board with this driver block:

- SBS IP-16ADC Analog Input (A/D)

---

**Note** To operate the IP-16ADC I/O board, connect an external  $\pm 15$  volt power supply to the I/O interface of the board. Failure to do so prevents the conversion.

---

## Board Characteristics

Board name	IP-16ADC
Manufacturer	GE Intelligent Platforms (formerly 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.

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. 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.

**Purpose** Support for the SBS IP-16DAC I/O board

**Board** SBS IP-16DAC

**General Description** The IP-16DAC I/O board has 3 independent precalibrated analog output (D/A) channels (16-bit).

The xPC Target block library supports this board with this driver block:

- SBS IP-16DAC Analog Output (D/A)

<b>Board Characteristics</b>	Board name	IP-16DAC
	Manufacturer	GE Intelligent Platforms (formerly 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.

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. 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

---

**Purpose** Support for the SBS IP-DAC I/O board

**Board** SBS IP-DAC

**General Description** The IP-DAC I/O board has 6 independent precalibrated analog output (D/A) channels (12-bit).

The xPC Target block library supports this board with this driver blocks:

- SBS IP-DAC Analog Output (D/A)

## Board Characteristics

Board name	IP-DAC
Manufacturer	GE Intelligent Platforms (formerly 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.

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. 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.

**Purpose** Support for the SBS IP-Digital 24 I/O board

**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.  
The xPC Target block library 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	GE Intelligent Platforms (formerly 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.

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. 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.

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.

**Purpose** Support for the SBS IP-HiADC board

**Board** SBS IP-HiADC

**General Description** The IP-HiADC I/O board has 16 analog input (A/D) channels (12-bit).  
The xPC Target block library supports this board with this driver block:

- SBS IP-HiADC Analog Input (A/D)

<b>Board Characteristics</b>	Board name	IP-HiADC
	Manufacturer	GE Intelligent Platforms (formerly 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.

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. 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

---

**Purpose** Support for the SBS IP-Synchro board

**Board** SBS IP-Synchro

**General Description** IP-Synchro provides two channels of position measurement using synchro, resolver, LVDT, or Inductosyn® transducers.

The xPC Target block library supports this board with this driver block:

- SBS IP-Synchro

<b>Board Characteristics</b>	Board name	IP-Synchro
	Manufacturer	GE Intelligent Platforms (formerly 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.

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. 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) 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.

**Purpose** Support for the SBS IP-UD-IHV-16I80 board

**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.

The xPC Target block library 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-16180

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	GE Intelligent Platforms (formerly 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

## Scaling Input to Output

Hardware Input	Block Output Data Type	Scaling
Volts	Double ( <b>Format:16 1-bit Channels</b> )	Double: < Threshold = 0.0 > Threshold = 1.0
	uint8 ( <b>Format:Two 8-bit Channels</b> )	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. 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

## Scaling of Output to Input

Hardware Input	Block Output Data Type	Scaling
Volts	Double ( <b>Format:8</b> 1-bit Channels)	Double: < 0.5, off >0.5, on
	uint8 ( <b>Format:One</b> 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

Select the slot on the carrier board into which the module IP-UD-IHV-16I80 is plugged. 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 Channels**  
Configures block to accept one 8-bit channels.

# SBS IP-UD-IHV-16I80 Digital Output

---

## Channel vector

This value depends on the value of **Format**, as follows:

- Vector of numbers between 1 and 16

If you select the **Format** parameter to be **8 1 bit 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 8 bit 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 1 bit 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.

To enable double-buffering, you must have an external clock attached to the board. See the IP-UD-IHV user documentation for details.

# SBS IP-UD-IHV-16180 Digital Output

---

## Sample time

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

**Purpose** Support for the SBS IP-UD-IHV-8I160 board

**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.

The xPC Target block library 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 IP-UD-IHV-8I160

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

**Board  
Characteristics**

Board name	SBS IP-UD-IHV-8I160
Manufacturer	GE Intelligent Platforms (formerly 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

## Scaling Input to Output

Hardware Input	Block Output Data Type	Scaling
Volts	Double ( <b>Format:8</b> 1-bit Channels)	Double: < Threshold = 0.0 > Threshold = 1.0
	uint8 ( <b>Format:One</b> 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. 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

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 1 bit 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 8 bit 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 1 bit 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.

# SBS IP-UD-IHV-81160 Digital Input

---

## 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	Block Output Data Type	Scaling
Volts	Double ( <b>Format:16</b> 1-bit Channels)	Double: < 0.5, off >0.5, on
	uint8 ( <b>Format:Two</b> 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

Select the slot on the carrier board into which the module IP-UD-IHV-16I80 is plugged. 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 two 8-bit channels.

### Channel vector

This value depends on the value of Format, as follows:

# SBS IP-UD-IHV-8I160 Digital Output

---

- 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 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 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.

## **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.

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 P-Unidig-E-48

---

**Purpose** Support for the SBS P-Unidig-E-48 board

**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.  
The xPC Target block library 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	GE Intelligent Platforms (formerly SBS Technologies)
	Bus type	N/A
	Access method	I/O Mapped
	Multiple block instance support	Yes
	Multiple board support	Yes

**Purpose** IP-Unidig-E-48 Digital Input block

**Library** The xPC Target block Library for SBS Technologies

## Scaling Input to Output

Hardware Input	Block Output Data Type	Scaling
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.

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. 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** The xPC Target block 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.

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.

# SBS PCI-40A Carrier Board

---

**Purpose** Support for the SBS PCI-40A Carrier board

**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. If your model contains multiple carriers, be sure to use a unique ID for each carrier. 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. The xPC Target block library supports this board with this driver block:

- SBS PCI-40A

<b>Board Characteristics</b>	Board name	PCI-40A carrier board
	Manufacturer	GE Intelligent Platforms (formerly SBS Technologies)
	Bus type	PCI
	Access method	I/O Mapped
	Multiple block instance support	No
	Multiple board support	Yes

**Purpose** PCI-40A block

**Library** xPC Target block Library for SBS Technologies

**Block  
Parameters**

**Carrier ID**

Enter a number to uniquely identify this IP carrier board within the model. If your model contains multiple carriers, be sure to use a unique ID for each carrier.

**PCI Slot (-1:autosearch)**

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

-1

to automatically locate the board.

If two or more boards of this type are in the target computer, enter the bus number and the PCI slot number of 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 PCI-40B Carrier Board

---

**Purpose** Support for the SBS PCI-40B Carrier board

**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. If your model contains multiple carriers, be sure to use a unique ID for each carrier. 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. The xPC Target block library supports this board with this driver block:

- SBS PCI-40B

<b>Board Characteristics</b>	Board name	PCI-40B carrier board
	Manufacturer	GE Intelligent Platforms (formerly SBS Technologies)
	Bus type	PCI
	Multiple block instance support	No
	Multiple board support	Yes

**Purpose**

PCI-40B block

**Library**

xPC Target Library for SBS Technologies

**Block  
Parameters****Carrier ID**

Enter a number to uniquely identify this IP carrier board within the model. If your model contains multiple carriers, be sure to use a unique ID for each carrier.

**PCI Slot (-1:autosearch)**

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

-1

to automatically locate the board.

If two or more boards of this type are in the target computer, enter the bus number and the PCI slot number of 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

---

This topic describes General Standards I/O boards supported by the xPC Target product (<http://www.generalstandards.com>).

“PMC-ADADIO Functionality” on page 32-2	Describes how the PMC-ADADIO blocks can be configured to interact with one another and other blocks.
“Overview of Audio Applications” on page 32-17	Describes how the PMC-24DSI12 and PMC66-16AO16 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.
General Standards PMC66-16AO16	High speed analog output board.
General Standards PMC-24DSI12	High speed analog input board.
“Boards and Blocks — Alphabetical List” on page 32-18	Description of block parameters for General Standards driver blocks.

## PMC-ADADIO Functionality

### In this section...

“PMC-ADADIO Basics” on page 32-2

“A/D Blocks” on page 32-3

“Create Enable Signal Blocks” on page 32-6

“D/A Blocks” on page 32-9

“Interleaving Analog Input and Output Blocks” on page 32-11

“Using Multiple PMC-ADADIO Boards” on page 32-13

### PMC-ADADIO Basics

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.

The xPC Target block library 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.

- Specifies the order in which A/D and D/A blocks execute. 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.

## 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**.

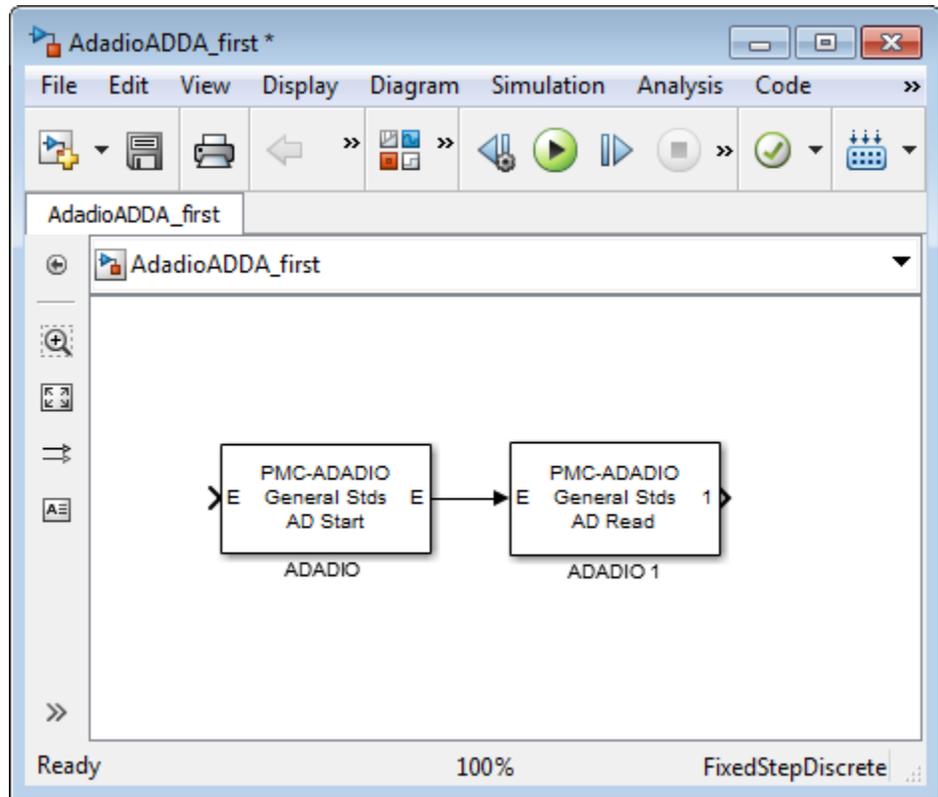
- 5** Drag and drop an AD Start block from the General Standards window to the new model.

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'.

---

**Tip**

- 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.
  - Drag and drop a Terminator block from the Simulink Sink library and connect that block to the unconnected output ports of AD Read to prevent build errors until you add another block.
- 
- Connecting the output enable port of the AD Start block to the input enable port of the AD Read block causes the AD Start block to execute 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 32-7.

## **Create Enable Signal Blocks**

The A/D and D/A series of blocks both have Create Enable Signal blocks. You can use these blocks 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 32-3). 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.

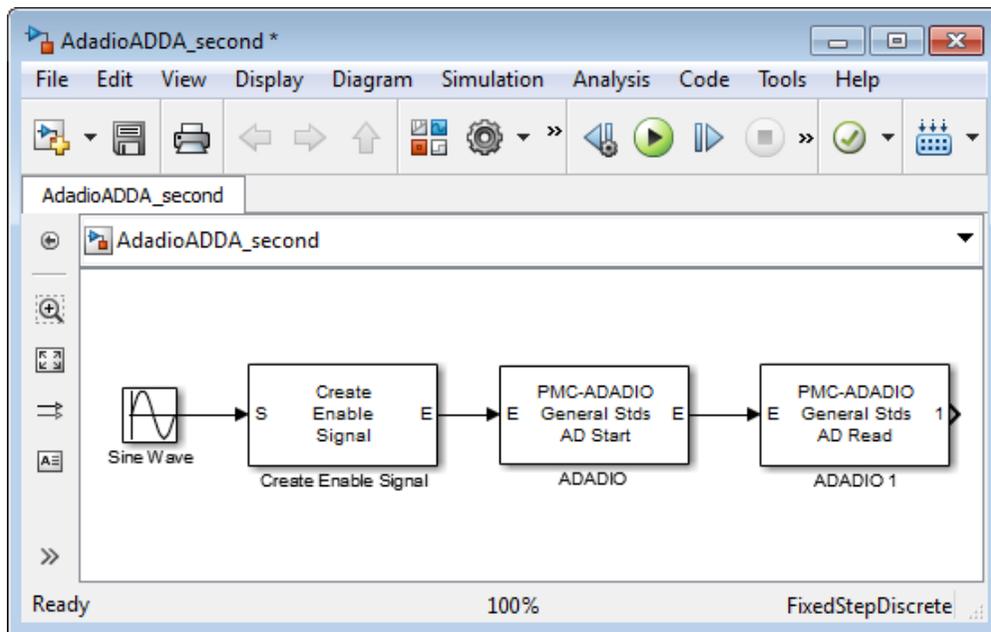
- 5 Drag and drop a Create Enable Signal block from the General Standards window to the new model.

- 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 In the Simulink dialog box, click **View > Library Browser**.
  - 8 Click node **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.

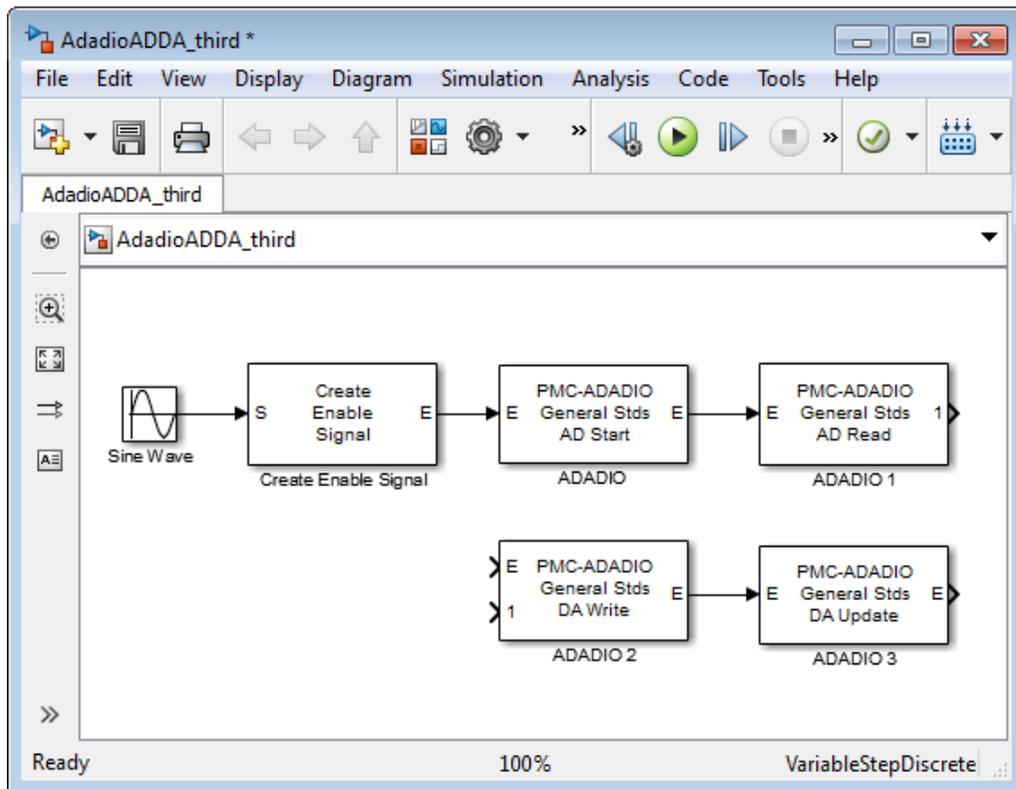
- 4** Double-click the General Standards group block.

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 Write block to the DA Update 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 causes the DA Write block to execute 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 Output Blocks” on page 32-11.

## Interleaving Analog Input and 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 specify their order of execution. See the following enable line for the execution order:

- 1 AD Start enable out to DA Write enable in
- 2 DA Write enable out to DA Update enable in
- 3 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

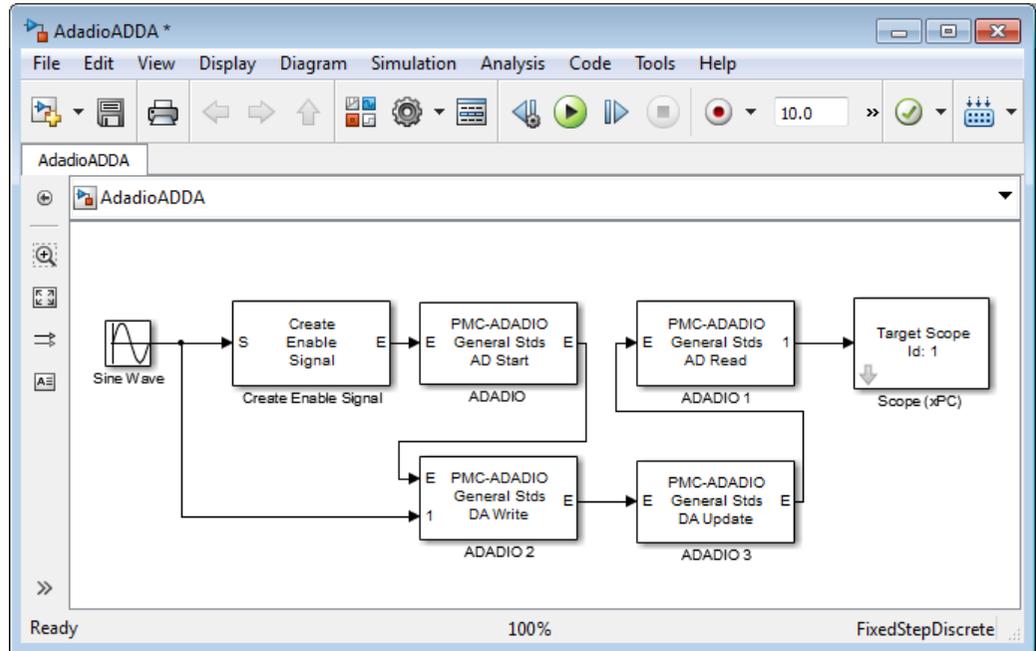
```
xpclib
```

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 computer like any other xPC Target model.

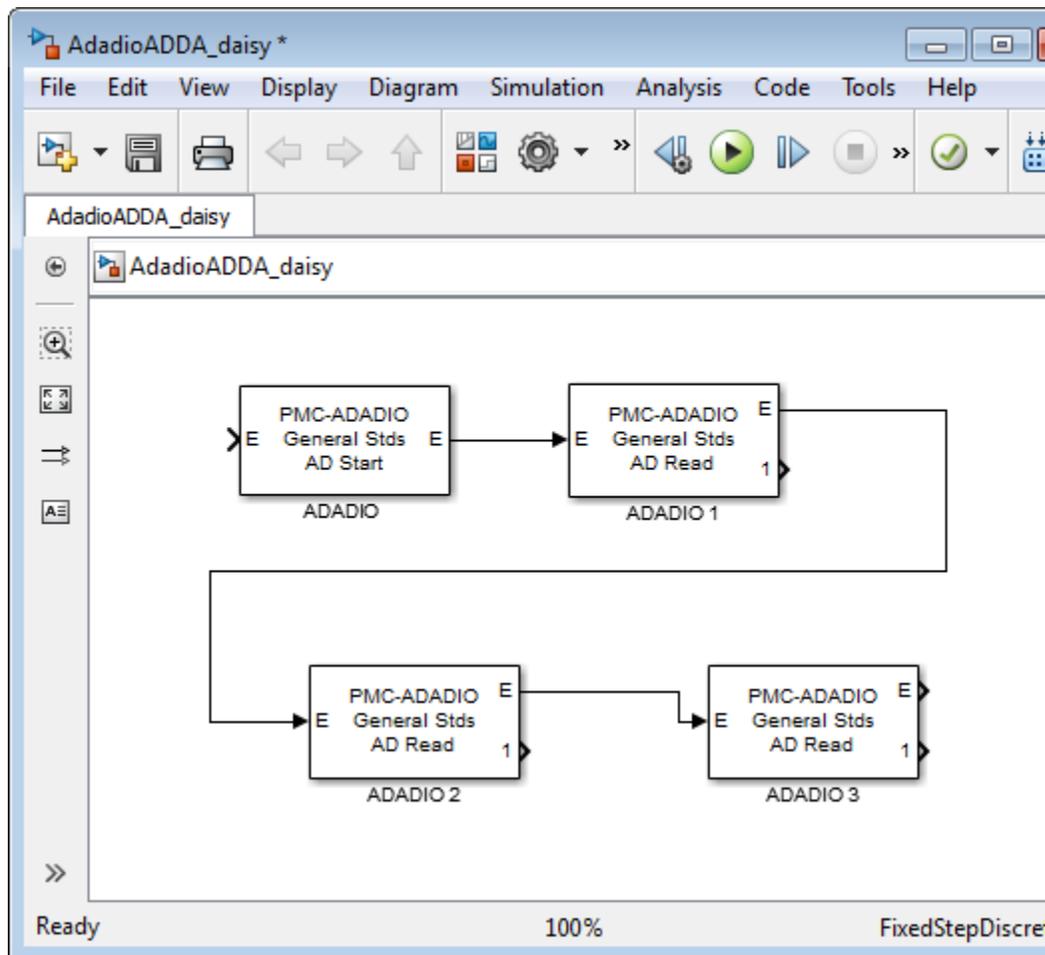
## Using Multiple PMC-ADADIO Boards

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.

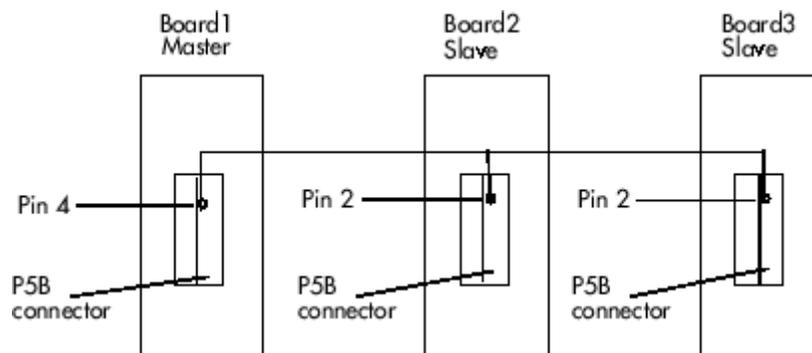
In such a configuration, observe the following guidelines:

- 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.

---

## Overview of Audio Applications

The xPC Target product supports audio applications with the General Standards PMC-24DSI12 and General Standards PMC66-16AO16 boards. The PMC-24DSI12 board provides frame audio input and the PMC66-16AO1 board supplies frame audio output. The following xPC Target blocks support these blocks:

### General Standards 24DSI12 Analog Input

Your audio application can include one or more General Standards PMC-24DSI12 boards. If you use multiple boards, designate one as the master. Then, daisy chain the other PMC-24DSI12 board as slaves to this board (see the General Standards PMC-24DSI12 documentation for details).

### General Standards 16AO16 Analog Output

For audio output, connect the clock output pins from the General Standards PMC-24DSI12 board to the clock input pins of the General Standards PMC66-16AO16 board.

---

### Tip

- Connect the boards with two wires twisted along their length.
  - If you are using more than one board, connect the clock output pins from the last board of the chain. See the General Standards PMC66-16AO1 documentation for details.
- 

The PMC-24DSI12 clock signal is a high-frequency signal, which requires you to preserve signal integrity when using this board. This signal is a differential signal. When the PMC66-16AO16 board receives the input clock signal from the PMC-24DSI12 board, the PMC66-16AO16 board divides that clock signal down to the sample rate. When using low voltage differential signaling (LVDS), you cannot slow down the clock for the PMC-24DSI12 board.

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

# General Standards PMC-ADADIO

**Purpose** Support for PMC-ADADIO multifunction ADC/DAC board

**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.

The xPC Target block library 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.

<b>Board Characteristics</b>	Board name	PMC-ADADIO
	Manufacturer	General Standards
	Bus type	PCI

# General Standards PMC-ADADIO

---

Access method	Memory mapped or I/O mapped
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 computer, enter  -1  to automatically locate the board.  If two or more boards of this type are in the target computer, enter the bus number and the PCI slot number 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></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. The xPC Target software 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 target computer. The output ports show a square wave during

# General Standards PMC-ADADIO Analog Input (A/D) Read

---

the cycle. Unless you subject this board to harsh conditions, you should only need to run autocalibration occasionally—for example, once a month.

---

**Note** The calibration cycle takes several seconds to run. 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 computer, enter

-1

to automatically locate the board.

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

# General Standards PMC-ADADIO Analog Input (A/D) Read

---

getxpci

# 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. The xPC Target software 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 target computer. The output ports show a square wave during the cycle. Unless you subject this board to harsh conditions, you should only need to run autocalibration occasionally— for example, once a month.

# General Standards PMC-ADADIO Analog Output Write Block

---

---

**Note** The calibration cycle takes several seconds to run. Refer to the PMC-ADADIO documentation for further information on autocalibration.

---

## **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 computer, enter

-1

to automatically locate the board.

If two or more boards of this type are in the target computer, enter the bus number and the PCI slot number of 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 computer, 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 computer, enter the bus number and the PCI slot number of 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.

## Scaling Input to Output

### Hardware Input

TTL

### Block Output Data Type

Double

### Scaling

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 computer, enter

-1

to automatically locate the board.

If two or more boards of this type are in the target computer, enter the bus number and the PCI slot number of 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

# General Standards PMC-ADADIO 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. 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 computer, enter

-1

to automatically locate the board.

If two or more boards of this type are in the target computer, enter the bus number and the PCI slot number of 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-16AO-12

---

**Purpose** Support for the GS PMC-16AO-12 high-speed analog board.

**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.

The xPC Target block library 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  $\pm 2.5$ ,  $\pm 5.0$ , or  $\pm 10.0$ .

<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 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.

# General Standards PMC-16AO-12 Analog 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. 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 target computer. 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 run autocalibration occasionally—for example, once a month.

# General Standards PMC-16AO-12 Analog Output

---

---

**Note** The calibration cycle takes several seconds to run. 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 computer, enter

-1

to automatically locate the board.

If two or more boards of this type are in the target computer, enter the bus number and the PCI slot number of 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 PMC66-16AO16

---

**Purpose** Support for GS PMC66–16AO16 high-speed analog output board

**Board** General Standards PMC66-16AO16

**General Description** The PMC66-16AO16 is a high speed analog output board. You can purchase the board with either:

- Eight, twelve, or sixteen output channels
- None, 10 kHz, or 100 kHz analog filters on the output ports

---

**Note** The analog output of this board contains a relatively high frequency component. Because of this, purchase the PMC66-16AO16 board with one of the filter options. If your board does not have a filter, its output will have very fast rise and fall times.

---

- Single-ended or differential output

The board simultaneously samples up to eight input channels from a single acquisition start. The board PMC form factor allows it to be attached to different adapters for use on PCI-express, PCI-X, cPCI-x, PCI, cPCI, or PC104-plus. You can have the board attached to your selected adapter.

The xPC Target block library supports this board with this driver block:

- General Standards 16AO16 Analog Output

You must use the General Standards 16AO16 Analog Output block with the General Standards 24DSI12 Analog Input for your xPC Target audio application.

# General Standards PMC66-16AO16

---

## **Board Characteristics**

Board name	General Standards PMC-16AO16
Manufacturer	General Standards
Bus type	PCI, PCI-express, PCI-X, cPCI-x, cPCI, or PC104-plus
Multiple block instance support	No
Multiple board support	Yes

# General Standards 16AO16 Analog Output

---

**Purpose** General Standards 16AO16 Analog Output

**Library** xPC Target Library for General Standards

<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 is a vector of channels. Specifies the output channels that the block works on. Each channel can appear only once in this vector. For example, to use the first, third, and fifth analog output channels, enter:

[1,3,5]

### **Output range**

From the list, select an output range code for all D/A channels. This driver does not allow the selection of a different range for individual 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
-2.5 to +2.5	+ -2.5
-1.5 to +1.5	+ -1.5

# General Standards 16AO16 Analog Output

---

## **Enable output ground sense hardware correction**

Select this check box to enable the REMOTE GROUND SENSE bit in the Board Control Register (BCR). See the board documentation for more information about this function.

## **Frame Size, Sample Rate, and Sampletime**

These three parameters are not independent but are related by:

$$\text{FrameSize} = \text{Sampletime} * \text{SampleRate}.$$

After you specify two of the parameters, the equation determines the third parameter. Enter -1 to have the block calculate that value from the other values.

For example, if you set **Frame Size** to 64 and **Sample Rate** to 44100, specify **Sampletime** as -1. It is computed internally to 0.001451247165533 seconds. This example model executes every 0.001451247165533 seconds, which is every 64 samples at 44.1 kHz.

## **Frame size (samples)**

Enter the number of samples in one frame for each channel listed in **Channel vector**. An interrupt occurs on the board each time the board acquires this number of samples.

## **Sample rate**

Enter the frequency at which the samples are taken, in Hz. This frequency must be in the range 2000 to 200000 Hz (2 kHz to 200 kHz).

## **Sample time (Frame completion time)**

Enter the time between frame completions.

## **Autocal at download**

Select this check box to execute the autocalibration cycle as part of the initial setup when the model is downloaded to the target computer. You might autocalibrate, for example, if the clock frequency changes when you download the model to the target

# General Standards 16AO16 Analog Output

---

computer. You can consider clearing this check box if you do not need to worry about exact voltage levels.

---

## Note

- The autocalibration cycle requires approximately 5 seconds to complete. Refer to the General Standards documentation for further information on autocalibration.
  - Autocalibration on the General Standards 16AO16 board might create undesirable output during model download.
- 

## External sync signal type

From the list, select one of the following to select a signal type for the clock output signal that can be connected to other input boards or to the PMC66-16AO16 output board.

### LVDS

Low voltage differential signaling. Select this value for greater reliability, especially when using the General Standards twisted-pair ribbon cables.

### TTL

See the PMC-16AO16 board documentation for guidelines on using this signal type.

## PCI Slot (-1:autosearch)

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

-1

to automatically locate the board.

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

# General Standards 16AO16 Analog Output

---

getxpci

# General Standards PMC-24DSI12

---

**Purpose** Support for GS PMC-24DSI12 high-speed analog input board.

**Board** General Standards PMC-24DSI12

**General Description** The PMC-24DSI12 is a high speed analog input board. You can purchase the board with either four, eight, or twelve input channels. The board PMC form factor allows it to be attached to different adapters for use on PCI-express, PCI-X, cPCI-x, PCI, cPCI, or PC104-plus. You can order the board attached to your selected adapter.

- This board operates in frame mode. To run the model with this board, you must perform additional setup procedures to route the buffer completion interrupt, as described in the following bullets.
- If your application requires more channels, you need to:
  - 1 Designate one board as the master board.
  - 2 Add slave boards PMC-24DSI12 to the target computer.
  - 3 Connect the output pins B29 and B30 of the master board to the clock input pins A29 and A30 of each slave board. Use twisted-pair wiring according to the LVDS directions.

The master board generates the interrupts from which to run the model.

- For frame audio output, route the clock output to a General Standards PMC66-16AO16 board. To do this
  - 1 Connect the output pins B29 and B30 from the last PMC-24DSI12 board to the input pins B33 and B34 pins of the PMC66-16AO16 board.
  - 2 Set the **Frame Size**, **Sample Rate**, and **Sampletime** parameters to the same value for each PMC-24DSI12 and PMC66-16AO16 block. For example, if **Sample Rate** is 41000, set the **Sample Rate** parameter to 44100 for each block.
- The PMC-24DSI12 uses DMA completion interrupts. To use this interrupt to run your model:

- 1 From the MATLAB Command Window, type `getxpcpci all`.

This function returns a list of all installed PCI devices on your target computer.

- 2 Find the string 24DSI12 and note the IRQ for that board.
- 3 From the MATLAB Command Window, type the name of your Simulink model.

The Simulink model appears.

- 4 From the model, select **Simulation > Model Configuration Parameter**.

The Configuration Parameter dialog box is displayed for the model.

- 5 Select the **Code Generation** node.
- 6 In the **xPC Target** options node, from the **Real-time interrupt source** list, select the same value as the IRQ value that `xpctarget.xpc.getxpcpci` returns (see step 2).
- 7 In the **xPC Target** options node, from the **I/O board generating the interrupt** list, select the value **General Standards 24DSI12**. This specifies that the General Standards 24DSI12 board generates the interrupt.
- 8 In the **PCI slot (-1: autosearch) or ISA base address** field, set the PCI bus and slot for your interrupting board. If you have only one PMC-24DSI12 board, you can enter -1 to enable the driver to find the only board.
- 9 Click **OK** and save the model.

The xPC Target block library supports this board with this driver block:

- General Standards 16AO16 Analog Output

# General Standards PMC-24DSI12

---

For frame audio output, use the General Standards PMC-24DSI12 Analog Input block with the General Standards 16AO16 Analog Output block for your xPC Target audio application.

## Board Characteristics

Board name	General Standards PMC-24DSI12
Manufacturer	General Standards
Bus type	PCI, PCI-express, PCI-X, cPCI-x, cPCI, or PC104-plus
Multiple block instance support	One block per board
Multiple board support	Yes. Only one can be the timing initiator; the others must be targets.

# General Standards 24DSI12 Analog Input

**Purpose** General Standards 24DSI12 Analog Input

**Library** xPC Target Library for General Standards

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

## Block Parameters

### Channel vector

This is a vector of channels. Specifies the output channels that the block works on. Each channel can appear only once in this vector. For example, to use the first, third, and fifth analog input channels, enter:

[1,3,5]

### Input range

From the list, select an input range code for all A/D channels. This driver does not allow the selection of a different range for individual 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

### Frame Size, Sample Rate, and Sampletime

These three parameters are not independent but are related by:

$$FrameSize = Sampletime \times SampleRate.$$

# General Standards 24DS112 Analog Input

---

After you specify two of the parameters, the equation determines the third parameter. Enter -1 to have the block calculate that value from the other values.

For example, if you set **Frame Size** to 64 and **Sample Rate** to 44100, specify **Sampletime** as -1. It is computed internally to 0.001451247165533 seconds. This example model executes every 0.001451247165533 seconds, which is every 64 samples at 44 kHz.

## Frame size (samples)

Enter the number of samples in one frame for each channel listed in **Channel vector**. An interrupt occurs on the board each time the board acquires this number of samples.

## Sample rate

Enter the frequency at which the samples are taken, in Hz. This frequency must be in the range 2000 to 200000 Hz (2 kHz to 200 kHz).

## Sample time (Frame completion time)

Enter the time between frame completions.

## Output format

From the list, select one of the following. The output value is always of type double.

+ - Input Range (+-10 + -5 or +-2.5)

Specifies that the value is scaled to the input voltage range.

signed 32 bit integer [-2<sup>23</sup> 2<sup>23</sup>-1]

Specifies that the value is a direct 24-bit value converted to a floating integer.

## Filter

From the list, select one of the following analog input filters:

- Low Frequency
- High Frequency

# General Standards 24DSI12 Analog Input

---

This parameter takes effect only if the **Sample rate** parameter exceeds 100 kHz. See the board documentation for more information.

## Triggering mode

From the list, select one of the following:

Hardware clock/Initiator

Select this value if the board is the timing master.

Target

Select this value if the board is connected to another board that is the timing master.

## Autocal at download

Select this check box to execute the autocalibration cycle as part of the initial setup when the model is downloaded to the target computer. You might autocalibrate, for example, after setting the acquisition frequency. You can consider clearing this check box if you do not need to worry about exact voltage levels.

---

**Note** The autocalibration cycle requires approximately 5 seconds to complete. Refer to the General Standards documentation for further information on autocalibration.

---

## External sync signal type

From the list, select one of the following to select a signal type for the clock output signal that can be connected to other input boards or to the PMC66-16AO16 output board.

LVDS

Low voltage differential signaling. Select this value for greater reliability, especially when using the General Standards twisted-pair ribbon cables.

# General Standards 24DSI12 Analog Input

---

TTL

See the board documentation for guidelines on using this signal type.

## **PCI Slot (-1:autosearch)**

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

-1

to automatically locate the board.

If two or more boards of this type are in the target computer, enter the bus number and the PCI slot number of 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 topic describes the HUMUSOFT I/O board supported by the xPC Target product (<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.

Humusoft AD622

I/O board with 8 single analog input (A/D) channels (14-bit), 8 analog output (D/A) channels (14-bit), 8 digital inputs, and 8 digital outputs.

Humusoft MF624

The MF 624 is an I/O board with 8 single analog input (A/D) channels (14-bit), 8 analog output (D/A) channels (14-bit), 8 digital inputs, 8 digital outputs, 5 counters/timers, and 4 quadrature encoder inputs with single-ended or differential input.

## **Boards and Blocks**

**Purpose** Humusoft AD 512 board

**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.

The xPC Target block library 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	I/O mapped
Multiple block instance support	Digital Input: Yes, Others: No
Multiple board support	Yes

# Humusoft AD 512 Analog Input (A/D)

---

**Purpose** HUMUSOFT 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.

Input Range (V)	Range Code
-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**

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 jumper settings on the board. For example, if the base address is 300 (hexadecimal), enter

0x300

# Humusoft AD 512 Analog Output (D/A)

---

**Purpose** HUMUSOFT 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.

<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

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** HUMUSOFT AD 512 Digital Input block

**Library** xPC Target Library for Humusoft

## Scaling Input to Output

Hardware Input	Block Output Data Type	Scaling
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** HUMUSOFT AD 512 Digital Output block

**Library** xPC Target Library for Humusoft

## Scaling Input to Output

Hardware Output	Block Input Data Type	Scaling
TTL	Double	< 0.5 = TTL low ≥ 0.5 = TTL high

### Channel vector

Enter 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

# Humusoft AD622

---

**Purpose** Humusoft AD622 board

**Board** HUMUSOFT AD622

**General Description** The AD622 is an I/O board with 8 single analog input (A/D) channels (14-bit), 8 analog output (D/A) channels (14-bit), 8 digital inputs, and 8 digital outputs.

The xPC Target block library supports this board with these driver blocks:

- Humusoft AD622 Analog Input (A/D)
- Humusoft AD622 Analog Output (D/A)
- Humusoft AD622 Digital Input
- Humusoft AD622 Digital Output

## **Board Characteristics**

Board name	AD622
Manufacturer	Humusoft
Bus type	PCI
Access method	Memory mapped
Multiple block instance support	Digital Input: Yes, Others: No
Multiple board support	Yes

# Humusoft AD622 Analog Input (A/D)

---

**Purpose** HUMUSOFT AD622 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.

### **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
```

# Humusoft AD622 Analog Output (D/A)

---

**Purpose** HUMUSOFT AD622 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**

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 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.

### **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** HUMUSOFT AD622 Digital Input block

**Library** xPC Target Library for Humusoft

## Scaling Input to Output

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

### Channel Vector

Enter 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.

### PCI Slot (-1:autosearch)

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

```
-1
```

to automatically locate the board.

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of 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 AD622 Digital Output

---

**Purpose** HUMUSOFT AD622 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 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.

## **PCI Slot (-1:autosearch)**

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

-1

to automatically locate the board.

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of 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** Humusoft MF624 board

**Board** HUMUSOFT MF624

## **General Description**

The MF624 is an I/O board with 8 single analog input (A/D) channels (14-bit), 8 analog output (D/A) channels (14-bit), 8 digital inputs, and 8 digital outputs. It also has 5 counters/timers and 4 quadrature encoder inputs with single-ended or differential input.

The xPC Target block library supports this board with these driver blocks:

- Humusoft MF624 Analog Input (A/D)
- Humusoft MF624 Analog Output (D/A)
- Humusoft MF624 Digital Input
- Humusoft MF624 Digital Output
- Humusoft MF624 Counter Input
- Humusoft MF624 Encoder Input
- Humusoft MF624 PWM Output

## **Board Characteristics**

Board name	MF624
Manufacturer	Humusoft
Bus type	PCI
Access method	Memory mapped
Multiple block instance support	Digital Input: Yes, Others: No
Multiple board support	Yes

# Humusoft MF624 Analog Input (A/D)

---

**Purpose** HUMUSOFT MF624 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.

### **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
```

# Humusoft MF624 Analog Output (D/A)

**Purpose** HUMUSOFT MF624 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**

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 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.

### **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
```

# Humusoft MF624 Digital Input

---

**Purpose** HUMUSOFT MF624 Digital Input block

**Library** xPC Target Library for Humusoft

## Scaling Input to Output

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

### Channel Vector

Enter 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.

### PCI Slot (-1:autosearch)

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

```
-1
```

to automatically locate the board.

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of 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** HUMUSOFT MF624 Digital Output block

**Library** xPC Target Library for Humusoft

## Scaling Input to Output

Hardware Output	Block Input Data Type	Scaling
TTL	Double	< 0.5 = TTL low ≥ 0.5 = TTL high

### Channel Vector

Enter 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.

### PCI Slot (-1:autosearch)

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

```
-1
```

to automatically locate the board.

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of 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 MF624 Counter Input

---

**Purpose** HUMUSOFT MF624 Counter Input 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 numbers between 1 and 4. This driver allows the selection of individual counter line numbers in any order. The number of elements defines the number of counter lines used.

For example, to use the first, second, and fourth counter lines, enter

[1,2,4]

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

## Reset after read

From the list, select:

- No — Do not reset counter after reading. Counting continues without stop.
- Yes — Reset counter after reading. Read number of pulses per sample period.

## Clock active edge

From the list, select:

- Rising — Increment the counter on the rising edge of the signal.
- Falling — Increment the counter on the falling edge of the signal.

- **Either** — Increment the counter on both edges of the signal. For example, with periodic signals, count twice the number of pulses (rising or falling).

### **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
```

# Humusoft MF624 Encoder Input

---

**Purpose** HUMUSOFT MF624 Encoder Input 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**

From the list, select 1, 2, 3, or 4. This driver allows the selection of the encoder channel numbers in any order. The number of elements defines the number of encoder input channels used.

For example, to use the first, second, and fourth PWM channel lines, enter

[1,2,4]

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

## **Input Filter Vector (0=Off, 1=On)**

Enter a vector of 0s and 1s, one for each encoder channel in **Channel Vector**. The length of this vector must be the same as the length of **Channel Vector**. Each **Input Filter Vector** element corresponds to its equivalent in **Channel Vector**.

Enter 1 to filter, or 0 to not filter the measured signal. Use this parameter to eliminate short spurious pluses. The input noise filters are disabled by default.

- 0 — Disables the input noise filter.
- 1 — Enables the input noise filter.

## **Index Function Vector (see help)**

Specify the function of the I input (index). Enter a vector of values, one for each PWM channel in **Channel Vector**. The

# Humusoft MF624 Encoder Input

length of this vector must be the same as the length of **Channel Vector**. Each **Input Function Vector** element corresponds to its equivalent in **Channel Vector**. Enter one of the following values for each channel:

Value	Operation
0	No function (disabled)
1	Reset counter when low
2	Reset counter when high
3	Reset counter on rising edge
4	Reset counter on falling edge
5	Reset counter on either edge
6	Enable counting when high
7	Enable counting when low

## 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
```

# Humusoft MF624 PWM Output

---

**Purpose** HUMUSOFT MF624 PWM 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 numbers between 1 and 8. This driver allows the selection of individual PWM output line numbers in any order. The number of elements defines the number of PWM 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.

## PCI Slot (-1:autosearch)

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

-1

to automatically locate the board.

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of 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

---

This topic describes I/O boards supported by the xPC Target product (<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 34-2	Description of block parameters for Keithley® driver blocks.

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

**Purpose** Keithley DAS-1800HR board

**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.

The xPC Target block library supports this board with these driver blocks:

- Keithley DAS-1800HR Analog Input (A/D)
- Keithley DAS-1800HR Digital Input
- Keithley DAS-1800HR Digital Output

<b>Board Characteristics</b>	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

**Purpose** Keithley KCPI-1801HC board

**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.

The xPC Target block library 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

The xPC Target software does not support the counter/timers on this board.

<b>Board Characteristics</b>	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
```

**Purpose** Keithley KPCI-1802HC board

**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.

The xPC Target block library 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

The xPC Target software does not support the counter/timers on this board.

<b>Board Characteristics</b>	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.

## 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-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

<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.

# 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
```

# MathWorks

---

This topic describes the xPC TargetBox<sup>®</sup> boards and blocks supported by xPC Target. xPC TargetBox is an industrial target computer that is optimized for executing real-time code generated with xPC Target, Simulink Coder, and a C/C++ compiler. MathWorks supplies and supports the blocks and documentation for users who already own this system.

---

**Note** xPC TargetBox is no longer available from MathWorks<sup>®</sup>. Instead, see its replacement, Classic Real-Time Target Machine or consider xPC Target Turnkey for other target computer recommendations.

---

“xPC TargetBox I/O Options” on page 35-2

Correlation between xPC TargetBox I/O options and supported xPC Target boards

“Blocks — Alphabetical List” on page 35-3

Description of block parameters for MathWorks blocks

## xPC TargetBox I/O Options

The following table lists the supported xPC Target boards that correspond to xPC TargetBox I/O options.

<b>I/O Option Name</b>	<b>xPC Target Equivalent</b>
xPC TargetBox IO 301	Diamond Diamond-MM-32-AT
xPC TargetBox IO 302	Diamond Ruby-MM-1612
xPC TargetBox IO 303	Diamond Ruby-MM-416
xPC TargetBox IO 304	Diamond Onyx-MM
xPC TargetBox IO 305	Diamond Quartz-MM-10
xPC TargetBox IO 306	Real Time Devices DM6814
xPC TargetBox IO 308	Softing CAN-AC2-104 with SJA1000 Setup
xPC TargetBox IO 309	MPL PATI

## **Blocks – Alphabetical List**

# LED

---

## Purpose

Panel LEDs

## Description

Some units have two LEDs that you can control. These LEDs are labeled USER 1 and USER 2.

From the library, drag an LED block into your Simulink model. You can have only one block of this type in any model.

## Library

xPC Target Library for LED

## Block Parameters

### LED vector

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

[1 2]

### Reset vector

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

### Initial value vector

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

**Sample time**

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

# Watchdog Timer

---

<b>Purpose</b>	Watchdog timer
<b>Description</b>	On some units, you can program a watchdog to perform a system reboot when a programmable timeout occurs. The timeout interval can range from 1 to 255 seconds with one-second resolution, or from 1 to 255 minutes with one-minute resolution. When the timeout expires, an IRQ15 interrupt is asserted. With the <b>Reboot upon expiration of watchdog</b> parameter, you can specify that the system reboot at this time. At most, one block of this type can be included in any model.
<b>Library</b>	xPC Target Library for Watchdog/MathWorks
<b>Block Parameters</b>	<p><b>Timeout units</b> From the list, select either seconds or minutes. These are the units for the time.</p> <p><b>Timeout interval</b> Enter a number between 1 and 255.</p> <p><b>Show reset port (R)</b> Select this check box to display an input port labeled R. A signal connected to this port resets the watchdog whenever its value exceeds 0.5.</p> <p><b>Reset upon keyboard activity</b> Select this check box to reset the watchdog whenever there is keyboard activity.</p> <p><b>Reboot upon expiration of watchdog</b> Select this check box to reboot the unit when the watchdog timer expires. When the timeout expires, an IRQ15 interrupt is always asserted, regardless of the state of this check box. If you choose not to reboot the system when the watchdog timer expires, you might want to write your own procedure to handle the IRQ15 assertion.</p> <p><b>Sample time</b> Enter a base sample time or a multiple of the base sample time.</p>

# Measurement Computing

---

This topic describes Measurement Computing (formerly Computer Boards) I/O boards supported by the xPC Target product (<http://www.measurementcomputing.com>)

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-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-PDISO8	I/O board with 8 inputs and 8 relay outputs.
Measurement Computing PCI-PDISO16	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

“Boards and Blocks —  
Alphabetical List” on page  
36-6

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.

Description of block parameters for Measurement Computing driver blocks.

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

# Measurement Computing CIO-DAC08 (/12)

---

**Purpose** Support for the MC CIO-DAC08 (/12) board

**Board** CIO-DAC08 (/12)

**General Description** The CIO-DAC08 (/12) is an I/O board with eight analog output (D/A) channels (12-bit).

The xPC Target block library 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.

<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-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

---

**Purpose** Support for the MC CIO-DAC08/16 I/O board

**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).

The xPC Target block library supports this board with this driver block:

- Measurement Computing CIO-DAC08/16 Analog Output (D/A)

## **Board Characteristics**

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.

<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-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)

---

## **Purpose**

Support for the MC CIO-DAC16 (/12) I/O board

## **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).

The xPC Target block library supports this board with one driver block:

- Measurement Computing CIO-DAC16 Analog Output (D/A)

## **Board Characteristics**

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
0 to +5	5
0 to +2.5	2.5

# Measurement Computing CIO-DAC16 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

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-DAC16/16

---

**Purpose** Support for the MC CIO-DAC16/16 I/O board

**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).

The xPC Target block library supports this board with this driver block:

- Measurement Computing CIO-DAC16/16 Analog Output (D/A)

## **Board Characteristics**

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
0 to +5	5
0 to +2.5	2.5

# Measurement Computing CIO-DAC16/16 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

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

---

**Purpose** Support for the MC CIO-DAS16/330 I/O board

**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.

The xPC Target block library supports this board with one driver block:

- Measurement Computing CIO-DAS16/330 Analog Input (A/D)

---

**Note** The xPC Target product 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)

---

**Purpose** Support for the MC CIO-DAS16/JR (/12) I/O board

**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.

The xPC Target block library 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** The xPC Target software 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  $\pm 10V$  (-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 (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 output 1  
the Signal of EXP-channel 0 is output at block output 2  
the Signal of EXP-channel 12 is output at block output 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 you to use the A/D-channels of a CIO-DAS16Jr either for EXP-channels or 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

---

**Purpose** Support for the MC CIO-DAS16JR/16 I/O board

**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.

The xPC Target block library supports this board with this driver block:

- Measurement Computing CIO-DAS16JR/16 Analog Input (A/D)

---

**Note** The xPC Target software does not support the digital I/O or the counters on this board.

---

<b>Board Characteristics</b>	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

---

**Purpose** Support for the MC CIO-DAS1601/12 I/O board

**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).

The xPC Target block library 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** The xPC Target software 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 an 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 an 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 required 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 an 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 an 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 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 required 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

---

**Purpose** Support for the MC CIO-DAS1602/12 I/O board

**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).

The xPC Target block library 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** The xPC Target software 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 an 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 an 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 required 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 an 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 an 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 required 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

## Purpose

Support for the MC CIO-DAS1602/16 I/O board

## 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).

The xPC Target block library 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** The xPC Target software 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 an 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 an 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 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 required 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 an 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 an 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 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 required 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)

---

**Purpose** Support for the MC CIO-DDA06 (/12) I/O board

**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.

The xPC Target block library 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

## Board Characteristics

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 an 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 an 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 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 required 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 an 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 an 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 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 required 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

---

**Purpose** Support for the MC CIO-DDA06/16 I/O board

**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.

The xPC Target block library 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

## Board Characteristics

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 an 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 an 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 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 required 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 an 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 an 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 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 required 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

---

**Purpose** Support for the MC CIO-DIO24 I/O board

**Board** Measurement Computing CIO-DIO24

**General Description** The CIO-DIO24 is an I/O board with 24 digital I/O lines. The xPC Target block library 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

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 an 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 required 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 an 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 required 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

---

<b>Purpose</b>	CIO-DIO24 Signal Conditioning block
<b>Library</b>	xPC Target Library for Measurement Computing
<b>Block Parameters</b>	<b>Genix initialization file (path\file)</b> Provide the filename of the Genix initialization file.
	<b>Base address</b> 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

---

**Purpose** Support for the MC CIO-DIO24H I/O board

**Board** Measurement Computing CIO-DIO24H

**General Description** The CIO-DIO24H is an I/O board with 24 digital I/O lines. The xPC Target block library 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 an 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 required 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	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 an 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

# Measurement Computing CIO-DIO24H Digital Output

---

depending on which driver block is chosen. In each case, one block is required 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

**Purpose** Support for the MC CIO-DIO48 I/O board

**Board** Measurement Computing CIO-DIO48

**General Description** The CIO-DIO48 is an I/O board with 48 digital I/O lines. The xPC Target block library 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 required 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.

## 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 required 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

---

**Purpose** Support for the MC CIO-DIO48H I/O board

**Board** Measurement Computing CIO-DIO48H

**General Description** The CIO-DIO48H is an I/O board with 48 digital I/O lines. The xPC Target block library 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 required 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 required 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

**Purpose** Support for the MC CIO-DIO96 I/O board

**Board** Measurement Computing CIO-DIO96

**General Description** The CIO-DIO96 is an I/O board with 96 digital I/O lines. The xPC Target block library 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 required 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.

## 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 required 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

---

**Purpose** Support for the MC CIO-DIO192 I/O board

**Board** Measurement Computing CIO-DIO192

**General Description** The CIO - DIO192 is an I/O board with 192 digital I/O lines. The xPC Target block library 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 required 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 required 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

# Measurement Computing CIO-DO24DD

---

**Purpose** Support for the MC CIO-DO24DD I/O board

**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 an 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 required 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

---

**Purpose** Support for the MC CIO-PDISO16 I/O board

**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.

The xPC Target block library supports this board with these driver blocks:

- Measurement Computing CIO-PDISO16 Digital Input
- Measurement Computing CIO-PDISO16 Digital Output

---

**Note** The xPC Target software 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 the xPC Target software.

# 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.

<b>Scaling Input to Output</b>	<b>Hardware Output</b>	<b>Block Input Data Type</b>	<b>Scaling</b>
	relay	Double	< 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 the xPC Target software.

# Measurement Computing CIO-QUAD02

---

**Purpose** Support for the MC CIO-QUAD02 counting board

**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.

The xPC Target block library 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.

<b>Board Characteristics</b>	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> 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>
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</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,</p>

# Measurement Computing CIO-QUAD02 Incremental Encoder

Counting Mode	Index input resets counter Selected	Index input resets counter Deselected
	<p><b>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>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>
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>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.</p>

# 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> 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

## Purpose

Support for the MC CIO-QUAD04 counting board

## 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.

The xPC Target block library 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 a value that differs from the actual 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 CIO-QUAD04

---

## **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> 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>
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</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>

# Measurement Computing CIO-QUAD04 Incremental Encoder

Counting Mode	Index input resets counter Selected	Index input resets counter Deselected
	decrements one revolution then resets to <b>Count Limit or Reset Value</b> .	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.
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.
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 <b>Count Limit</b>

# Measurement Computing CIO-QUAD04 Incremental Encoder

Counting Mode	Index input resets counter Selected	Index input resets counter Deselected
		<p><b>or Reset Value</b> 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

# Measurement Computing CIO-QUAD04 Incremental Encoder

---

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.

## **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)

---

**Purpose** Support for the MC PC104–DAC06 (/12) I/O board

**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).

The xPC Target block library supports this board with this driver block:

- Measurement Computing PC104-DAC06 (/12) Analog Output (D/A)

## Board Characteristics

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.

<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 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

---

**Purpose** Support for the MC PC104–DAS16JR/12 I/O board

**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.

The xPC Target block library 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 +-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 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 the xPC Target software.

# Measurement Computing PC104-DAS16JR/16

---

**Purpose** Support for the MC PC104–DAS16JR/16 I/O board

**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.

The xPC Target block library 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  $\pm 10V$  (-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 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

---

**Purpose** Support for the MC PC104–DIO48 I/O board

**Board** Measurement Computing PC104-DIO48

**General Description** The PC104-DIO48 is an I/O board with 48 digital I/O lines. The xPC Target block library 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 required 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.

## 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 required 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-DAC6703

---

**Purpose** Support for the MC PCI-DAC6703 I/O board

**Board** Measurement Computing PCI-DAC6703

**General Description** The PCI-DAC6703 is an I/O board with 16 analog output (D/A) channels (16-bit) and 8 digital input and output lines.

The xPC Target block library supports this board with these driver blocks:

- Measurement Computing PCI-DAC6703 Analog Output (D/A)
- Measurement Computing PCI-DAC6703 Digital Input
- Measurement Computing PCI-DAC6703 Digital Output

## Board Characteristics

Board name	PCI-DAS6703
Manufacturer	Measurement Computing
Bus type	PCI
Access method	I/O mapped
Multiple block instance support	D/A: one block per board Digital I/O: each bit in at most one block
Multiple board support	Yes

# Measurement Computing PCI-DAC6703 Analog Output (D/A)

---

**Purpose** PCI-DAC6703 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 16. This driver allows the selection of individual D/A in any order. The number of elements defines the number of analog output channels used. For example, to use all of the analog output channels, enter

[1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16]

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 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 PCI-DAC6703 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 PCI-DAC6703 Digital Input

**Purpose** PCI-DAC6703 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**

### **Channel vector**

Enter numbers between 1 and 8 to select the digital input lines. 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]
```

Each digital line can be in either a digital input or digital output block, but not both.

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.

If two or 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-DAC6703 Digital Input

---

this driver block. Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```

# Measurement Computing PCI-DAC6703 Digital Output

**Purpose** PCI-DAC6703 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

### 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 ]

Each digital line can be in either a digital input or digital output block, but not both.

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.

# Measurement Computing PCI-DAC6703 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.

## **PCI Slot (-1:autosearch)**

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

-1

to automatically locate the board.

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of 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** Support for the MC PCI-DAS1002 I/O board

**Board** Measurement Computing PCI-DAS1002

**General Description** The PCI-DAS1002 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), and 24 digital input and output lines.

The xPC Target block library supports this board with these driver blocks:

- Measurement Computing PCI-DAS1002 Analog Input (A/D)
- Measurement Computing PCI-DAS1002 Analog Output (D/A)
- Measurement Computing PCI-DAS1002 Digital Input
- Measurement Computing PCI-DAS1002 Digital Output

## Board Characteristics

Board name	PCI-DAS1002
Manufacturer	Measurement Computing
Bus type	PCI
Access method	I/O mapped
Multiple block instance support	A/D — one block per board D/A — each input port can be in one block Digital I/O — one block per 8-bit port
Multiple board support	Yes

# Measurement Computing PCI-DAS1002 Analog Input (A/D)

---

**Purpose** PCI-DAS1002 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**

Enter  $N$ , the number of channels to acquire. This board acquires data from channels 1 to  $N$ .

- Enter  $N$  as a value between 1 and 16 if **Input coupling** is Single-ended (16 channels).
- Enter  $N$  as a value between 1 and 8 if **Input coupling** is Differential (8 channels).

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 though the board manufacturer starts numbering the lines with 0.

### **Range vector**

From the list, choose either  $\pm 10V$  (-10 volts to +10 volts),  $\pm 5V$ ,  $\pm 2.5V$ ,  $\pm 1.25V$ ,  $0-10V$ ,  $0-5V$ ,  $0-2.5V$ , 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 following input modes:

- Single-ended (16 channels)
- Differential (8 channels)

### **Sample time**

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

# Measurement Computing PCI-DAS1002 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-DAS1002 Analog Output (D/A)

---

**Purpose** PCI-DAS1002 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 though 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 PCI-DAS1002 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-DAS1002 Digital Input

---

**Purpose** PCI-DAS1002 Digital Input block

**Library** xPC Target Library for Measurement Computing

**Note** The PCI-DAS1002 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 though 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 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 required for each port.

# Measurement Computing PCI-DAS1002 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-DAS1002 Digital Output

**Purpose** PCI-DAS1002 Digital Output block

**Library** xPC Target Library for Measurement Computing

**Note** The PCI-DAS1002 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 = 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 though 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 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 required 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

---

**Purpose** Support for the MC PCI-DAS1200 I/O board

**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.

The xPC Target block library 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**

Enter  $N$ , the number of channels to acquire. This board acquires data from channels 1 to  $N$ .

- Enter  $N$  as a value between 1 and 16 if **Input coupling** is Single-ended (16 channels).
- Enter  $N$  as a value between 1 and 8 if **Input coupling** is Differential (8 channels).

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 though the board manufacturer starts numbering the lines with 0.

### **Range**

From the list, choose either  $\pm 10V$  (-10 volts to +10 volts),  $\pm 5V$ ,  $\pm 2.5V$ ,  $\pm 1.25V$ ,  $0-10V$ ,  $0-5V$ ,  $0-2.5V$ , 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)

### **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 though 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 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]

## **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-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.

## 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 though 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 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 required 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.

## 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 though 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 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 Digital Output

---

depending on which driver block is chosen. In each case, one block is required 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

---

**Purpose** Support for the MC PCI-DAS1200/JR I/O board

**Board** Measurement Computing PCI-DAS1200/JR

**General 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.

The xPC Target block library 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

<b>Board Characteristics</b>	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.

<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 an 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 required 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-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.

<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 an 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 required 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

---

**Purpose** Support for the MC PCI-DAS1602/12 I/O board

**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).

The xPC Target block library 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** The xPC Target software 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.

### **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.

### **PCI Slot (-1:autosearch)**

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

# Measurement Computing PCI-DAS1602/12 Analog Input (A/D)

---

- 1

to automatically locate the board.

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of 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 an 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 an 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 required 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 an 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 an 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 or 8 digital lines that can be configured as inputs or outputs

# Measurement Computing PCI-DAS1602/12 Digital Output

---

depending on which driver block is chosen. In each case, one block is required 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

---

**Purpose** Support for the MC PCI-DAS1602/16 I/O board

**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).

The xPC Target block library 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** The xPC Target software 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.

### **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.

### **PCI Slot (-1:autosearch)**

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

# Measurement Computing PCI-DAS1602/16 Analog Input (A/D)

---

- 1

to automatically locate the board.

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number 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-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]

## **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 1602/16 Digital Input

**Purpose** PCI-DAS 1602/16 Digital Input block

**Library** xPC Target Library for Measurement Computing

**Note** The DAS1601/16 has an 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 an 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 or 8 digital lines that can be configured as inputs or outputs

# Measurement Computing PCI-DAS 1602/16 Digital Input

---

depending on which driver block is chosen. In each case, one block is required 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 an 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 an 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 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 required 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

---

**Purpose** Support for the MC PCI-DDA02/12 I/O board

**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.

The xPC Target block library 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.

<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
0 to +5	5
0 to +2.5	2.5

# Measurement Computing PCI-DDA02/12 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.

# 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.

## 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 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 required 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 required 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

---

**Purpose** Support for the MC PCI-DDA02/16 I/O board

**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.

The xPC Target block library 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

## 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-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
0 to +5	5
0 to +2.5	2.5

# Measurement Computing PCI-DDA02/16 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 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.

## 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 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 required 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 = 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 required 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

---

**Purpose** Support for the MC PCI-DDA04/12 I/O board

**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.

The xPC Target block library 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
0 to +5	5
0 to +2.5	2.5

# Measurement Computing PCI-DDA04/12 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.

# 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

# Measurement Computing PCI-DDA04/12 Digital Input

---

depending on which driver block is chosen. In each case, one block is required 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-DDA04/12 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 required 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

---

**Purpose** Support for the MC PCI-DDA04/16 I/O board

**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.

The xPC Target block library 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
0 to +5	5
0 to +2.5	2.5

# Measurement Computing PCI-DDA04/16 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 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 required 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 required 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

---

**Purpose** Support for the MC PCI-DDA08/12 I/O board

**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.

The xPC Target block library 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

## Board Characteristics

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

<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
0 to +5	5
0 to +2.5	2.5

# Measurement Computing PCI-DDA08/12 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.

# 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 required 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 required 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

---

**Purpose** Support for the MC PCI-DDA08/16 I/O board

**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.

The xPC Target block library 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

<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
0 to +5	5
0 to +2.5	2.5

# Measurement Computing PCI-DDA08/16 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 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 required 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.

<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 required 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

---

**Purpose** Support for the MC PCI-DIO24 I/O board

**Board** Measurement Computing PCI-DIO24

**General Description** The PCI-DIO24 is an I/O board with 24 digital I/O lines.  
The xPC Target block library 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

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 an 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 required 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 an 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

# Measurement Computing PCI-DIO24 Digital Output

---

depending on which driver block is chosen. In each case, one block is required 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

---

**Purpose** Support for the MC PCI-DIO24H I/O board

**Board** Measurement Computing PCI-DIO24H

**General Description** The PCI-DIO24H is an I/O board with 24 digital I/O lines. The xPC Target block library 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 an 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

# Measurement Computing PCI-DIO24H Digital Input

---

depending on which driver block is chosen. In each case, one block is required 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-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 an 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

# Measurement Computing PCI-DIO24H Digital Output

---

depending on which driver block is chosen. In each case, one block is required 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
```

**Purpose** Support for the MC PCI-DIO48H I/O board

**Board** Measurement Computing PCI-DIO48H

**General Description** The PCI-DIO48H is an I/O board with 48 digital I/O lines. The xPC Target block library 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 required 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

depending on which driver block is chosen. In each case, one block is required 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

---

**Purpose** Support for the MC PCI-DIO96H I/O board

**Board** Measurement Computing PCI-DIO96H

**General Description** The PCI-DIO96H is an I/O board with 96 digital I/O lines. The xPC Target block library 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 required 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 required 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
```

**Purpose** Support for the MC PCI-DIO96 I/O board

**Board** Measurement Computing PCI-DIO96

**General Description** The PCI-DIO96 is an I/O board with 96 digital I/O lines. The xPC Target block library 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 = 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 **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

---

**Purpose** Support for the PCI-PDISO8 I/O board

**Board** Measurement Computing PCI-PDISO8

**General Description** The PCI-PDISO8 is an I/O board with eight inputs and eight relay outputs.

The xPC Target block library 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

## Scaling Input to Output

### Hardware Input

OPTO

### Block Output Data Type

Double

### Scaling

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
```

**Purpose** Support for the MC PCI-PDISO16 I/O board

**Board** Measurement Computing PCI-PDISO16

**General Description** The PCI-PDISO16 is an I/O board with 16 inputs and 16 relay outputs. The xPC Target block library supports this board with these driver blocks:

- Measurement Computing PCI-PDISO16 Digital Input
- Measurement Computing PCI-PDISO16 Digital Output

<b>Board Characteristics</b>	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 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.

## PCI Slot (-1:autosearch)

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

-1

to automatically locate the board.

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of 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

---

**Purpose** Support for the MC PCIM-DAS1602/16 I/O board

**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).

The xPC Target block library 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** The xPC Target software 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 an 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 an 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 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 required 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 an 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 an 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 required 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

---

**Purpose** Support for the MC PCIM-DDA06/16 I/O board

**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.

The xPC Target block library 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 the xPC Target software, 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 an 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.

## 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 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 required 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 an 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 required 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

---

**Purpose** Support for the MC PCI-DUAL-AC5 I/O board

**Board** Measurement Computing PCI-DUAL-AC5

**General Description** The PCI-DUAL-AC5 is an I/O board with 48 digital I/O lines. The xPC Target block library 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

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 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 required 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

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 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 required 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
```

## Purpose

Support for the MC PCI-QUAD04 counting board

## 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.

The xPC Target block library 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 a value that differs from the actual 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 Measurement Computing PCI-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 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> 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>
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</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>

# Measurement Computing PCI-QUAD04 Incremental Encoder

Counting Mode	Index input resets counter Selected	Index input resets counter Deselected
	decrements one revolution then resets to <b>Count Limit or Reset Value</b> .	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.
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.
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

# Measurement Computing PCI-QUAD04 Incremental Encoder

Counting Mode	Index input resets counter Selected	Index input resets counter Deselected
		to <b>Count Limit or Reset Value</b> 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).

## 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

# Measurement Computing PCI-QUAD04 Incremental Encoder

---

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.

## 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
```

# Measurement Computing PCI-DAS-TC

---

**Purpose** Support for the MC PCI-DAS-TC I/O board

**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.

The xPC Target block library supports this board with this driver block:

- Measurement Computing PCI-DAS-TC Thermocouple

## Board Characteristics

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

```
getxpcpci
```

---

**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 “Increase the Time for Downloads”.

---

# Measurement Computing PCI-DAS-TC Thermocouple

---

# MPL

---

This topic describes the MPL board supported by the xPC Target product (<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 – Alphabetical List**

**Purpose** Support for MPL PATI board.

**Board** MPL PATI

## General Description

The MPL PowerPC controlled Analog and Timing I/O Intelligence (PATI) has 32 32-bit TPU channels, which the xPC Target software supports for PWM measurement, PWM generation, time base selection, incremental encoder measurement, and digital I/O. It is based on the Freescale™ MPC555 processor. The xPC Target software supports only the TPU channels for this board. The xPC Target block library supports the MPL PATI board with these driver blocks:

The xPC Target block library 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
- MPL PATI EEPROM Write

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

# MPL PATI

---

- S3 — off
- S4 — on (enable reboot)
- S5 — off
- 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

**General Description** 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.

---

**Note** To execute a model using this block, you must first write MPL boot code on the EEPROM using the MPL PATI EEPROM Write block.

To flash the EEPROM with the boot code, add the EEPROM Write block to an empty model, then build and execute that model before executing the actual model. See MPL PATI EEPROM Write block for further information.

---

## 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.

## MPL PATI 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
```

**Purpose** MPL PATI Digital Output block

**Library** xPC Target Library for MPL

**General Description** 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.

---

**Note** To execute a model using this block, you must first write MPL boot code on the EEPROM using the MPL PATI EEPROM Write block.

To flash the EEPROM with the boot code, add the EEPROM Write block to an empty model, then build and execute that model before executing the actual model. See MPL PATI EEPROM Write block for further information.

---

## 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

# MPL PATI Digital Output

---

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
```

**Purpose**

MPL PATI Incremental Encoder block

**Library**

xPC Target Library for MPL

**General Description**

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.

# MPL PATI Incremental Encoder

---

---

**Note** To execute a model using this block, you must first write MPL boot code on the EEPROM using the MPL PATI EEPROM Write block.

To flash the EEPROM with the boot code, add the EEPROM Write block to an empty model, then build and execute that model before executing the actual model. See MPL PATI EEPROM Write block for further information.

---

## 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.

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

`getxpcpci`

# MPL PATI PWM generate

---

**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

## **General Description**

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**.

---

**Note** To execute a model using this block, you must first write MPL boot code on the EEPROM using the MPL PATI EEPROM Write block.

To flash the EEPROM with the boot code, add the EEPROM Write block to an empty model, then build and execute that model before executing the actual model. See MPL PATI EEPROM Write block for further information.

---

## 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.

## 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.

# MPL PATI PWM generate

---

Clear this check box to allow the current output cycle to finish before the update. Clearing this check box generates smoother frequency and phase transitions without transients.

The block does not interrupt long pulses. Note that your selection of this parameter value depends on your application.

## Arm input

Select this check box to indicate that the ARM input is present. If this check box is selected, the channel behaves as follows. Note that there might be a delay, on the order of the output period, between the time that the ARM input becomes 1 and the output port starts changing.

- If the input signal to this port is 1, the channel outputs the signal specified by the H and L ports.
- If the input signal to this port is 0, the output stops at the level specified by the **Disarm state** drop-down list.

If you do not select this check box, the block behaves as if the ARM input is 1. No A input port appears.

## Disarm state

From the drop-down list, select

- Disarm to LOW
- Disarm to HIGH

When the ARM input is 0, this parameter specifies the state of the output.

## State on model stop

From the drop-down list, select

- Leave running  
Continue to get output pulses at the last H and L port values.
- Stop to LOW

Immediately go to the LOW output state and maintain that level.

- Stop to HIGH

Immediately go to the HIGH output state and maintain that level.

## **Timebase**

From the list, select TCR1 or TCR2. This depends on the MPL PATI Timebase Setup 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

# MPL PATI PWM measure

---

**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

**General Description** 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 use the TPUs, you must 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.

---

**Note** To execute a model using this block, you must first write MPL boot code on the EEPROM using the MPL PATI EEPROM Write block.

To flash the EEPROM with the boot code, add the EEPROM Write block to an empty model, then build and execute that model before executing the actual model. See MPL PATI EEPROM Write block for further information.

---

**Block Parameters** Channel  
From the list, select a channel from 1 to 32. This board has two TPUs. Channels 1 through 16 are on the first TPU, channels 17

through 32 are on the second TPU. The channel you select works with the time base parameter.

## **Channel priority**

From the list, select Low, Medium, or High priority. This has a small effect on the order that individual timers are serviced. See the MPC555 TPU documentation for more information.

## **Mode**

From the list, select

- Total High Time — For measuring pulse widths
- Total Low Time — For measuring pulse widths
- Between Rising Edges — For period measurements
- Between Falling Edges — For period measurements

## **Number of periods to count**

Enter the number of periods to count. To measure every cycle separately, enter a value of 1 (default). To get higher precision by averaging, enter a higher value, then divide the block output by this value. For example, to gain a period with a precision of 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.

## MPL PATI PWM measure

---

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of 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

## General Description

The Freescale 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.

---

**Note** To execute a model using this block, you must first write MPL boot code on the EEPROM using the MPL PATI EEPROM Write block.

To flash the EEPROM with the boot code, add the EEPROM Write block to an empty model, then build and execute that model before executing the actual model. See MPL PATI EEPROM Write block for further information.

---

# MPL PATI Timebase Setup

---

## Block Parameters

### TCR1 divider (1–16)

From the list, choose the divider that sets the TCR1 time base for channels 1 to 16. The minimum divider of 2 gives the maximum time base frequency of 20 MHz. The maximum divider of 512 gives a time base frequency of 78.125 kHz.

### TCR2 divider (1–16)

From the list, choose one of the following:

- 8 (5 Mhz) or ext
- 16 (2.5 Mhz) or ext/2
- 32 (1.25 Mhz) or ext/4
- 64 (625 Khz) or ext/8

This divider sets the TCR2 time base for channels 1 to 16. You can specify that the TCR2 time base can be driven by either the 40 MHz CPU clock or by a signal on the external T2CLK input pin. Note that the maximum frequency of the external signal is approximately  $40 \text{ MHz}/9 = 4.44 \text{ MHz}$ .

If you want to use the external clock, select the **TCR2 external clock input (1–16)** check box.

### TCR2 external clock input (1–16)

Select this check box to use the signal on the T2CLK pin as the source for TCR2. Use this parameter with the **TCR2 divider (1–16)** parameter.

### TCR1 divider (17–32)

From the list, choose the divider that sets the TCR1 time base for 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
```

# MPL PATI EEPROM Write

---

**Purpose** MPL PATI EEPROM Write

**Library** xPC Target Library for MPL

**General Description** The MPL PATI EEPROM Write block writes MPL boot code on the EEPROM. Before executing any xPC Target model that contains an MPL PATI block, you must first write MPL boot code on the EEPROM using the MPL PATI EEPROM Write block.

To flash the EEPROM with the boot code:

- 1 Add the EEPROM Write block to an empty model. To access the library that contains this block, type

```
xpcmpllib
```

- 2 Build and execute this model. All the work is completed when the model is downloaded to the target machine.

- 3 Reboot the target PC.

- 4 Build and execute the actual model.

---

**Note** Always use this block by itself in an empty model to re-flash the EEPROM. Be sure to run this model only once to write the boot code on the EEPROM. You must reboot the target PC before executing the actual model. Do not include this block in any other model.

---

**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
```

## **MPL PATI EEPROM Write**

---

# National Instruments

---

This topic describes National Instruments I/O boards supported by the xPC Target product (<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
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-6221	I/O board with up to 16 single-ended or 8 differential 16-bit analog inputs, two 16-bit analog outputs, 24 digital I/O lines, and two 32-bit counters.

National Instruments PCI-6221/37	I/O board with up to 16 single-ended or 8 differential 16-bit analog inputs, two 16-bit analog outputs, 10 digital I/O lines, and two 32-bit counters.
National Instruments PCI-6225	I/O board with two 16-bit analog outputs, 24 digital I/O lines, and two 32-bit counters.
National Instruments PCI-6229	I/O board with 32 single-ended or 16 differential 16-bit analog inputs, four 16-bit analog outputs, 48 digital I/O lines, and two 32-bit counters.
National Instruments PCI-6251	I/O board with up to 16 single-ended or 8 differential 16-bit analog inputs, two 16-bit analog outputs, 24 digital I/O lines, and two 32-bit counters.
National Instruments PCI-6259	I/O board with up to 32 single-ended or 16 differential 16-bit analog inputs, four 16-bit analog outputs, 48 digital I/O lines, and 32-bit counters.
National Instruments PCI-6280	I/O board with up to 16 single-ended or 8 differential 18-bit analog inputs, 24 digital I/O lines, and two 32-bit 80 MHz counters.
National Instruments PCI-6281	I/O board with up to 16 single-ended or 8 differential 18-bit analog inputs, two 16-bit analog outputs, 24 digital I/O lines, and two 32-bit 80 MHz counters.
National Instruments PCI-6289	I/O board with up to 32 single-ended or 16 differential 18-bit analog inputs, four 16-bit analog outputs, 48 digital I/O lines, and two 32-bit 80 MHz counters.
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 four analog output (D/A) channels (12-bit) and eight digital input and output lines.
National Instruments PCI/PXI-6713	I/O board with eight analog output (D/A) channels (12-bit) and eight digital input and output lines.
National Instruments PCI/PXI-6731	I/O board with four analog output (D/A) channels (16-bit) and eight digital input and output lines.
National Instruments PCI/PXI-6733	I/O board with eight analog output (D/A) channels (16-bit) and eight 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 38-6	Description of block parameters for National Instruments driver blocks.

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

**Purpose** Support an I/O board with 6 D/A channels and 12 digital I/O lines

**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.

The xPC Target block library 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 from 1 through 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 volts to +10 volts and the second channel is 0 volts 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, the block uses that setting 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 block sets the channels to the initial values between the time the software downloads the model and the time the software starts it.

## **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 AT-AO-10

---

**Purpose** Support for the National Instruments AT-AO-10

**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.

The xPC Target block library 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 from 1 through 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 volts to +10 volts and the second channel is 0 volts 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, the block uses that setting 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 block sets the channels to the initial values between the time the software downloads the model and the time the software starts it.

## **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

**Purpose** Support for National Instruments PC-DIO-24 board

**Board** National Instruments PC-DIO-24

**General Description** The PC-DIO-24 is an I/O board with 24 digital input and output lines. The xPC Target block library 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 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 input driver block, you configure the port 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 an 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 required 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.

<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 an 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 required 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

# National Instruments PC-TIO-10

---

**Purpose** Support for National Instruments PC-TIO-10 board

**Board** National Instruments PC-TIO-10

**General Description** The PC-TIO-10 is an I/O board with 16 digital input and output lines. The xPC Target block library supports this board with these driver blocks:

- National Instruments PC-TIO-10 Digital Input
- National Instruments PC-TIO-10 Digital Output

<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

**Purpose** 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.

## 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 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 required 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.

## 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

### 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 required 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 PCI-6011E (formerly PCI-MIO-16XE-50)

---

## **Purpose**

Support for National Instruments PCI-6011E board

## **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.

The xPC Target block library 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; Pulse Width/Period measurement: Yes; Pulse Train 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.
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

# National Instruments PCI-6011E Analog Input (A/D) (formerly PCI-MIO-16XE-50)

---

Coupling	Coupling Code	Description
		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.

If the board is set up for differential mode, you can read the data from either of the channels in the differential pair. For example, if you have a differential pair of 1 and 9, you can read the data from channel 1 or channel 9. However, you might want to read the lower channel number of the pair because it remains unchanged when you switch the input mode between single-ended and differential.

## 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.

## 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.

### 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

## 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-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 required 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

---

**Purpose** Support for National Instruments PCI-6023E board

**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.

The xPC Target block library 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; Pulse Width/Period Measurement: Yes; Pulse Train 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.

If the board is set up for differential mode, you can read the data from either of the channels in the differential pair. For example, if you have a differential pair of 1 and 9, you can read the data from channel 1 or channel 9. However, you might want to read the lower channel number of the pair because it remains unchanged when you switch the input mode between single-ended and differential.

## **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

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 required 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
```

**Purpose** Support for National Instruments PCI-6024E board

**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.

The xPC Target block library 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; Pulse Width/Period measurement: Yes; Pulse Train 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.

If the board is set up for differential mode, you can read the data from either of the channels in the differential pair. For example, if you have a differential pair of 1 and 9, you can read the data from channel 1 or channel 9. However, you might want to read the lower channel number of the pair because it remains unchanged when you switch the input mode between single-ended and differential.

## **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 required 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

---

**Purpose** Support for National Instruments PCI-6025E board

**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.

The xPC Target block library 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; Pulse Width/Period measurement: Yes; Pulse Train 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.

If the board is set up for differential mode, you can read the data from either of the channels in the differential pair. For example, if you have a differential pair of 1 and 9, you can read the data from channel 1 or channel 9. However, you might want to read the lower channel number of the pair because it remains unchanged when you switch the input mode between single-ended and differential.

## **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 required 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 required 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 required 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)

---

## **Purpose**

Support for National Instruments PCI-6030E board

## **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.

The xPC Target block library 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; Pulse Width/Period measurement: Yes; Pulse Train 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.

If the board is set up for differential mode, you can read the data from either of the channels in the differential pair. For example, if you have a differential pair of 1 and 9, you can read the data from channel 1 or channel 9. However, you might want to read the lower channel number of the pair because it remains unchanged when you switch the input mode between single-ended and differential.

# National Instruments PCI-6030E Analog Input (A/D) (formerly PCI-MIO-16XE-10)

---

## 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-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

<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-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 required 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

---

**Purpose** Support for National Instruments PCI-6031E board

**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.

The xPC Target block library 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

## Board Characteristics

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; Pulse Width/Period measurement: Yes; Pulse Train 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.

If the board is set up for differential mode, you can read the data from either of the channels in the differential pair. For example, if you have a differential pair of 1 and 9, you can read the data from channel 1 or channel 9. However, you might want to read the lower channel number of the pair because it remains unchanged when you switch the input mode between single-ended and differential.

# National Instruments PCI-6031E Analog Input (A/D)

---

## **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-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 required 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)

---

**Purpose** Support for National Instruments PCI-6040E board

**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.

The xPC Target block library 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)

# National Instruments PCI-6040E (formerly PCI-MIO-16E-4)

---

## Board Characteristics

Board name	PCI-6040E
Manufacturer	National Instruments
Bus type	PCI
Access method	Memory mapped
Multiple block instance support	A/D: No, D/A: No, Digital I/O: Yes; Pulse Width/Period measurement: Yes; Pulse Train 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)

---

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.

If the board is set up for differential mode, you can read the data from either of the channels in the differential pair. For example, if you have a differential pair of 1 and 9, you can read the data from channel 1 or channel 9. However, you might want to read the lower channel number of the pair because it remains unchanged when you switch the input mode between single-ended and differential.

# National Instruments PCI-6040E Analog Input (A/D) (formerly PCI-MIO-16E-4)

---

## 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-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.

<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-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 required 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
```

## Purpose

Support for National Instruments PCI/PXI-6052E board

## 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.

The xPC Target block library 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; Pulse Width/Period measurement: Yes; Pulse Train 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.

If the board is set up for differential mode, you can read the data from either of the channels in the differential pair. For example, if you have a differential pair of 1 and 9, you can read the data from channel 1 or channel 9. However, you might want to read the lower channel number of the pair because it remains unchanged when you switch the input mode between single-ended and differential.

# National Instruments PXI/PCI-6052E Analog Input (A/D)

---

## 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 PXI/PCI-6052E Analog Output (D/A)

---

**Purpose** PXI/PCI-6052E Analog Output block

**Library** xPC Target Library for National Instruments

## 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 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

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 required 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)

---

## **Purpose**

Support for National Instruments PCI-6070E board

## **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.

The xPC Target block library 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)

# National Instruments PCI-6070E (formerly PCI-MIO-16E-1)

---

## Board Characteristics

Board name	PCI-6070E
Manufacturer	National Instruments
Bus type	PCI
Access method	Memory mapped
Multiple block instance support	A/D: No, D/A: No, Digital I/O: Yes; Pulse Width/Period measurement: Yes; Pulse Train 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.

If the board is set up for differential mode, you can read the data from either of the channels in the differential pair. For example, if you have a differential pair of 1 and 9, you can read the data from channel 1 or channel 9. However, you might want to read the lower channel number of the pair because it remains unchanged when you switch the input mode between single-ended and differential.

# National Instruments PCI-6070E Analog Input (A/D) (formerly PCI-MIO-16E-1)

---

## 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-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 required 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

---

**Purpose** Support for National Instruments PCI-6071E board

**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.

The xPC Target block library 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; Pulse Width/Period measurement: Yes; Pulse Train 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.

If the board is set up for differential mode, you can read the data from either of the channels in the differential pair. For example, if you have a differential pair of 1 and 9, you can read the data from channel 1 or channel 9. However, you might want to read the lower channel number of the pair because it remains unchanged when you switch the input mode between single-ended and differential.

# National Instruments PCI-6071E Analog Input (A/D)

---

## **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-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.

<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-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 required 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-6221

---

**Purpose** Support for National Instruments PCI-6221 board

**Board** National Instruments PCI-6221

**General Description** The PCI-6221 is a data acquisition I/O board with up to 16 single-ended or 8 differential 16-bit analog inputs, two 16-bit analog outputs, 24 digital I/O lines, and two 32-bit counters.

The xPC Target block library supports this board with these driver blocks:

- National Instruments PCI-6221 Analog Input
- National Instruments PCI-6221 Analog Output
- National Instruments PCI-6221 Digital Input
- National Instruments PCI-6221 Digital Output
- National Instruments PCI-6221 Incremental Encoder
- National Instruments PCI-6221 PFI Digital Input
- National Instruments PCI-6221 PFI Digital Output
- National Instruments PCI-6221 PWM Generate
- National Instruments PCI-6221 PWM Measure

## Board Characteristics

Board name	PCI-6221
Manufacturer	National Instruments
Bus type	PCI, PXI (use PCI version of blocks)
Multiple block instance support	A/D: No, D/A: No, Digital I/O: Yes
Multiple board support	Yes

# National Instruments PCI-6221 Analog Input

**Purpose** National Instruments PCI-6221 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 from 1 through 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 lines beginning with 1 even though the board manufacturer starts numbering the lines with 0.

### **Range vector**

Enter a range code for each of the channels in the channel vector. You can specify one value to be used for all channels, or specify one for each channel. This driver allows each channel to be different. This board works only in bipolar mode.

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
-1 to +1	1
-0.2 to +0.2	0.2

# National Instruments PCI-6221 Analog Input

For example, if the first channel is -10 volts to +10 volts and the second and fifth channels are -1 volts to +1 volts, enter

[10, 1, 1]

If all channels use the same range, you specify one entry.

## Input coupling vector

Enter a coupling code for each of the channels in the channel vector. You can specify one value for all channels, or specify one for each channel. 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 National Instruments user 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 National Instruments user 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 National Instruments user manual.

For example, if the first and second channels are single input and the fifth channel is a differential input, enter

# National Instruments PCI-6221 Analog Input

[0,0,2]

National Instruments boards use a hardware number scheme for differential mode channel pairs. xPC Target drivers use a software number scheme that adds a 1 to the hardware number, as shown in the following table:

Hardware Channel Number		Software Channel Number = Hardware Channel Number + 1	
+	-	+	-
0	8	1	9
1	9	2	10
2	10	3	11
3	11	4	12
4	12	5	13
5	13	6	14
6	14	7	15
7	15	8	16

The hardware uses a second input 8 channels higher than the first channel as the negative input of the pair. In the example above, the hardware would use the 13th channel as a differential input with the fifth channel.

If the board is set up for differential mode, you can read the data from either of the channels in the differential pair. For example, if you have a differential pair of 1 and 9, you can read the data from channel 1 or channel 9. However, you might want to read the lower channel number of the pair because it remains unchanged when you switch the input mode between single-ended and differential.

# National Instruments PCI-6221 Analog Input

---

## Scan interval

If **Channel vector** has more than one channel in it, enter the time between sampling different channels. Because execution waits for data to be available, this value increases the time to execute the block. If you enter a shorter time, you get the minimum of 4e-6, which is the fastest this board can acquire data. If you enter a time smaller than this value, the time will still be 4e-6. See the National Instruments user manual for a discussion about settling time and the need to control the time between sampling different channels depending on the impedance of the signal source.

## 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-6221 Analog Output

**Purpose** National Instruments PCI-6221 Analog Output block

**Library** xPC Target Library for National Instruments

## Scaling of Output to Input

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 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 the initial values between the time the model is downloaded and the time it is started.

# National Instruments PCI-6221 Analog Output

---

## **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-6221 Digital Input

**Purpose** National Instruments PCI-6221 Digital Input block

**Library** xPC Target Library for National Instruments

**Note** Models can have multiple digital input blocks for one physical board. In this case, the xPC Target environment executes the blocks based on their sorted order as determined by the Simulink software.

Each board has one set of static digital I/O ports. You can independently configure each port to be input or output.

If a digital input block shares channels with a digital output block, the input block reads the last value written to the digital output block on the overlapping channels.

## Scaling of Input to Output

Hardware Input	Block Output Data Type	Scaling
TTL	Double	TTL low $\geq$ 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 the first eight 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.

### 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

# National Instruments PCI-6221 Digital Input

---

- 1

to automatically locate the board.

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of 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-6221 Digital Output

**Purpose** National Instruments PCI-6221 Digital Output block

**Library** xPC Target Library for National Instruments

**Note** Models can have multiple digital output blocks for one physical board. In this case, the xPC Target environment executes the blocks based on their sorted order as determined by the Simulink software.

Each board has one set of digital I/O ports. You can independently configure each port to be input or output.

If a digital input block shares channels with a digital output block, the input block reads the last value written to the digital output block on the overlapping channels.

The block compares the value written to each port with 0.5.

Value written	Block writes digital output with a value of...
$\geq 0.5$	1
$< 0.5$	0

## Scaling of Output to Input

Hardware Output	Block Input Data Type	Scaling
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 the first eight digital outputs, enter

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

# National Instruments PCI-6221 Digital Output

---

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-6221 Incremental Encoder

**Purpose** National Instruments PCI-6221 Incremental Encoder block

**Library** xPC Target Library for National Instruments

**Scaling Input to Output**

Hardware Input	Block Output Data Type	Scaling
TTL level quadrature encoder signals	Double	counts

## Block Parameters

### Channel

From the list, choose 0 or 1 for the channel counter you want to use. For information on input for these counters, see the National Instruments M Series user manual. The pin associations are:

Counter	Channel 0	Channel 1
A	PFI 8 (Pin 37)	PFI 3 (Pin 42)
B	PFI 10 (Pin 45)	PFI 11 (Pin 46)
Z	PFI 9 (Pin 3)	PFI 4 (Pin 41)

### Counting mode

From the list, select a counting mode:

Mode	Description
Quadrature Mode x1	Count on rising edges of the A signal
Quadrature Mode x2	Count on both rising and falling edges of the A signal
Quadrature Mode x4	Count on both rising and falling edges of both A and B signals

# National Instruments PCI-6221 Incremental Encoder

---

The level of the opposite signal when an edge occurs determines the count direction. For example, a rising edge of A when B is high counts one way, while an edge of A when B is low counts the other. For more information, see the National Instruments M Series user manual.

## Initial count

The initial count specifies the initial value for the counter. The block resets the count to this value under the following conditions:

- When the model starts
- When the following is true, which typically occurs once per revolution for a rotary encoder.
  - **Reload at index pulse** check box is selected
  - Index pulse (Z input) is true
  - The phases of A and B match the setting of **Index phase**

## Reload at index pulse

Select this check box to reset the counter to the value of **Initial count** at each index pulse. The reset occurs when:

- The encoder outputs a high level on its index output
- The phases of A and B match the setting of **Index phase**

## Index phase

If you have selected the **Reload at index pulse** check box, the **Index phase** parameter specifies the phase of the quadrature signals during which the count reloads with **Initial count**. Your choice of value primarily depends on the particular incremental encoder you use. From the list, select one:

- A low B low
- A low B high
- A high B low
- A high B high

# National Instruments PCI-6221 Incremental Encoder

---

To choose an **Index phase** value for reset, use an oscilloscope, triggered on the index pulse, to observe the states of counters A and B. If your choice does not meet the encoder requirements, resets may be inconsistent.

## Filter

You can apply a digital debouncing filter to the input pins before processing.

From the list, select a filter value to ignore pulses that are shorter than the filter time. The value of **None** indicates that there is no filtering of the input.

- None
- Minimum pulse width 125 nanoseconds
- Minimum pulse width 6.25 microseconds
- Minimum pulse width 1.25 milliseconds

## Sample time

Model 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 locate the board automatically.

If two or more boards of this type are in the target PC, for the board associated with this driver block, enter:

- Bus number
- PCI slot number

Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```

# National Instruments PCI-6221 PFI Digital Input

**Purpose** National Instruments PCI-6221 PFI Digital Input block

**Library** xPC Target Library for National Instruments

**Description** A model can have more than one PFI digital input block for any given physical board. If a model has more than one digital input block, it executes the blocks in the order that the Simulink software has determined.

The two counters and the PFI digital input and output blocks share the 16 PFI digital I/O lines. Follow these rules when using these blocks:

<b>You can read a PFI bit with this digital input block when...</b>	<b>You cannot read a PFI bit with this digital input block when...</b>
PWM measure block is using the bit	PWM generate block is using the bit
Incremental encoder block is using the bit	PFI digital output block is using the bit

## Scaling of Input to Output

<b>Hardware Input</b>	<b>Block Output Data Type</b>	<b>Scaling</b>
TTL	Double	TTL low $\geq 0.0$ TTL high = 1.0

## Block Parameters

### PFI bit vector (0-15)

This block uses 0 based numbering to reflect the names of the bits described in the National Instruments M Series user manual.

Enter a vector that contains all bits you want to monitor. For example, to read bits PFI 0, PFI 5, and PFI 7, enter

[0, 5, 7]

## Sample time

Model 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 locate the board automatically.

If two or more boards of this type are in the target PC, for the board associated with this driver block, enter:

- Bus number
- PCI slot number

Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```

# National Instruments PCI-6221 PFI Digital Output

---

**Purpose** National Instruments PCI-6221 PFI Digital Output block

**Library** xPC Target Library for National Instruments

**Description** A model can have more than one PFI digital input block for any given physical board. If a model has more than one digital input block, it executes the blocks in the order that the Simulink software has determined.

The two counters and the PFI digital input and output blocks share the 16 PFI digital I/O lines. Follow these rules when using these blocks:

- Do not use a PFI bit as a digital output when the PDF digital input block is using the bit.
- Do not use a PFI bit as a digital output when the incremental encoder is using the bit.
- Do not use a PFI bit for digital output when the PWM generate block is using the bit.

## Scaling of Output to Input

Hardware Output	Block Input Data Type	Scaling
TTL	Double	< 0.5 = TTL low ≥ 0.5 = TTL high

## Block Parameters

### PFI bit vector (0-15)

This block uses 0 based numbering to reflect the names of the bits described in the National Instruments M Series user manual.

Enter a vector that contains all bits you want to control. For example, to write bits PFI 1, PFI 3, and PFI 9, enter

[1, 3, 9]

## Reset vector

The reset vector controls the behavior at model termination. If you specify a value of 1, the output is reset to the value specified in **Initial value vector**. Enter a scalar or a vector of values, one for each output. If you specify a scalar value, the driver uses that setting for all outputs. If you specify a value of 1, when the model execution stops, the corresponding output is reset to the value specified in **Initial value vector**. If you specify a value of 0, the output 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 block outputs. Enter a scalar or a vector of values, one for each output. If you enter a scalar, the driver uses that value for all outputs.

The block applies this value as follows:

- The model uses this value when you first download the model to the target PC.
- If you set **Reset vector** to 1, the block output is reset to this value when model execution stops. If the value is greater than 0.5, the block sets output to HIGH. If the value is less than or equal to 0.5, the block sets the output to LOW.

## Sample time

Model 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 locate the board automatically.

If two or more boards of this type are in the target PC, for the board associated with this driver block, enter:

# National Instruments PCI-6221 PFI Digital Output

---

- Bus number
- PCI slot number

Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```

# National Instruments PCI-6221 PWM Generate

**Purpose** National Instruments PCI-6221 PWM Generate block

**Library** xPC Target Library for National Instruments

**Description** Use this block to generate variable frequency and pulse width output. All M Series boards have two 32-bit counters that you can use to generate pulse trains. Inputs are in counts of the 80 MHz reference frequency on the onboard clock.

- H input is the number of counts that the output is high.
- L input is the number of counts that the output is low.

If the H input goes to 0, the output stays low. If the L input goes to 0, the output stays high. If both are 0, the output goes low.

## Block Parameters

### Counter

From the list, select the counter, 0 or 1, that you want to use. The output for these counters is on the following pins:

Counter	Pin
0	PFI 12 (Pin 2)
1	PFI 13 (Pin 40)

### Initial high count

Enter the initial H count to take effect between the time the Simulink engine calls the `md1Start` routine for this block and the first time it calls `md1Outputs`. This length of time depends on the number of blocks in the model and the sorted order as determined by the Simulink software.

If you enter a value of 0, the output stays low until the first time the Simulink engine calls `md1Outputs`.

# National Instruments PCI-6221 PWM Generate

---

## Initial low count

Enter the initial L count to take effect between the time the Simulink engine calls the `md1Start` routine for this block and the first time it calls `md1Outputs`. This length of time depends on the number of blocks in the model and the sorted order as determined by the Simulink software. If you enter a value of 0, the output stays high until the first time the Simulink engine calls `md1Outputs`, unless you have also set H to 0.

## Arm input

Select this check box to add a third input port to the block.

- If the input connected to this port is greater than 0.5, the block arms the counter. Output then occurs.
- If the input connected to this port is less than or equal to 0.5, the block disarms the counter and the output goes to the level of the value in the **Disarm level** parameter.

## Disarm level

From the list, select the disarm level **Disarm to LOW** or **Disarm to HIGH**. If the model disarms the block and the level of the value is **Arm input**, the output goes to the level you select here.

## Stop level

From the list, select the stop level **Stop to LOW** or **Stop to HIGH**. When model execution stops, the output goes to the level you choose here.

## Sample time

Model 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 locate the board automatically.

If two or more boards of this type are in the target PC, for the board associated with this driver block, enter:

# National Instruments PCI-6221 PWM Generate

---

- Bus number
- PCI slot number

Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```

# National Instruments PCI-6221 PWM Measure

---

**Purpose** National Instruments PCI-6221 PWM Measure block

**Library** xPC Target Library for National Instruments

**Description** Use this block to measure the pulse width or the period of a signal.  
To measure the pulse width of a signal, connect the signal to the gate, as follows:

Counter	Gate Pin
0	PFI 9 (Pin 3)
1	PFI 4 (Pin 41)

The input waveform must have fast rise and fall times and conform to TTL signal levels of [0,5] volts. Use external signal conditioning to achieve these levels. If you input a sine wave, the decision points are less definite and the measurement is noisy. In extreme cases, the gate can transition multiple times during a slow signal transition.

This block does not queue or buffer measurements. If the gate changes state, the driver holds the current count, overwriting any previous value.

This block returns a value that is always the length of the last gate period before the block executes.

To compute the duty cycle percentage, use two counters. Because the period of a cycle is H counter + L counter, you can measure any two of these counters and compute the third.

You cannot measure a 0 width pulse.

## Block Parameters

### Counter

From the list, select a counter, 0 or 1.

### Trigger mode

From the list, select the gating mode to use:

# National Instruments PCI-6221 PWM Measure

Mode	Description
Width of Gate High	Returns the number of 80 MHz cycles that occur when a single cycle of the input is high.
Width of Gate Low	Returns the number of 80 MHz cycles that occur when a single cycle of the input is low.
Between Rising Edges	Returns the number of 80 MHz cycles that occur between rising edges of the input signal. This cycle is the full period of the input signal.
Between Falling Edges	Returns the number of 80 MHz cycles that occur between falling edges of the input signal. This cycle is the full period of the input signal.

If the input signal has both fast rise and fall edges, **Between Rising Edges** and **Between Falling Edges** return the same value. If the input is asymmetric, one of these options can give a better result than the other.

## Filter

You can apply a digital debouncing filter to the input pins before processing.

From the list, select a filter value to ignore pulses that are shorter than the filter time. The block ignores pulses shorter than the chosen duration. The value of **None** indicates that there is no filtering of the input

- None
- Minimum pulse width 125 nanoseconds
- Minimum pulse width 6.25 microseconds
- Minimum pulse width 1.25 milliseconds

# National Instruments PCI-6221 PWM Measure

---

## Sample time

Model 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 locate the board automatically.

If two or more boards of this type are in the target PC, for the board associated with this driver block, enter:

- Bus number
- PCI slot number

Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```

**Purpose** Support for National Instruments PCI-6221/37 board

**Board** National Instruments PCI-6221/37

**General Description** The PCI-6221/37 is a data acquisition I/O board with up to 16 single-ended or 8 differential 16-bit analog inputs, two 16-bit analog outputs, 10 digital I/O lines, and two 32-bit counters.

The xPC Target block library supports this board with these driver blocks:

- National Instruments PCI-6221/37 Analog Input
- National Instruments PCI-6221/37 Analog Output
- National Instruments PCI-6221/37 Digital Input
- National Instruments PCI-6221/37 Digital Output
- National Instruments PCI-6221/37 Incremental Encoder
- National Instruments PCI-6221/37 PFI Digital Input
- National Instruments PCI-6221/37 PFI Digital Output
- National Instruments PCI-6221/37 PWM Generate
- National Instruments PCI-6221/37 PWM Measure

## Board Characteristics

Board name	PCI-6221/37
Manufacturer	National Instruments
Bus type	PCI, PXI (use PCI version of blocks)
Multiple block instance support	A/D: No, D/A: No, Digital I/O: Yes
Multiple board support	Yes

# National Instruments PCI-6221/37 Analog Input

**Purpose** National Instruments PCI-6221/37 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 from 1 through 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 lines beginning with 1 even though the board manufacturer starts numbering the lines with 0.

### **Range vector**

Enter a range code for each of the channels in the channel vector. You can specify one value to be used for all channels, or specify one for each channel. This driver allows each channel to be different. This board works only in bipolar mode.

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
-1 to +1	1
-0.2 to +0.2	0.2

# National Instruments PCI-6221/37 Analog Input

For example, if the first channel is -10 volts to +10 volts and the second and fifth channels are -1 volts to +1 volts, enter

```
[10, 1, 1]
```

If all channels use the same range, you specify one entry.

## Input coupling vector

Enter a coupling code for each of the channels in the channel vector. You can specify one value for all channels, or specify one for each channel. 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 National Instruments user 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 National Instruments user 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 National Instruments user manual.

For example, if the first and second channels are single input and the fifth channel is a differential input, enter

# National Instruments PCI-6221/37 Analog Input

[0,0,2]

National Instruments boards use a hardware number scheme for differential mode channel pairs. xPC Target drivers use a software number scheme that adds a 1 to the hardware number, as shown in the following table:

Hardware Channel Number		Software Channel Number = Hardware Channel Number + 1	
+	-	+	-
0	8	1	9
1	9	2	10
2	10	3	11
3	11	4	12
4	12	5	13
5	13	6	14
6	14	7	15
7	15	8	16

The hardware uses a second input 8 channels higher than the first channel as the negative input of the pair. In the example above, the hardware would use the 13th channel as a differential input with the fifth channel.

If the board is set up for differential mode, you can read the data from either of the channels in the differential pair. For example, if you have a differential pair of 1 and 9, you can read the data from channel 1 or channel 9. However, you might want to read the lower channel number of the pair because it remains unchanged when you switch the input mode between single-ended and differential.

## Scan interval

If **Channel vector** has more than one channel in it, enter the time between sampling different channels. Because execution waits for data to be available, this value increases the time to execute the block. If you enter a shorter time, you get the minimum of 4e-6, which is the fastest this board can acquire data. If you enter a time smaller than this value, the time will still be 4e-6. See the National Instruments user manual for a discussion about settling time and the need to control the time between sampling different channels depending on the impedance of the signal source.

## 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-6221/37 Analog Output

---

**Purpose** National Instruments PCI-6221/37 Analog Output block

**Library** xPC Target Library for National Instruments

<b>Scaling of Output to Input</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 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 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-6221/37 Digital Input

---

**Purpose** National Instruments PCI-6221/37 Digital Input block

**Library** xPC Target Library for National Instruments

**Note** Models can have multiple digital input blocks for one physical board. In this case, the xPC Target environment executes the blocks based on their sorted order as determined by the Simulink software.

Each board has one set of static digital I/O ports. You can independently configure each port to be input or output.

If a digital input block shares channels with a digital output block, the input block reads the last value written to the digital output block on the overlapping channels.

## Scaling of Input to Output

Hardware Input	Block Output Data Type	Scaling
TTL	Double	TTL low $\geq 0.0$ TTL high = 1.0

## Block Parameters

### Channel vector

Enter numbers 1 or 2 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 both digital inputs for one port, enter

[1,2]

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.

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number 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
```

# National Instruments PCI-6221/37 Digital Output

**Purpose** National Instruments PCI-6221/37 Digital Output block

**Library** xPC Target Library for National Instruments

**Note** Models can have multiple digital output blocks for one physical board. In this case, the xPC Target environment executes the blocks based on their sorted order as determined by the Simulink software.

Each board has one set of digital I/O ports. You can independently configure each port to be input or output.

If a digital input block shares channels with a digital output block, the input block reads the last value written to the digital output block on the overlapping channels.

The block compares the value written to each port with 0.5.

Value written	Block writes digital output with a value of...
$\geq 0.5$	1
$< 0.5$	0

## Scaling of Output to Input

Hardware Output	Block Input Data Type	Scaling
TTL	Double	$< 0.5 = \text{TTL low}$ $\geq 0.5 = \text{TTL high}$

## Block Parameters

### Channel vector

Enter numbers 1 or 2 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 the first output, enter

```
[1]
```

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-6221/37 Incremental Encoder

**Purpose** National Instruments PCI-6221/37 Incremental Encoder 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 level quadrature encoder signals	Double	counts

## Block Parameters

### Channel

From the list, choose 0 or 1 for the channel counter you want to use. For information on input for these counters, see the National Instruments M Series user manual. The pin associations are:

Counter	Channel 0	Channel 1
A	PFI 0 (Pin 13)	PFI 3 (Pin 15)
B	PFI 2 (Pin 33)	PFI 5 (Pin 35)
Z	PFI 1 (Pin 32)	PFI 4 (Pin 34)

### Counting mode

From the list, select a counting mode:

Mode	Description
Quadrature Mode x1	Count on rising edges of the A signal
Quadrature Mode x2	Count on both rising and falling edges of the A signal
Quadrature Mode x4	Count on both rising and falling edges of both A and B signals

The level of the opposite signal when an edge occurs determines the count direction. For example, a rising edge of A when B is high counts one way, while an edge of A when B is low counts the other. For more information, see the National Instruments M Series user manual.

## Initial count

The initial count specifies the initial value for the counter. The block resets the count to this value under the following conditions:

- When the model starts
- When the following is true, which typically occurs once per revolution for a rotary encoder.
  - **Reload at index pulse** check box is selected
  - Index pulse (Z input) is true
  - The phases of A and B match the setting of **Index phase**

## Reload at index pulse

Select this check box to reset the counter to the value of **Initial count** at each index pulse. The reset occurs when:

- The encoder outputs a high level on its index output
- The phases of A and B match the setting of **Index phase**

## Index phase

If you have selected the **Reload at index pulse** check box, the **Index phase** parameter specifies the phase of the quadrature signals during which the count reloads with **Initial count**. Your choice of value primarily depends on the particular incremental encoder you use. From the list, select one:

- A low B low
- A low B high
- A high B low
- A high B high

# National Instruments PCI-6221/37 Incremental Encoder

---

To choose an **Index phase** value for reset, use an oscilloscope, triggered on the index pulse, to observe the states of counters A and B. If your choice does not meet the encoder requirements, resets may be inconsistent.

## Filter

You can apply a digital debouncing filter to the input pins before processing.

From the list, select a filter value to ignore pulses that are shorter than the filter time. The value of `None` indicates that there is no filtering of the input.

- `None`
- Minimum pulse width 125 nanoseconds
- Minimum pulse width 6.25 microseconds
- Minimum pulse width 1.25 milliseconds

## Sample time

Model 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 locate the board automatically.

If two or more boards of this type are in the target PC, for the board associated with this driver block, enter:

- Bus number
- PCI slot number

Use the format `[BusNumber, SlotNumber]`. To determine the bus number and the PCI slot number, type

```
getxpcpci
```

# National Instruments PCI-6221/37 PFI Digital Input

**Purpose** National Instruments PCI-6221/37 PFI Digital Input block

**Library** xPC Target Library for National Instruments

**Description** A model can have more than one PFI digital input block for any given physical board. If a model has more than one digital input block, it executes the blocks in the order that the Simulink software has determined.

The two counters and the PFI digital input and output blocks share the 8 PFI digital I/O lines. Follow these rules when using these blocks:

<b>You can read a PFI bit with this digital input block when...</b>	<b>You cannot read a PFI bit with this digital input block when...</b>
PWM measure block is using the bit	PWM generate block is using the bit
Incremental encoder block is using the bit	PFI digital output block is using the bit

## Scaling of Input to Output

<b>Hardware Input</b>	<b>Block Output Data Type</b>	<b>Scaling</b>
TTL	Double	TTL low $\geq 0.0$ TTL high = 1.0

## Block Parameters

### PFI bit vector (0-7)

This block uses 0 based numbering to reflect the names of the bits described in the National Instruments M Series user manual.

Enter a vector that contains all bits you want to monitor. For example, to read bits PFI 0, PFI 5, and PFI 7, enter

[0, 5, 7]

# National Instruments PCI-6221/37 PFI Digital Input

---

## Sample time

Model 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 locate the board automatically.

If two or more boards of this type are in the target PC, for the board associated with this driver block, enter:

- Bus number
- PCI slot number

Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```

# National Instruments PCI-6221/37 PFI Digital Output

**Purpose** National Instruments PCI-6221/37 PFI Digital Output block

**Library** xPC Target Library for National Instruments

**Description** A model can have more than one PFI digital input block for any given physical board. If a model has more than one digital input block, it executes the blocks in the order that the Simulink software has determined.

The two counters and the PFI digital input and output blocks share the 8 PFI digital I/O lines. Follow these rules when using these blocks:

- Do not use a PFI bit as a digital output when the PDF digital input block is using the bit.
- Do not use a PFI bit as a digital output when the incremental encoder is using the bit.
- Do not use a PFI bit for digital output when the PWM generate block is using the bit.

## Scaling of Output to Input

Hardware Output	Block Input Data Type	Scaling
TTL	Double	< 0.5 = TTL low ≥ 0.5 = TTL high

## Block Parameters

### PFI bit vector (0-7)

This block uses 0 based numbering to reflect the names of the bits described in the National Instruments M Series user manual.

Enter a vector that contains all bits you want to control. For example, to write bits PFI 1, PFI 3, and PFI 7, enter

[1, 3, 7]

# National Instruments PCI-6221/37 PFI Digital Output

---

## Reset vector

The reset vector controls the behavior at model termination. If you specify a value of 1, the output is reset to the value specified in **Initial value vector**. Enter a scalar or a vector of values, one for each output. If you specify a scalar value, the driver uses that setting for all outputs. If you specify a value of 1, when the model execution stops, the corresponding output is reset to the value specified in **Initial value vector**. If you specify a value of 0, the output 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 block outputs. Enter a scalar or a vector of values, one for each output. If you enter a scalar, the driver uses that value for all outputs.

The block applies this value as follows:

- The model uses this value when you first download the model to the target PC.
- If you set **Reset vector** to 1, the block output is reset to this value when model execution stops. If the value is greater than 0.5, the block sets output to HIGH. If the value is less than or equal to 0.5, the block sets the output to LOW.

## Sample time

Model 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 locate the board automatically.

If two or more boards of this type are in the target PC, for the board associated with this driver block, enter:

# National Instruments PCI-6221/37 PFI Digital Output

---

- Bus number
- PCI slot number

Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```

# National Instruments PCI-6221/37 PWM Generate

**Purpose** National Instruments PCI-6221/37 PWM Generate block

**Library** xPC Target Library for National Instruments

**Description** Use this block to generate variable frequency and pulse width output. All M Series boards have two 32-bit counters that you can use to generate pulse trains. Inputs are in counts of the 80 MHz reference frequency on the onboard clock.

- H input is the number of counts that the output is high.
- L input is the number of counts that the output is low.

If the H input goes to 0, the output stays low. If the L input goes to 0, the output stays high. If both are 0, the output goes low.

## Block Parameters

### Counter

From the list, select the counter, 0 or 1, that you want to use. The output for these counters is on the following pins:

Counter	Pin
0	PFI 6 (Pin 17)
1	PFI 7 (Pin 36)

### Initial high count

Enter the initial H count to take effect between the time the Simulink engine calls the `md1Start` routine for this block and the first time it calls `md1Outputs`. This length of time depends on the number of blocks in the model and the sorted order as determined by the Simulink software.

If you enter a value of 0, the output stays low until the first time the Simulink engine calls `md1Outputs`.

## Initial low count

Enter the initial L count to take effect between the time the Simulink engine calls the `mdlStart` routine for this block and the first time it calls `mdlOutputs`. This length of time depends on the number of blocks in the model and the sorted order as determined by the Simulink software. If you enter a value of 0, the output stays high until the first time the Simulink engine calls `mdlOutputs`, unless you have also set H to 0.

## Arm input

Select this check box to add a third input port to the block.

- If the input connected to this port is greater than 0.5, the block arms the counter. Output then occurs.
- If the input connected to this port is less than or equal to 0.5, the block disarms the counter and the output goes to the level of the value in the **Disarm level** parameter.

## Disarm level

From the list, select the disarm level **Disarm to LOW** or **Disarm to HIGH**. If the model disarms the block and the level of the value is **Arm input**, the output goes to the level you select here.

## Stop level

From the list, select the stop level **Stop to LOW** or **Stop to HIGH**. When model execution stops, the output goes to the level you choose here.

## Sample time

Model 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 locate the board automatically.

If two or more boards of this type are in the target PC, for the board associated with this driver block, enter:

# National Instruments PCI-6221/37 PWM Generate

---

- Bus number
- PCI slot number

Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```

# National Instruments PCI-6221/37 PWM Measure

**Purpose** National Instruments PCI-6221/37 PWM Measure block

**Library** xPC Target Library for National Instruments

**Description** Use this block to measure the pulse width or the period of a signal.  
To measure the pulse width of a signal, connect the signal to the gate, as follows:

Counter	Gate Pin
0	PFI 1 (Pin 32)
1	PFI 4 (Pin 34)

The input waveform must have fast rise and fall times and conform to TTL signal levels of [0,5] volts. Use external signal conditioning to achieve these levels. If you input a sine wave, the decision points are less definite and the measurement is noisy. In extreme cases, the gate can transition multiple times during a slow signal transition.

This block does not queue or buffer measurements. If the gate changes state, the driver holds the current count, overwriting any previous value.

This block returns a value that is always the length of the last gate period before the block executes.

To compute the duty cycle percentage, use two counters. Because the period of a cycle is H counter + L counter, you can measure any two of these counters and compute the third.

You cannot measure a 0 width pulse.

## Block Parameters

### Counter

From the list, select a counter, 0 or 1.

### Trigger mode

From the list, select the gating mode to use:

# National Instruments PCI-6221/37 PWM Measure

Mode	Description
Width of Gate High	Returns the number of 80 MHz cycles that occur when a single cycle of the input is high.
Width of Gate Low	Returns the number of 80 MHz cycles that occur when a single cycle of the input is low.
Between Rising Edges	Returns the number of 80 MHz cycles that occur between rising edges of the input signal. This cycle is the full period of the input signal.
Between Falling Edges	Returns the number of 80 MHz cycles that occur between falling edges of the input signal. This cycle is the full period of the input signal.

If the input signal has both fast rise and fall edges, **Between Rising Edges** and **Between Falling Edges** return the same value. If the input is asymmetric, one of these options can give a better result than the other.

## Filter

You can apply a digital debouncing filter to the input pins before processing.

From the list, select a filter value to ignore pulses that are shorter than the filter time. The block ignores pulses shorter than the chosen duration. The value of **None** indicates that there is no filtering of the input

- None
- Minimum pulse width 125 nanoseconds
- Minimum pulse width 6.25 microseconds
- Minimum pulse width 1.25 milliseconds

# National Instruments PCI-6221/37 PWM Measure

---

## Sample time

Model 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 locate the board automatically.

If two or more boards of this type are in the target PC, for the board associated with this driver block, enter:

- Bus number
- PCI slot number

Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```

# National Instruments PCI-6225

---

**Purpose** Support for National Instruments PCI-6225 board

**Board** National Instruments PCI-6225

**General Description** The PCI-6225 is a data acquisition I/O board with two 16-bit analog outputs, 24 digital I/O lines, and two 32-bit counters.

The xPC Target block library supports this board with these driver blocks:

- National Instruments PCI-6225 Analog Input
- National Instruments PCI-6225 Analog Output
- National Instruments PCI-6225 Digital Input
- National Instruments PCI-6225 Digital Output
- National Instruments PCI-6225 Incremental Encoder
- National Instruments PCI-6225 PFI Digital Input
- National Instruments PCI-6225 PFI Digital Output
- National Instruments PCI-6225 PWM Generate
- National Instruments PCI-6225 PWM Measure

## Board Characteristics

Board name	PCI-6225
Manufacturer	National Instruments
Bus type	PCI, PXI (use PCI version of blocks)

Multiple block instance support	<ul style="list-style-type: none"><li>• A/D One block per board</li><li>• D/A One block per board</li><li>• Digital I/O Multiple blocks per board</li><li>• Incremental Encoder One block per counter per board</li><li>• PFI Digital Input/Output Multiple blocks per board</li><li>• PWM Generate and Measure One block per counter per board</li></ul>
Multiple board support	Yes

# National Instruments PCI-6225 Analog Input

---

**Purpose** National Instruments PCI-6225 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 80. 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 lines beginning with 1 even though the board manufacturer starts numbering the lines with 0.

### **Range vector**

Enter a range code for each of the channels in the channel vector. You can specify one value to be used for all channels, or specify one for each channel. This driver allows each channel to be different. This board works only in bipolar mode.

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
-1 to +1	1
-0.2 to +0.2	0.2

# National Instruments PCI-6225 Analog Input

For example, if the first channel is -10 to +10 volts and the second and fifth channels are -1 to +1 volts, enter

[10, 1, 1]

If all channels use the same range, you specify one entry.

## Input coupling vector

Enter a coupling code for each of the channels in the channel vector. You can specify one value for all channels, or specify one for each channel. 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 National Instruments user 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 National Instruments user 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 National Instruments user manual.

For example, if the first and second channels are single input and the fifth channel is a differential input, enter

# National Instruments PCI-6225 Analog Input

[0,0,2]

National Instruments boards use a hardware number scheme for differential mode channel pairs. xPC Target drivers use a software number scheme that adds a 1 to the hardware number, as shown in the following table:

Hardware Channel		Software Channel Number = Hardware Channel Number + 1	
+	-	+	-
0 to 7	8 to 15	1 to 8	9 to 16
16 to 23	24 to 31	17 to 24	25 to 32
32 to 39	40 to 47	33 to 40	41 to 48
48 to 55	56 to 63	49 to 56	57 to 64
64 to 71	72 to 79	65 to 72	73 to 80

The hardware uses a second input 8 channels higher than the first channel as the negative input of the pair. In the example above, the hardware would use channel 13 as a differential input with the fifth channel.

If the board is set up for differential mode, you can read the data from either of the channels in the differential pair. For example, if you have a differential pair of 1 and 9, you can read the data from channel 1 or channel 9. However, you might want to read the lower channel number of the pair because it remains unchanged when you switch the input mode between single-ended and differential.

## Scan interval

If **Channel vector** has more than one channel in it, enter the time between sampling different channels. Because execution waits for data to be available, this value increases the time to execute the block. If you enter a shorter time, you get the minimum of 4e-6.

# National Instruments PCI-6225 Analog Input

---

If you enter a time smaller than this value, the time will still be  $4e-6$ , which is the fastest that this board can acquire data.

See the National Instruments user manual for a discussion about settling time and the need to control the time between sampling different channels depending on the impedance of the signal source. To attain equilibrium, the setting of this time is more important with a higher gain.

## **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-6225 Analog Output

---

**Purpose** National Instruments PCI-6225 Analog Output block

**Library** xPC Target Library for National Instruments

<b>Scaling of Output to Input</b>	<b>Hardware Output</b>	<b>Block Input Data Type</b>	<b>Scaling</b>
	Volts	Double	1

## **Block Parameters**

### **Channel vector**

Enter 1 to 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 all two of the analog output channels, enter

[1,2]

Number the channels begin 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 the initial values between the time the model is downloaded and the time it is started.

# National Instruments PCI-6225 Analog Output

---

## **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-6225 Digital Input

---

**Purpose** National Instruments PCI-6225 Digital Input block

**Library** xPC Target Library for National Instruments

**Note** Models can have multiple digital input blocks for one physical board. In this case, the xPC Target environment executes the blocks based on their sorted order as determined by the Simulink software.

Each board has one set of static digital I/O ports. You can independently configure each port to be input or output.

If a digital input block shares channels with a digital output block, the input block reads the last value written to the digital output block on the overlapping channels.

## Scaling of Input to Output

Hardware Input	Block Output Data Type	Scaling
TTL	Double	TTL low $\geq 0.0$ 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 the first eight 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.

### 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-6225 Digital Output

**Purpose** National Instruments PCI-6225 Digital Output block

**Library** xPC Target Library for National Instruments

**Note** Models can have multiple digital output blocks for one physical board. In this case, the xPC Target environment executes the blocks based on their sorted order as determined by the Simulink software.

Each board has one set of static digital I/O ports. You can independently set each port to be input or output.

If a digital input block shares channels with a digital output block, the input block reads the last value written to the digital output block on the overlapping channels.

The block compares the value written to each port with 0.5.

Value written	Block writes digital output with a value of...
$\geq 0.5$	1
$< 0.5$	0

## Scaling of Output to Input

Hardware Output	Block Input Data Type	Scaling
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 the first eight digital outputs for, 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.

## **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-6225 Incremental Encoder

**Purpose** National Instruments PCI-6225 Incremental Encoder block

**Library** xPC Target Library for National Instruments

**Scaling Input to Output**

Hardware Input	Block Output Data Type	Scaling
TTL level quadrature encoder signals	Double	counts

## Block Parameters

### Channel

From the list, choose 0 or 1 for the channel counter you want to use. For information on input for these counters, see the National Instruments M Series user manual. The pin associations are:

Counter	Channel 0	Channel 1
A	PFI 8 (Pin 37)	PFI 3 (Pin 42)
B	PFI 10 (Pin 45)	PFI 11 (Pin 46)
Z	PFI 9 (Pin 3)	PFI 4 (Pin 41)

### Counting mode

From the list, select a counting mode:

Mode	Description
Quadrature Mode x1	Count on rising edges of the A signal
Quadrature Mode x2	Count on both rising and falling edges of the A signal
Quadrature Mode x4	Count on both rising and falling edges of both A and B signals

The level of the opposite signal when an edge occurs determines the count direction. For example, a rising edge of A when B is high counts one way, while an edge of A when B is low counts the other. For more information, see the National Instruments M Series user manual.

## Initial count

The initial count specifies the initial value for the counter. The block resets the count to this value under the following conditions:

- When the model starts
- When the following is true, which typically occurs once per revolution for a rotary encoder.
  - **Reload at index pulse** check box is selected
  - Index pulse (Z input) is true
  - The phases of A and B match the setting of **Index phase**

## Reload at index pulse

Select this check box to reset the counter to the value of **Initial count** at each index pulse. The reset occurs when:

- The encoder outputs a high level on its index output
- The phases of A and B match the setting of **Index phase**

## Index phase

If you have selected the **Reload at index pulse** check box, the **Index phase** parameter specifies the phase of the quadrature signals during which the count reloads with **Initial count**. Your choice of value primarily depends on the particular incremental encoder you use. From the list, select one:

- A low B low
- A low B high
- A high B low
- A high B high

# National Instruments PCI-6225 Incremental Encoder

---

To choose an **Index phase** for reset, use an oscilloscope, triggered on the index pulse, to observe the states of counters A and B. If your choice does not meet the encoder requirements, resets may be inconsistent.

## Filter

You can apply a digital debouncing filter to the input pins before processing.

From the list, select a filter value to ignore pulses that are shorter than the filter time. The value of `None` indicates that there is no filtering of the input.

- `None`
- `Minimum pulse width 125 nanoseconds`
- `Minimum pulse width 6.25 microseconds`
- `Minimum pulse width 1.25 milliseconds`

## Sample time

Model 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 locate the board automatically.

If two or more boards of this type are in the target PC, for the board associated with this driver block, enter:

- `Bus number`
- `PCI slot number`

Use the format `[BusNumber, SlotNumber]`. To determine the bus number and the PCI slot number, type

```
getxpcpci
```

# National Instruments PCI-6225 PFI Digital Input

**Purpose** National Instruments PCI-6225 PFI Digital Input block

**Library** xPC Target Library for National Instruments

**Description** A model can have more than one PFI digital input block for any given physical board. If a model has more than one digital input block, it executes the blocks in the order that the Simulink software has determined.

The two counters and the PFI digital input and output blocks share the 16 PFI digital I/O lines. Follow these rules when using these blocks:

<b>You can read a PFI bit with this digital input block when...</b>	<b>You cannot read a PFI bit with this digital input block when...</b>
PWM measure block is using the bit	PWM generate block is using the bit
Incremental encoder block is using the bit	PFI digital output block is using the bit

## Scaling of Input to Output

<b>Hardware Input</b>	<b>Block Output Data Type</b>	<b>Scaling</b>
TTL	Double	TTL low $\geq 0.0$ TTL high = 1.0

## Block Parameters

### PFI bit vector (0-15)

This block uses 0 based numbering to reflect the names of the bits described in the National Instruments M Series user manual.

Enter a vector that contains all bits you want to monitor. For example, to read bits PFI 0, PFI 5, and PFI 7, enter

[0, 5, 7]

# National Instruments PCI-6225 PFI Digital Input

---

## Sample time

Model 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 locate the board automatically.

If two or more boards of this type are in the target PC, for the board associated with this driver block, enter:

- Bus number
- PCI slot number

Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```

# National Instruments PCI-6225 PFI Digital Output

**Purpose** National Instruments PCI-6225 PFI Digital Output block

**Library** xPC Target Library for National Instruments

**Description** A model can have more than one PFI digital input block for any given physical board. If a model has more than one digital input block, it executes the blocks in the order that the Simulink software has determined.

The two counters and the PFI digital input and output blocks share the 16 PFI digital I/O lines. Follow these rules when using these blocks:

- Do not use a PFI bit as a digital output when the PDF digital input block is using the bit.
- Do not use a PFI bit as a digital output when the incremental encoder is using the bit.
- Do not use a PFI bit for digital output when the PWM generate block is using the bit.

## Scaling of Output to Input

Hardware Output	Block Input Data Type	Scaling
TTL	Double	< 0.5 = TTL low ≥ 0.5 = TTL high

## Block Parameters

### PFI bit vector (0-15)

This block uses 0 based numbering to reflect the names of the bits described in the National Instruments M Series user manual.

Enter a vector that contains all bits you want to control. For example, to write bits PFI 1, PFI 3, and PFI 9, enter

[1, 3, 9]

# National Instruments PCI-6225 PFI Digital Output

---

## Reset vector

The reset vector controls the behavior at model termination. If you specify a value of 1, the output is reset to the value specified in **Initial value vector**. Enter a scalar or a vector of values, one for each output. If you specify a scalar value, the driver uses that setting for all outputs. If you specify a value of 1, when the model execution stops, the corresponding output is reset to the value specified in **Initial value vector**. If you specify a value of 0, the output 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 block outputs. Enter a scalar or a vector of values, one for each output. If you enter a scalar, the driver uses that value for all outputs.

The block applies this value as follows:

- The model uses this value when you first download the model to the target PC.
- If you set **Reset vector** to 1, the block output is reset to this value when model execution stops. If the value is greater than 0.5, the block sets output to HIGH. If the value is less than or equal to 0.5, the block sets the output to LOW.

## Sample time

Model 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 locate the board automatically.

If two or more boards of this type are in the target PC, for the board associated with this driver block, enter:

# National Instruments PCI-6225 PFI Digital Output

---

- Bus number
- PCI slot number

Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```

# National Instruments PCI-6225 PWM Generate

---

**Purpose** National Instruments PCI-6225 PWM Generate block

**Library** xPC Target Library for National Instruments

**Description** Use this block to generate variable frequency and pulse width output. All M Series boards have two 32-bit counters that you can use to generate pulse trains. Inputs are in counts of the 80 MHz reference frequency on the onboard clock.

- H input is the number of counts that the output is high.
- L input is the number of counts that the output is low.

If the H input goes to 0, the output stays low. If the L input goes to 0, the output stays high. If both are 0, the output goes low.

## Block Parameters

### Counter

From the list, select the counter, 0 or 1, that you want to use. The output for these counters is on the following pins:

Counter	Pin
0	PFI 12 (Pin 2)
1	PFI 13 (Pin 40)

### Initial high count

Enter the initial H count to take effect between the time the Simulink engine calls the `md1Start` routine for this block and the first time it calls `md1Outputs`. This length of time depends on the number of blocks in the model and the sorted order as determined by the Simulink software.

If you enter a value of 0, the output stays low until the first time the Simulink engine calls `md1Outputs`.

## Initial low count

Enter the initial L count to take effect between the time the Simulink engine calls the `mdlStart` routine for this block and the first time it calls `mdlOutputs`. This length of time depends on the number of blocks in the model and the sorted order as determined by the Simulink software. If you enter a value of 0, the output stays high until the first time the Simulink engine calls `mdlOutputs`, unless you have also set H to 0.

## Arm input

Select this check box to add a third input port to the block.

- If the input connected to this port is greater than 0.5, the block arms the counter. Output then occurs.
- If the input connected to this port is less than or equal to 0.5, the block disarms the counter and the output goes to the level of the value in the **Disarm level** parameter.

## Disarm level

From the list, select the disarm level **Disarm to LOW** or **Disarm to HIGH**. If the model disarms the block and the level of the value is **Arm input**, the output goes to the level you select here.

## Stop level

From the list, select the stop level **Stop to LOW** or **Stop to HIGH**. When model execution stops, the output goes to the level you choose here.

## Sample time

Model 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 locate the board automatically.

If two or more boards of this type are in the target PC, for the board associated with this driver block, enter:

# National Instruments PCI-6225 PWM Generate

---

- Bus number
- PCI slot number

Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```

# National Instruments PCI-6225 PWM Measure

**Purpose** National Instruments PCI-6225 PWM Measure block

**Library** xPC Target Library for National Instruments

**Description** Use this block to measure the pulse width or the period of a signal.  
To measure the pulse width of a signal, connect the signal to the gate, as follows:

Counter	Gate Pin
0	PFI 9 (Pin 3)
1	PFI 4 (Pin 41)

The input waveform must have fast rise and fall times and conform to TTL signal levels of [0,5] volts. Use external signal conditioning to achieve these levels. If you input a sine wave, the decision points are less definite and the measurement is noisy. In extreme cases, the gate can transition multiple times during a slow signal transition.

This block does not queue or buffer measurements. If the gate changes state, the driver holds the current count, overwriting any previous value.

This block returns a value that is always the length of the last gate period before the block executes.

To compute the duty cycle percentage, use two counters. Because the period of a cycle is H counter + L counter, you can measure any two of these counters and compute the third.

You cannot measure a 0 width pulse.

## Block Parameters

### Counter

From the list, select a counter, 0 or 1.

### Trigger mode

From the list, select the gating mode to use:

# National Instruments PCI-6225 PWM Measure

Mode	Description
Width of Gate High	Returns the number of 80 MHz cycles that occur when a single cycle of the input is high.
Width of Gate Low	Returns the number of 80 MHz cycles that occur when a single cycle of the input is low.
Between Rising Edges	Returns the number of 80 MHz cycles that occur between rising edges of the input signal. This cycle is the full period of the input signal.
Between Falling Edges	Returns the number of 80 MHz cycles that occur between falling edges of the input signal. This cycle is the full period of the input signal.

If the input signal has both fast rise and fall edges, **Between Rising Edges** and **Between Falling Edges** return the same value. If the input is asymmetric, one of these options can give a better result than the other.

## Filter

You can apply a digital debouncing filter to the input pins before processing.

From the list, select a filter value to ignore pulses that are shorter than the filter time. The block ignores pulses shorter than the chosen duration. The value of **None** indicates that there is no filtering of the input

- None
- Minimum pulse width 125 nanoseconds
- Minimum pulse width 6.25 microseconds
- Minimum pulse width 1.25 milliseconds

## Sample time

Model 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 locate the board automatically.

If two or more boards of this type are in the target PC, for the board associated with this driver block, enter:

- Bus number
- PCI slot number

Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```

# National Instruments PCI-6229

---

**Purpose** Support for National Instruments PCI-6229 board

**Board** National Instruments PCI-6229

**General Description** The PCI-6229 is a data acquisition I/O board with 32 single-ended or 16 differential 16-bit analog inputs, four 16-bit analog outputs, 48 digital I/O lines, and two 32-bit counters.

The xPC Target block library supports this board with these driver blocks:

- National Instruments PCI-6229 Analog Input
- National Instruments PCI-6229 Analog Output
- National Instruments PCI-6229 Digital Input
- National Instruments PCI-6229 Digital Output
- National Instruments PCI-6229 Incremental Encoder
- National Instruments PCI-6229 PFI Digital Input
- National Instruments PCI-6229 PFI Digital Output
- National Instruments PCI-6229 PWM Generate
- National Instruments PCI-6229 PWM Measure

## Board Characteristics

Board name	PCI-6229
Manufacturer	National Instruments
Bus type	PCI, PXI (use PCI version of blocks)
Multiple block instance support	A/D: No, D/A: No, Digital I/O: Yes
Multiple board support	Yes

# National Instruments PCI-6229 Analog Input

**Purpose** National Instruments PCI-6229 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 32. 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 lines beginning with 1 even though the board manufacturer starts numbering the lines with 0.

### Range vector

Enter a range code for each of the channels in the channel vector. You can specify one value to be used for all channels, or specify one for each channel. This driver allows each channel to be different. This board works only in bipolar mode.

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
-1 to +1	1
-0.2 to +0.2	0.2

# National Instruments PCI-6229 Analog Input

For example, if the first channel is -10 to +10 volts and the second and fifth channels are -1 to +1 volts, enter

[10, 1, 1]

If all channels use the same range, you specify one entry.

## Input coupling vector

Enter a coupling code for each of the channels in the channel vector. You can specify one value for all channels, or specify one for each channel. 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 National Instruments user 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 National Instruments user 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 National Instruments user manual.

For example, if the first and second channels are single input and the fifth channel is a differential input, enter

# National Instruments PCI-6229 Analog Input

[0,0,2]

National Instruments boards use a hardware number scheme for differential mode channel pairs. xPC Target drivers use a software number scheme that adds a 1 to the hardware number, as shown in the following table:

Hardware Channel Number		Software Channel Number = Hardware Channel Number + 1	
+	-	+	-
0	8	1	9
1	9	2	10
2	10	3	11
3	11	4	12
4	12	5	13
5	13	6	14
6	14	7	15
7	15	8	16
16	24	17	25
17	25	18	26
18	26	19	27
19	27	20	28
20	28	21	29
21	29	22	30
22	30	23	31
23	31	24	32

The hardware uses a second input 8 channels higher than the first channel as the negative input of the pair. In the example

# National Instruments PCI-6229 Analog Input

---

above, the hardware would use the 13 as a differential input with the fifth channel.

If the board is set up for differential mode, you can read the data from either of the channels in the differential pair. For example, if you have a differential pair of 1 and 9, you can read the data from channel 1 or channel 9. However, you might want to read the lower channel number of the pair because it remains unchanged when you switch the input mode between single-ended and differential.

## Scan interval

If **Channel vector** has more than one channel in it, enter the time between sampling different channels. Because execution waits for data to be available, this value increases the time to execute the block. If you enter a shorter time, you get the minimum of 4e-6. If you enter a time smaller than this value, the time will still be 4e-6, which is the fastest this board can acquire data. See the National Instruments user manual for a discussion about settling time and the need to control the time between sampling different channels depending on the impedance of the signal source.

## 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-6229 Analog Output

**Purpose** National Instruments PCI-6229 Analog Output block

**Library** xPC Target Library for National Instruments

## Scaling of Output to Input

Hardware Output	Block Input Data Type	Scaling
Volts	Double	1

## Block Parameters

### Channel vector

Enter 1 to 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 all four of the analog output channels, enter

[1,2, 3, 4]

Number the channels begin 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 the initial values between the time the model is downloaded and the time it is started.

# National Instruments PCI-6229 Analog Output

---

## **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-6229 Digital Input

**Purpose** National Instruments PCI-6229 Digital Input block

**Library** xPC Target Library for National Instruments

**Note** Models can have multiple digital input blocks for one physical board. In this case, the xPC Target environment executes the blocks based on their sorted order as determined by the Simulink software.

Each board has one set of static digital I/O ports. You can independently configure each port to be input or output.

If a digital input block shares channels with a digital output block, the input block reads the last value written to the digital output block on the overlapping channels.

## Scaling of Input to Output

Hardware Input	Block Output Data Type	Scaling
TTL	Double	TTL low $\geq 0.0$ TTL high = 1.0

## Block Parameters

### Channel vector

Enter numbers between 1 and 32 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 the first eight 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.

### Sample time

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

# National Instruments PCI-6229 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
```

# National Instruments PCI-6229 Digital Output

**Purpose** National Instruments PCI-6229 Digital Output block

**Library** xPC Target Library for National Instruments

**Note** Models can have multiple digital output blocks for one physical board. In this case, the xPC Target environment executes the blocks based on their sorted order as determined by the Simulink software.

Each board has one set of digital I/O ports. You can independently set each port to be input or output.

If a digital input block shares channels with a digital output block, the input block reads the last value written to the digital output block on the overlapping channels.

The block compares the value written to each port with 0.5.

Value written	Block writes digital output with a value of...
$\geq 0.5$	1
$< 0.5$	0

## Scaling of Output to Input

Hardware Output	Block Input Data Type	Scaling
TTL	Double	$< 0.5$ = TTL low $\geq 0.5$ = TTL high

## Block Parameters

### Channel vector

Enter numbers between 1 and 32 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 the first eight digital outputs for, enter

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

# National Instruments PCI-6229 Digital Output

---

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-6229 Incremental Encoder

**Purpose** National Instruments PCI-6229 Incremental Encoder block

**Library** xPC Target Library for National Instruments

**Scaling Input to Output**

Hardware Input	Block Output Data Type	Scaling
TTL level quadrature encoder signals	Double	counts

## Block Parameters

### Channel

From the list, choose 0 or 1 for the channel counter you want to use. For information on input for these counters, see the National Instruments M Series user manual. The pin associations are:

Counter	Channel 0	Channel 1
A	PFI 8 (Pin 37)	PFI 3 (Pin 42)
B	PFI 10 (Pin 45)	PFI 11 (Pin 46)
Z	PFI 9 (Pin 3)	PFI 4 (Pin 41)

### Counting mode

From the list, select a counting mode:

Mode	Description
Quadrature Mode x1	Count on rising edges of the A signal
Quadrature Mode x2	Count on both rising and falling edges of the A signal
Quadrature Mode x4	Count on both rising and falling edges of both A and B signals

# National Instruments PCI-6229 Incremental Encoder

---

The level of the opposite signal when an edge occurs determines the count direction. For example, a rising edge of A when B is high counts one way, while an edge of A when B is low counts the other. For more information, see the National Instruments M Series user manual.

## Initial count

The initial count specifies the initial value for the counter. The block resets the count to this value under the following conditions:

- When the model starts
- When the following is true, which typically occurs once per revolution for a rotary encoder.
  - **Reload at index pulse** check box is selected
  - Index pulse (Z input) is true
  - The phases of A and B match the setting of **Index phase**

## Reload at index pulse

Select this check box to reset the counter to the value of Initial count at each index pulse. The reset occurs when:

- The encoder outputs a high level on its index output
- The phases of A and B match the setting of **Index phase**

## Index phase

If you have selected the **Reload at index pulse** check box, the **Index phase** parameter specifies the phase of the quadrature signals during which the count reloads with **Initial count**. Your choice of value primarily depends on the particular incremental encoder you use. From the list, select one:

- A low B low
- A low B high
- A high B low
- A high B high

# National Instruments PCI-6229 Incremental Encoder

---

To choose an **Index phase** for reset, use an oscilloscope, triggered on the index pulse, to observe the states of counters A and B. If your choice does not meet the encoder requirements, resets may be inconsistent.

## Filter

You can apply a digital debouncing filter to the input pins before processing.

From the list, select a filter value to ignore pulses that are shorter than the filter time. The value of `None` indicates that there is no filtering of the input.

- `None`
- `Minimum pulse width 125 nanoseconds`
- `Minimum pulse width 6.25 microseconds`
- `Minimum pulse width 1.25 milliseconds`

## Sample time

Model 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 locate the board automatically.

If two or more boards of this type are in the target PC, for the board associated with this driver block, enter:

- `Bus number`
- `PCI slot number`

Use the format `[BusNumber, SlotNumber]`. To determine the bus number and the PCI slot number, type

```
getxpcpci
```

# National Instruments PCI-6229 PFI Digital Input

**Purpose** National Instruments PCI-6229 PFI Digital Input block

**Library** xPC Target Library for National Instruments

**Description** A model can have more than one PFI digital input block for any given physical board. If a model has more than one digital input block, it executes the blocks in the order that the Simulink software has determined.

The two counters and the PFI digital input and output blocks share the 16 PFI digital I/O lines. Follow these rules when using these blocks:

<b>You can read a PFI bit with this digital input block when...</b>	<b>You cannot read a PFI bit with this digital input block when...</b>
PWM measure block is using the bit	PWM generate block is using the bit
Incremental encoder block is using the bit	PFI digital output block is using the bit

## Scaling of Input to Output

<b>Hardware Input</b>	<b>Block Output Data Type</b>	<b>Scaling</b>
TTL	Double	TTL low $\geq 0.0$ TTL high = 1.0

## Block Parameters

### PFI bit vector (0-15)

This block uses 0 based numbering to reflect the names of the bits described in the National Instruments M Series user manual.

Enter a vector that contains all bits you want to monitor. For example, to read bits PFI 0, PFI 5, and PFI 7, enter

[0, 5, 7]

## Sample time

Model 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 locate the board automatically.

If two or more boards of this type are in the target PC, for the board associated with this driver block, enter:

- Bus number
- PCI slot number

Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```

# National Instruments PCI-6229 PFI Digital Output

---

**Purpose** National Instruments PCI-6229 PFI Digital Output block

**Library** xPC Target Library for National Instruments

**Description** A model can have more than one PFI digital input block for any given physical board. If a model has more than one digital input block, it executes the blocks in the order that the Simulink software has determined.

The two counters and the PFI digital input and output blocks share the 16 PFI digital I/O lines. Follow these rules when using these blocks:

- Do not use a PFI bit as a digital output when the PDF digital input block is using the bit.
- Do not use a PFI bit as a digital output when the incremental encoder is using the bit.
- Do not use a PFI bit for digital output when the PWM generate block is using the bit.

## Scaling of Output to Input

Hardware Output	Block Input Data Type	Scaling
TTL	Double	< 0.5 = TTL low ≥ 0.5 = TTL high

## Block Parameters

### PFI bit vector (0-15)

This block uses 0 based numbering to reflect the names of the bits described in the National Instruments M Series user manual.

Enter a vector that contains all bits you want to control. For example, to write bits PFI 1, PFI 3, and PFI 9, enter

[1, 3, 9]

## Reset vector

The reset vector controls the behavior at model termination. If you specify a value of 1, the output is reset to the value specified in **Initial value vector**. Enter a scalar or a vector of values, one for each output. If you specify a scalar value, the driver uses that setting for all outputs. If you specify a value of 1, when the model execution stops, the corresponding output is reset to the value specified in **Initial value vector**. If you specify a value of 0, the output 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 block outputs. Enter a scalar or a vector of values, one for each output. If you enter a scalar, the driver uses that value for all outputs.

The block applies this value as follows:

- The model uses this value when you first download the model to the target PC.
- If you set **Reset vector** to 1, the block output is reset to this value when model execution stops. If the value is greater than 0.5, the block sets output to HIGH. If the value is less than or equal to 0.5, the block sets the output to LOW.

## Sample time

Model 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 locate the board automatically.

If two or more boards of this type are in the target PC, for the board associated with this driver block, enter:

# National Instruments PCI-6229 PFI Digital Output

---

- Bus number
- PCI slot number

Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```

# National Instruments PCI-6229 PWM Generate

**Purpose** National Instruments PCI-6229 PWM Generate block

**Library** xPC Target Library for National Instruments

**Description** Use this block to generate variable frequency and pulse width output. All M Series boards have two 32-bit counters that you can use to generate pulse trains. Inputs are in counts of the 80 MHz reference frequency on the onboard clock.

- H input is the number of counts that the output is high.
- L input is the number of counts that the output is low.

If the H input goes to 0, the output stays low. If the L input goes to 0, the output stays high. If both are 0, the output goes low.

## Block Parameters

### Counter

From the list, select the counter, 0 or 1, that you want to use. The output for these counters is on the following pins:

Counter	Pin
0	PFI 12 (Pin 2)
1	PFI 13 (Pin 40)

### Initial high count

Enter the initial H count to take effect between the time the Simulink engine calls the `md1Start` routine for this block and the first time it calls `md1Outputs`. This length of time depends on the number of blocks in the model and the sorted order as determined by the Simulink software.

If you enter a value of 0, the output stays low until the first time the Simulink engine calls `md1Outputs`.

# National Instruments PCI-6229 PWM Generate

---

## Initial low count

Enter the initial L count to take effect between the time the Simulink engine calls the `md1Start` routine for this block and the first time it calls `md1Outputs`. This length of time depends on the number of blocks in the model and the sorted order as determined by the Simulink software. If you enter a value of 0, the output stays high until the first time the Simulink engine calls `md1Outputs`, unless you have also set H to 0.

## Arm input

Select this check box to add a third input port to the block.

- If the input connected to this port is greater than 0.5, the block arms the counter. Output then occurs.
- If the input connected to this port is less than or equal to 0.5, the block disarms the counter and the output goes to the level of the value in the **Disarm level** parameter.

## Disarm level

From the list, select the disarm level `Disarm to LOW` or `Disarm to HIGH`. If the model disarms the block and the level of the value is **Arm input**, the output goes to the level you select here.

## Stop level

From the list, select the stop level `Stop to LOW` or `Stop to HIGH`. When model execution stops, the output goes to the level you choose here.

## Sample time

Model 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 locate the board automatically.

If two or more boards of this type are in the target PC, for the board associated with this driver block, enter:

# National Instruments PCI-6229 PWM Generate

---

- Bus number
- PCI slot number

Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```

# National Instruments PCI-6229 PWM Measure

---

**Purpose** National Instruments PCI-6229 PWM Measure block

**Library** xPC Target Library for National Instruments

**Description** Use this block to measure the pulse width or the period of a signal.  
To measure the pulse width of a signal, connect the signal to the gate, as follows:

Counter	Gate Pin
0	PFI 9 (Pin 3)
1	PFI 4 (Pin 41)

The input waveform must have fast rise and fall times and conform to TTL signal levels of [0,5] volts. Use external signal conditioning to achieve these levels. If you input a sine wave, the decision points are less definite and the measurement is noisy. In extreme cases, the gate can transition multiple times during a slow signal transition.

This block does not queue or buffer measurements. If the gate changes state, the driver holds the current count, overwriting any previous value.

This block returns a value that is always the length of the last gate period before the block executes.

To compute the duty cycle percentage, use two counters. Because the period of a cycle is H counter + L counter, you can measure any two of these counters and compute the third.

You cannot measure a 0 width pulse.

## Block Parameters

### Counter

From the list, select a counter, 0 or 1.

### Trigger mode

From the list, select the gating mode to use:

# National Instruments PCI-6229 PWM Measure

Mode	Description
Width of Gate High	Returns the number of 80 MHz cycles that occur when a single cycle of the input is high.
Width of Gate Low	Returns the number of 80 MHz cycles that occur when a single cycle of the input is low.
Between Rising Edges	Returns the number of 80 MHz cycles that occur between rising edges of the input signal. This cycle is the full period of the input signal.
Between Falling Edges	Returns the number of 80 MHz cycles that occur between falling edges of the input signal. This cycle is the full period of the input signal.

If the input signal has both fast rise and fall edges, **Between Rising Edges** and **Between Falling Edges** return the same value. If the input is asymmetric, one of these options can give a better result than the other.

## Filter

You can apply a digital debouncing filter to the input pins before processing.

From the list, select a filter value to ignore pulses that are shorter than the filter time. The block ignores pulses shorter than the chosen duration. The value of **None** indicates that there is no filtering of the input

- None
- Minimum pulse width 125 nanoseconds
- Minimum pulse width 6.25 microseconds
- Minimum pulse width 1.25 milliseconds

# National Instruments PCI-6229 PWM Measure

---

## Sample time

Model 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 locate the board automatically.

If two or more boards of this type are in the target PC, for the board associated with this driver block, enter:

- Bus number
- PCI slot number

Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```

**Purpose** Support for National Instruments PCI-6251 board

**Board** National Instruments PCI-6251

**General Description** The PCI-6251 is a high-speed data acquisition I/O board with up to 16 single-ended or 8 differential 16-bit analog inputs, two 16-bit analog outputs, 24 digital I/O lines, and two 32-bit counters.

The xPC Target block library supports this board with these driver blocks:

- National Instruments PCI-6251 Analog Input
- National Instruments PCI-6251 Analog Output
- National Instruments PCI-6251 Digital Input
- National Instruments PCI-6251 Digital Output
- National Instruments PCI-6251 Incremental Encoder
- National Instruments PCI-6251 PFI Digital Input
- National Instruments PCI-6251 PFI Digital Output
- National Instruments PCI-6251 PWM Generate
- National Instruments PCI-6251 PWM Measure

<b>Board Characteristics</b>	Board name	PCI-6251
	Manufacturer	National Instruments
	Bus type	PCI, PXI (use PCI version of blocks)
	Multiple block instance support	A/D: No, D/A: No, Digital I/O: Yes
	Multiple board support	Yes

# National Instruments PCI-6251 Analog Input

**Purpose** National Instruments PCI-6251 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 lines beginning with 1 even though the board manufacturer starts numbering the lines with 0.

### Range vector

Enter a range code for each of the channels in the channel vector. You can specify one value to be used for all channels, or specify one for each channel. This driver allows each channel to be different. This board works only in bipolar mode.

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
-.5 to +.5	.5

# National Instruments PCI-6251 Analog Input

Input Range (V)	Range Code
-0.2 to +0.2	0.2
-0.1 to +0.1	0.1

For example, if the first channel is -10 to +10 volts and the second and fifth channels are -1 to +1 volts, enter

[10, 1, 1]

If all channels use the same range, you specify one entry.

## Input coupling vector

Enter a coupling code for each of the channels in the channel vector. You can specify one value for all channels, or specify one for each channel. 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 National Instruments user 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 National Instruments user manual.
DIFF	2	First analog input line connected to the positive input of the PGIA. Second analog input line connected to the

# National Instruments PCI-6251 Analog Input

Coupling	Coupling Code	Description
		negative input of the PGIA. See the National Instruments user 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 boards use a hardware number scheme for differential mode channel pairs. xPC Target drivers use a software number scheme that adds a 1 to the hardware number, as shown in the following table:

Hardware Channel Number		Software Channel Number = Hardware Channel Number + 1	
+	-	+	-
0	8	1	9
1	9	2	10
2	10	3	11
3	11	4	12
4	12	5	13
5	13	6	14
6	14	7	15
7	15	8	16

The hardware uses a second input 8 channels higher than the first channel as the negative input of the pair. In the example above, the hardware would use the thirteenth channel as a differential input with the fifth channel.

If the board is set up for differential mode, you can read the data from either of the channels in the differential pair. For example, if you have a differential pair of 1 and 9, you can read the data from channel 1 or channel 9. However, you might want to read the lower channel number of the pair because it remains unchanged when you switch the input mode between single-ended and differential.

## Scan interval

If **Channel vector** has more than one channel in it, enter the time between sampling different channels. Because execution waits for data to be available, this value increases the time to execute the block. If you enter a shorter time, you get the minimum 1e-6. If you enter a time smaller than this value, the time will still be 4e-6, which is the fastest this board can acquire data. See the National Instruments user manual for a discussion about settling time and the need to control the time between sampling different channels depending on the impedance of the signal source.

## 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-6251 Analog Output

**Purpose** National Instruments PCI-6251 Analog Output block

**Library** xPC Target Library for National Instruments

<b>Scaling of Output to Input</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 though 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
$V_{ref} \times sig$ , where $-1 \leq sig \leq +1$	APFIO

# National Instruments PCI-6251 Analog Output

---

For example, if the first channel is -10 to +10 volts and the second channel is -5 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 or a multiple of the base sample 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-6251 Digital Input

---

**Purpose** National Instruments PCI-6251 Digital Input block

**Library** xPC Target Library for National Instruments

**Note** Models can have multiple digital input blocks for one physical board. In this case, the xPC Target environment executes the blocks based on their sorted order as determined by the Simulink software.

Each board has one set of static digital I/O ports. You can independently configure each port to be input or output.

If a digital input block shares channels with a digital output block, the input block reads the last value written to the digital output block on the overlapping channels.

## Scaling of Input to Output

Hardware Input	Block Output Data Type	Scaling
TTL	Double	TTL low $\geq 0.0$ 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 the first eight 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.

### 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
```

# National Instruments PCI-6251 Digital Output

**Purpose** National Instruments PCI-6251 Digital Output block

**Library** xPC Target Library for National Instruments

**Note** Models can have multiple digital output blocks for one physical board. In this case, the xPC Target environment executes the blocks based on their sorted order as determined by the Simulink software.

Each board has one set of digital I/O ports. You can independently each port to be input or output.

If a digital input block shares channels with a digital output block, the input block reads the last value written to the digital output block on the overlapping channels.

The block compares the value written to each port with 0.5.

Value written	Block writes digital output with a value of...
$\geq 0.5$	1
$< 0.5$	0

## Scaling of Output to Input

Hardware Output	Block Input Data Type	Scaling
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 the first eight 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.

## **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-6251 Incremental Encoder

**Purpose** National Instruments PCI-6251 Incremental Encoder block

**Library** xPC Target Library for National Instruments

**Scaling Input to Output**

Hardware Input	Block Output Data Type	Scaling
TTL level quadrature encoder signals	Double	counts

## Block Parameters

### Channel

From the list, choose 0 or 1 for the channel counter you want to use. For information on input for these counters, see the National Instruments M Series user manual. The pin associations are:

Counter	Channel 0	Channel 1
A	PFI 8 (Pin 37)	PFI 3 (Pin 42)
B	PFI 10 (Pin 45)	PFI 11 (Pin 46)
Z	PFI 9 (Pin 3)	PFI 4 (Pin 41)

### Counting mode

From the list, select a counting mode:

Mode	Description
Quadrature Mode x1	Count on rising edges of the A signal
Quadrature Mode x2	Count on both rising and falling edges of the A signal
Quadrature Mode x4	Count on both rising and falling edges of both A and B signals

# National Instruments PCI-6251 Incremental Encoder

---

The level of the opposite signal when an edge occurs determines the count direction. For example, a rising edge of A when B is high counts one way, while an edge of A when B is low counts the other. For more information, see the National Instruments M Series user manual.

## Initial count

The initial count specifies the initial value for the counter. The block resets the count to this value under the following conditions:

- When the model starts
- When the following is true, which typically occurs once per revolution for a rotary encoder.
  - **Reload at index pulse** check box is selected
  - Index pulse (Z input) is true
  - The phases of A and B match the setting of **Index phase**

## Reload at index pulse

Select this check box to reset the counter to the value of **Initial count** at each index pulse. The reset occurs when:

- The encoder outputs a high level on its index output
- The phases of A and B match the setting of **Index phase**

## Index phase

If you have selected the **Reload at index pulse** check box, the **Index phase** parameter specifies the phase of the quadrature signals during which the count reloads with **Initial count**. Your choice of value primarily depends on the particular incremental encoder you use. From the list, select one:

- A low B low
- A low B high
- A high B low
- A high B high

# National Instruments PCI-6251 Incremental Encoder

---

To choose an **Index phase** for reset, use an oscilloscope, triggered on the index pulse, to observe the states of counters A and B. If your choice does not meet the encoder requirements, resets may be inconsistent.

## Filter

You can apply a digital debouncing filter to the input pins before processing.

From the list, select a filter value to ignore pulses that are shorter than the filter time. The value of `None` indicates that there is no filtering of the input.

- `None`
- `Minimum pulse width 125 nanoseconds`
- `Minimum pulse width 6.25 microseconds`
- `Minimum pulse width 1.25 milliseconds`

## Sample time

Model 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 locate the board automatically.

If two or more boards of this type are in the target PC, for the board associated with this driver block, enter:

- `Bus number`
- `PCI slot number`

Use the format `[BusNumber, SlotNumber]`. To determine the bus number and the PCI slot number, type

```
getxpcpci
```

# National Instruments PCI-6251 PFI Digital Input

**Purpose** National Instruments PCI-6251 PFI Digital Input block

**Library** xPC Target Library for National Instruments

**Description** A model can have more than one PFI digital input block for any given physical board. If a model has more than one digital input block, it executes the blocks in the order that the Simulink software has determined.

The two counters and the PFI digital input and output blocks share the 16 PFI digital I/O lines. Follow these rules when using these blocks:

<b>You can read a PFI bit with this digital input block when...</b>	<b>You cannot read a PFI bit with this digital input block when...</b>
PWM measure block is using the bit	PWM generate block is using the bit
Incremental encoder block is using the bit	PFI digital output block is using the bit

## Scaling of Input to Output

<b>Hardware Input</b>	<b>Block Output Data Type</b>	<b>Scaling</b>
TTL	Double	TTL low $\geq 0.0$ TTL high = 1.0

## Block Parameters

### PFI bit vector (0-15)

This block uses 0 based numbering to reflect the names of the bits described in the National Instruments M Series user manual.

Enter a vector that contains all bits you want to monitor. For example, to read bits PFI 0, PFI 5, and PFI 7, enter

[0, 5, 7]

# National Instruments PCI-6251 PFI Digital Input

---

## Sample time

Model 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 locate the board automatically.

If two or more boards of this type are in the target PC, for the board associated with this driver block, enter:

- Bus number
- PCI slot number

Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```

# National Instruments PCI-6251 PFI Digital Output

**Purpose** National Instruments PCI-6251 PFI Digital Output block

**Library** xPC Target Library for National Instruments

**Description** A model can have more than one PFI digital input block for any given physical board. If a model has more than one digital input block, it executes the blocks in the order that the Simulink software has determined.

The two counters and the PFI digital input and output blocks share the 16 PFI digital I/O lines. Follow these rules when using these blocks:

- Do not use a PFI bit as a digital output when the PDF digital input block is using the bit.
- Do not use a PFI bit as a digital output when the incremental encoder is using the bit.
- Do not use a PFI bit for digital output when the PWM generate block is using the bit.

## Scaling of Output to Input

Hardware Output	Block Input Data Type	Scaling
TTL	Double	< 0.5 = TTL low ≥ 0.5 = TTL high

## Block Parameters

### PFI bit vector (0-15)

This block uses 0 based numbering to reflect the names of the bits described in the National Instruments M Series user manual.

Enter a vector that contains all bits you want to control. For example, to write bits PFI 1, PFI 3, and PFI 9, enter

[1, 3, 9]

# National Instruments PCI-6251 PFI Digital Output

---

## **Reset vector**

The reset vector controls the behavior at model termination. If you specify a value of 1, the output is reset to the value specified in **Initial value vector**. Enter a scalar or a vector of values, one for each output. If you specify a scalar value, the driver uses that setting for all outputs. If you specify a value of 1, when the model execution stops, the corresponding output is reset to the value specified in **Initial value vector**. If you specify a value of 0, the output 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 block outputs. Enter a scalar or a vector of values, one for each output. If you enter a scalar, the driver uses that value for all outputs.

The block applies this value as follows:

- The model uses this value when you first download the model to the target PC.
- If you set **Reset vector** to 1, the block output is reset to this value when model execution stops. If the value is greater than 0.5, the block sets output to HIGH. If the value is less than or equal to 0.5, the block sets the output to LOW.

## **Sample time**

Model 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 locate the board automatically.

If two or more boards of this type are in the target PC, for the board associated with this driver block, enter:

# National Instruments PCI-6251 PFI Digital Output

---

- Bus number
- PCI slot number

Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```

# National Instruments PCI-6251 PWM Generate

**Purpose** National Instruments PCI-6251 PWM Generate block

**Library** xPC Target Library for National Instruments

**Description** Use this block to generate variable frequency and pulse width output. All M Series boards have two 32-bit counters that you can use to generate pulse trains. Inputs are in counts of the 80 MHz reference frequency on the onboard clock.

- H input is the number of counts that the output is high.
- L input is the number of counts that the output is low.

If the H input goes to 0, the output stays low. If the L input goes to 0, the output stays high. If both are 0, the output goes low.

## Block Parameters

### Counter

From the list, select the counter, 0 or 1, that you want to use. The output for these counters is on the following pins:

Counter	Pin
0	PFI 12 (Pin 2)
1	PFI 13 (Pin 40)

### Initial high count

Enter the initial H count to take effect between the time the Simulink engine calls the `md1Start` routine for this block and the first time it calls `md1Outputs`. This length of time depends on the number of blocks in the model and the sorted order as determined by the Simulink software.

If you enter a value of 0, the output stays low until the first time the Simulink engine calls `md1Outputs`.

## Initial low count

Enter the initial L count to take effect between the time the Simulink engine calls the `mdlStart` routine for this block and the first time it calls `mdlOutputs`. This length of time depends on the number of blocks in the model and the sorted order as determined by the Simulink software. If you enter a value of 0, the output stays high until the first time the Simulink engine calls `mdlOutputs`, unless you have also set H to 0.

## Arm input

Select this check box to add a third input port to the block.

- If the input connected to this port is greater than 0.5, the block arms the counter. Output then occurs.
- If the input connected to this port is less than or equal to 0.5, the block disarms the counter and the output goes to the level of the value in the **Disarm level** parameter.

## Disarm level

From the list, select the disarm level **Disarm to LOW** or **Disarm to HIGH**. If the model disarms the block and the level of the value is **Arm input**, the output goes to the level you select here.

## Stop level

From the list, select the stop level **Stop to LOW** or **Stop to HIGH**. When model execution stops, the output goes to the level you choose here.

## Sample time

Model 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 locate the board automatically.

If two or more boards of this type are in the target PC, for the board associated with this driver block, enter:

# National Instruments PCI-6251 PWM Generate

---

- Bus number
- PCI slot number

Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```

# National Instruments PCI-6251 PWM Measure

**Purpose** National Instruments PCI-6251 PWM Measure block

**Library** xPC Target Library for National Instruments

**Description** Use this block to measure the pulse width or the period of a signal.  
To measure the pulse width of a signal, connect the signal to the gate, as follows:

Counter	Gate Pin
0	PFI 9 (Pin 3)
1	PFI 4 (Pin 41)

The input waveform must have fast rise and fall times and conform to TTL signal levels of [0,5] volts. Use external signal conditioning to achieve these levels. If you input a sine wave, the decision points are less definite and the measurement is noisy. In extreme cases, the gate can transition multiple times during a slow signal transition.

This block does not queue or buffer measurements. If the gate changes state, the driver holds the current count, overwriting any previous value.

This block returns a value that is always the length of the last gate period before the block executes.

To compute the duty cycle percentage, use two counters. Because the period of a cycle is H counter + L counter, you can measure any two of these counters and compute the third.

You cannot measure a 0 width pulse.

## Block Parameters

### Counter

From the list, select a counter, 0 or 1.

### Trigger mode

From the list, select the gating mode to use:

# National Instruments PCI-6251 PWM Measure

Mode	Description
Width of Gate High	Returns the number of 80 MHz cycles that occur when a single cycle of the input is high.
Width of Gate Low	Returns the number of 80 MHz cycles that occur when a single cycle of the input is low.
Between Rising Edges	Returns the number of 80 MHz cycles that occur between rising edges of the input signal. This cycle is the full period of the input signal.
Between Falling Edges	Returns the number of 80 MHz cycles that occur between falling edges of the input signal. This cycle is the full period of the input signal.

If the input signal has both fast rise and fall edges, **Between Rising Edges** and **Between Falling Edges** return the same value. If the input is asymmetric, one of these options can give a better result than the other.

## Filter

You can apply a digital debouncing filter to the input pins before processing.

From the list, select a filter value to ignore pulses that are shorter than the filter time. The block ignores pulses shorter than the chosen duration. The value of **None** indicates that there is no filtering of the input

- None
- Minimum pulse width 125 nanoseconds
- Minimum pulse width 6.25 microseconds
- Minimum pulse width 1.25 milliseconds

## Sample time

Model 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 locate the board automatically.

If two or more boards of this type are in the target PC, for the board associated with this driver block, enter:

- Bus number
- PCI slot number

Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```

# National Instruments PCI-6259

---

**Purpose** Support for National Instruments PCI-6259 board

**Board** National Instruments PCI-6259

**General Description** The PCI-6259 is a data acquisition I/O board with up to 32 single-ended or 16 differential 16-bit analog inputs, four 16-bit analog outputs, 48 digital I/O lines, and two 32-bit counters.

The xPC Target block library supports this board with these driver blocks:

- National Instruments PCI-6259 Analog Input
- National Instruments PCI-6259 Analog Output
- National Instruments PCI-6259 Digital Input
- National Instruments PCI-6259 Digital Output
- National Instruments PCI-6259 Incremental Encoder
- National Instruments PCI-6259 PFI Digital Input
- National Instruments PCI-6259 PFI Digital Output
- National Instruments PCI-6259 PWM Generate
- National Instruments PCI-6259 PWM Measure

## Board Characteristics

Board name	PCI-6259
Manufacturer	National Instruments
Bus type	PCI, PXI (use PCI version of blocks)
Multiple block instance support	A/D: No, D/A: No, Digital I/O: Yes
Multiple board support	Yes

# National Instruments PCI-6259 Analog Input

**Purpose** National Instruments PCI-6259 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 32. 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 lines beginning with 1 even though the board manufacturer starts numbering the lines with 0.

### Range vector

Enter a range code for each of the channels in the channel vector. You can specify one value to be used for all channels, or specify one for each channel. This driver allows each channel to be different. This board works only in bipolar mode.

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-6259 Analog Input

Input Range (V)	Range Code
-0.2 to +0.2	0.2
-0.1 to +0.1	0.1

For example, if the first channel is -10 to +10 volts and the second and fifth channels are -1 to +1 volts, enter

[10, 1, 1]

If all channels use the same range, you specify one entry.

## Input coupling vector

Enter a coupling code for each of the channels in the channel vector. You can specify one value for all channels, or specify one for each channel. 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 National Instruments user 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 National Instruments user manual.
DIFF	2	First analog input line connected to the positive input of the PGIA. Second analog input line connected to the

# National Instruments PCI-6259 Analog Input

Coupling	Coupling Code	Description
		negative input of the PGIA. See the National Instruments user 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 boards use a hardware number scheme for differential mode channel pairs. xPC Target drivers use a software number scheme that adds a 1 to the hardware number, as shown in the following table:

Hardware Channel Number		Software Channel Number = Hardware Channel Number + 1	
+	-	+	-
0	8	1	9
1	9	2	10
2	10	3	11
3	11	4	12
4	12	5	13
5	13	6	14
6	14	7	15
7	15	8	16
16	24	17	25
17	25	18	26
18	26	19	27
19	27	20	28

# National Instruments PCI-6259 Analog Input

Hardware Channel Number		Software Channel Number = Hardware Channel Number + 1	
20	28	21	29
21	29	22	30
22	30	23	31
23	31	24	32

The hardware uses a second input 8 channels higher than the first channel as the negative input of the pair. In the example above, the hardware would use the thirteenth channel as a differential input with the fifth channel.

If the board is set up for differential mode, you can read the data from either of the channels in the differential pair. For example, if you have a differential pair of 1 and 9, you can read the data from channel 1 or channel 9. However, you might want to read the lower channel number of the pair because it remains unchanged when you switch the input mode between single-ended and differential.

## Scan interval

If **Channel vector** has more than one channel in it, enter the time between sampling different channels. Because execution waits for data to be available, this value increases the time to execute the block. If you enter a shorter time, you get the minimum 1e-6. If you enter a time smaller than this value, the time will still be 4e-6, which is the fastest this board can acquire data. See the National Instruments user manual for a discussion about settling time and the need to control the time between sampling different channels depending on the impedance of the signal source.

## 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-6259 Analog Output

**Purpose** National Instruments PCI-6259 Analog Output block

**Library** xPC Target Library for National Instruments

## Scaling of Output to Input

Hardware Output	Block Input Data Type	Scaling
Volts	Double	1

## Block Parameters

### Channel vector

Enter 1 to 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 all of the analog output channels, enter

[1,2,3,4]

Number the channels begin with 1 even though 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. This board works only in bipolar mode.

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

# National Instruments PCI-6259 Analog Output

Input Range (V)	Range Code
$V_{ref} \times sig$ , where $-1 \leq sig \leq +1$	APFI0
$V_{ref} \times sig$ , where $-1 \leq sig \leq +1$	APFI1

For example, if the first channel is -10 to +10 volts and the second channel -5 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 or a multiple of the base sample 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-6259 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
```

# National Instruments PCI-6259 Digital Input

**Purpose** National Instruments PCI-6259 Digital Input block

**Library** xPC Target Library for National Instruments

**Note** Models can have multiple digital input blocks for one physical board. In this case, the xPC Target environment executes the blocks based on their sorted order as determined by the Simulink software.

Each board has one set of static digital I/O ports. You can independently configure each port to be input or output.

If a digital input block shares channels with a digital output block, the input block reads the last value written to the digital output block on the overlapping channels.

## Scaling of Input to Output

Hardware Input	Block Output Data Type	Scaling
TTL	Double	TTL low $\geq$ 0.0 TTL high = 1.0

## Block Parameters

### Channel vector

Enter numbers between 1 and 32 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 the first eight 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.

### 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

# National Instruments PCI-6259 Digital Input

---

- 1

to automatically locate the board.

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of 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-6259 Digital Output

**Purpose** National Instruments PCI-6259 Digital Output block

**Library** xPC Target Library for National Instruments

**Note** Models can have multiple digital output blocks for one physical board. In this case, the xPC Target environment executes the blocks based on their sorted order as determined by the Simulink software.

Each board has one set of digital I/O ports. You can independently configure each port to be input or output.

If a digital input block shares channels with a digital output block, the input block reads the last value written to the digital output block on the overlapping channels.

The block compares the value written to each port with 0.5.

Value written	Block writes digital output with a value of...
$\geq 0.5$	1
$< 0.5$	0

## Scaling of Output to Input

Hardware Output	Block Input Data Type	Scaling
TTL	Double	$< 0.5$ = TTL low $\geq 0.5$ = TTL high

## Block Parameters

### Channel vector

Enter numbers between 1 and 32 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 the first eight digital outputs, enter

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

# National Instruments PCI-6259 Digital Output

---

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-6259 Incremental Encoder

**Purpose** National Instruments PCI-6259 Incremental Encoder 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 level quadrature encoder signals	Double	counts

## Block Parameters

### Channel

From the list, choose 0 or 1 for the channel counter you want to use. For information on input for these counters, see the National Instruments M Series user manual. The pin associations are:

Counter	Channel 0	Channel 1
A	PFI 8 (Pin 37)	PFI 3 (Pin 42)
B	PFI 10 (Pin 45)	PFI 11 (Pin 46)
Z	PFI 9 (Pin 3)	PFI 4 (Pin 41)

### Counting mode

From the list, select a counting mode:

Mode	Description
Quadrature Mode x1	Count on rising edges of the A signal
Quadrature Mode x2	Count on both rising and falling edges of the A signal
Quadrature Mode x4	Count on both rising and falling edges of both A and B signals

# National Instruments PCI-6259 Incremental Encoder

---

The level of the opposite signal when an edge occurs determines the count direction. For example, a rising edge of A when B is high counts one way, while an edge of A when B is low counts the other. For more information, see the National Instruments M Series user manual.

## Initial count

The initial count specifies the initial value for the counter. The block resets the count to this value under the following conditions:

- When the model starts
- When the following is true, which typically occurs once per revolution for a rotary encoder.
  - **Reload at index pulse** check box is selected
  - Index pulse (Z input) is true
  - The phases of A and B match the setting of **Index phase**

## Reload at index pulse

Select this check box to reset the counter to the value of **Initial count** at each index pulse. The reset occurs when:

- The encoder outputs a high level on its index output
- The phases of A and B match the setting of **Index phase**

## Index phase

If you have selected the **Reload at index pulse** check box, the **Index phase** parameter specifies the phase of the quadrature signals during which the count reloads with **Initial count**. Your choice of value primarily depends on the particular incremental encoder you use. From the list, select one:

- A low B low
- A low B high
- A high B low
- A high B high

# National Instruments PCI-6259 Incremental Encoder

---

To choose an **Index phase** for reset, use an oscilloscope, triggered on the index pulse, to observe the states of counters A and B. If your choice does not meet the encoder requirements, resets may be inconsistent.

## Filter

You can apply a digital debouncing filter to the input pins before processing.

From the list, select a filter value to ignore pulses that are shorter than the filter time. The value of **None** indicates that there is no filtering of the input.

- None
- Minimum pulse width 125 nanoseconds
- Minimum pulse width 6.25 microseconds
- Minimum pulse width 1.25 milliseconds

## Sample time

Model 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 locate the board automatically.

If two or more boards of this type are in the target PC, for the board associated with this driver block, enter:

- Bus number
- PCI slot number

Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```

# National Instruments PCI-6259 PFI Digital Input

**Purpose** National Instruments PCI-6259 PFI Digital Input block

**Library** xPC Target Library for National Instruments

**Description** A model can have more than one PFI digital input block for any given physical board. If a model has more than one digital input block, it executes the blocks in the order that the Simulink software has determined.

The two counters and the PFI digital input and output blocks share the 16 PFI digital I/O lines. Follow these rules when using these blocks:

You can read a PFI bit with this digital input block when...	You cannot read a PFI bit with this digital input block when...
PWM measure block is using the bit	PWM generate block is using the bit
Incremental encoder block is using the bit	PFI digital output block is using the bit

## Scaling of Input to Output

Hardware Input	Block Output Data Type	Scaling
TTL	Double	TTL low $\geq 0.0$ TTL high = 1.0

## Block Parameters

### PFI bit vector (0-15)

This block uses 0 based numbering to reflect the names of the bits described in the National Instruments M Series user manual.

Enter a vector that contains all bits you want to monitor. For example, to read bits PFI 0, PFI 5, and PFI 7, enter

[0, 5, 7]

## Sample time

Model 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 locate the board automatically.

If two or more boards of this type are in the target PC, for the board associated with this driver block, enter:

- Bus number
- PCI slot number

Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```

# National Instruments PCI-6259 PFI Digital Output

---

**Purpose** National Instruments PCI-6259 PFI Digital Output block

**Library** xPC Target Library for National Instruments

**Description** A model can have more than one PFI digital input block for any given physical board. If a model has more than one digital input block, it executes the blocks in the order that the Simulink software has determined.

The two counters and the PFI digital input and output blocks share the 16 PFI digital I/O lines. Follow these rules when using these blocks:

- Do not use a PFI bit as a digital output when the PDF digital input block is using the bit.
- Do not use a PFI bit as a digital output when the incremental encoder is using the bit.
- Do not use a PFI bit for digital output when the PWM generate block is using the bit.

## Scaling of Output to Input

Hardware Output	Block Input Data Type	Scaling
TTL	Double	$< 0.5 = \text{TTL low}$ $\geq 0.5 = \text{TTL high}$

## Block Parameters

### PFI bit vector (0-15)

This block uses 0 based numbering to reflect the names of the bits described in the National Instruments M Series user manual.

Enter a vector that contains all bits you want to control. For example, to write bits PFI 1, PFI 3, and PFI 9, enter

[1, 3, 9]

## Reset vector

The reset vector controls the behavior at model termination. If you specify a value of 1, the output is reset to the value specified in **Initial value vector**. Enter a scalar or a vector of values, one for each output. If you specify a scalar value, the driver uses that setting for all outputs. If you specify a value of 1, when the model execution stops, the corresponding output is reset to the value specified in **Initial value vector**. If you specify a value of 0, the output 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 block outputs. Enter a scalar or a vector of values, one for each output. If you enter a scalar, the driver uses that value for all outputs.

The block applies this value as follows:

- The model uses this value when you first download the model to the target PC.
- If you set **Reset vector** to 1, the block output is reset to this value when model execution stops. If the value is greater than 0.5, the block sets output to HIGH. If the value is less than or equal to 0.5, the block sets the output to LOW.

## Sample time

Model 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 locate the board automatically.

If two or more boards of this type are in the target PC, for the board associated with this driver block, enter:

# National Instruments PCI-6259 PFI Digital Output

---

- Bus number
- PCI slot number

Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```

# National Instruments PCI-6259 PWM Generate

**Purpose** National Instruments PCI-6259 PWM Generate block

**Library** xPC Target Library for National Instruments

**Description** Use this block to generate variable frequency and pulse width output. All M Series boards have two 32-bit counters that you can use to generate pulse trains. Inputs are in counts of the 80 MHz reference frequency on the onboard clock.

- H input is the number of counts that the output is high.
- L input is the number of counts that the output is low.

If the H input goes to 0, the output stays low. If the L input goes to 0, the output stays high. If both are 0, the output goes low.

## Block Parameters

### Counter

From the list, select the counter, 0 or 1, that you want to use. The output for these counters is on the following pins:

Counter	Pin
0	PFI 12 (Pin 2)
1	PFI 13 (Pin 40)

### Initial high count

Enter the initial H count to take effect between the time the Simulink engine calls the `md1Start` routine for this block and the first time it calls `md1Outputs`. This length of time depends on the number of blocks in the model and the sorted order as determined by the Simulink software.

If you enter a value of 0, the output stays low until the first time the Simulink engine calls `md1Outputs`.

# National Instruments PCI-6259 PWM Generate

---

## Initial low count

Enter the initial L count to take effect between the time the Simulink engine calls the `md1Start` routine for this block and the first time it calls `md1Outputs`. This length of time depends on the number of blocks in the model and the sorted order as determined by the Simulink software. If you enter a value of 0, the output stays high until the first time the Simulink engine calls `md1Outputs`, unless you have also set H to 0.

## Arm input

Select this check box to add a third input port to the block.

- If the input connected to this port is greater than 0.5, the block arms the counter. Output then occurs.
- If the input connected to this port is less than or equal to 0.5, the block disarms the counter and the output goes to the level of the value in the **Disarm level** parameter.

## Disarm level

From the list, select the disarm level **Disarm to LOW** or **Disarm to HIGH**. If the model disarms the block and the level of the value is **Arm input**, the output goes to the level you select here.

## Stop level

From the list, select the stop level **Stop to LOW** or **Stop to HIGH**. When model execution stops, the output goes to the level you choose here.

## Sample time

Model 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 locate the board automatically.

If two or more boards of this type are in the target PC, for the board associated with this driver block, enter:

# National Instruments PCI-6259 PWM Generate

---

- Bus number
- PCI slot number

Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```

# National Instruments PCI-6259 PWM Measure

---

**Purpose** National Instruments PCI-6259 PWM Measure block

**Library** xPC Target Library for National Instruments

**Description** Use this block to measure the pulse width or the period of a signal.  
To measure the pulse width of a signal, connect the signal to the gate, as follows:

Counter	Gate Pin
0	PFI 9 (Pin 3)
1	PFI 4 (Pin 41)

The input waveform must have fast rise and fall times and conform to TTL signal levels of [0,5] volts. Use external signal conditioning to achieve these levels. If you input a sine wave, the decision points are less definite and the measurement is noisy. In extreme cases, the gate can transition multiple times during a slow signal transition.

This block does not queue or buffer measurements. If the gate changes state, the driver holds the current count, overwriting any previous value.

This block returns a value that is always the length of the last gate period before the block executes.

To compute the duty cycle percentage, use two counters. Because the period of a cycle is H counter + L counter, you can measure any two of these counters and compute the third.

You cannot measure a 0 width pulse.

## Block Parameters

### Counter

From the list, select a counter, 0 or 1.

### Trigger mode

From the list, select the gating mode to use:

Mode	Description
Width of Gate High	Returns the number of 80 MHz cycles that occur when a single cycle of the input is high.
Width of Gate Low	Returns the number of 80 MHz cycles that occur when a single cycle of the input is low.
Between Rising Edges	Returns the number of 80 MHz cycles that occur between rising edges of the input signal. This cycle is the full period of the input signal.
Between Falling Edges	Returns the number of 80 MHz cycles that occur between falling edges of the input signal. This cycle is the full period of the input signal.

If the input signal has both fast rise and fall edges, **Between Rising Edges** and **Between Falling Edges** return the same value. If the input is asymmetric, one of these options can give a better result than the other.

## Filter

You can apply a digital debouncing filter to the input pins before processing.

From the list, select a filter value to ignore pulses that are shorter than the filter time. The block ignores pulses shorter than the chosen duration. The value of **None** indicates that there is no filtering of the input

- None
- Minimum pulse width 125 nanoseconds
- Minimum pulse width 6.25 microseconds
- Minimum pulse width 1.25 milliseconds

# National Instruments PCI-6259 PWM Measure

---

## Sample time

Model 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 locate the board automatically.

If two or more boards of this type are in the target PC, for the board associated with this driver block, enter:

- Bus number
- PCI slot number

Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```

**Purpose** Support for National Instruments PCI-6280 board

**Board** National Instruments PCI-6280

**General Description** The PCI-6280 is a data acquisition I/O board with up to 16 single-ended or 8 differential 18-bit analog inputs, 24 digital I/O lines, and two 32-bit 80 MHz counters.

The xPC Target block library supports this board with these driver blocks:

- National Instruments PCI-6280 Analog Input
- National Instruments PCI-6280 Digital Input
- National Instruments PCI-6280 Digital Output
- National Instruments PCI-6280 Incremental Encoder
- National Instruments PCI-6280 PFI Digital Input
- National Instruments PCI-6280 PFI Digital Output
- National Instruments PCI-6280 PWM Generate
- National Instruments PCI-6280 PWM Measure

<b>Board Characteristics</b>	Board name	PCI-6280
	Manufacturer	National Instruments
	Bus type	PCI, PXI (use PCI version of blocks)
	Multiple block instance support	A/D: No, Digital I/O: Yes
	Multiple board support	Yes

# National Instruments PCI-6280 Analog Input

**Purpose** National Instruments PCI-6280 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 lines beginning with 1 even though the board manufacturer starts numbering the lines with 0.

### **Range vector**

Enter a range code for each of the channels in the channel vector. You can specify one value to be used for all channels, or specify one for each channel. This driver allows each channel to be different. This board works only in bipolar mode.

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-6280 Analog Input

Input Range (V)	Range Code
-0.2 to +0.2	0.2
-0.1 to +0.1	0.1

For example, if the first channel is -10 to +10 volts and the second and fifth channels are -1 to +1 volts, enter

[10, 1, 1]

If all channels use the same range, you can specify one entry.

## Input coupling vector

Enter a coupling code for each of the channels in the channel vector. You can specify one value for all channels, or specify one for each channel. 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 National Instruments user 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 National Instruments user manual.
DIFF	2	First analog input line connected to the positive input of the PGIA. Second analog input line connected to the

# National Instruments PCI-6280 Analog Input

Coupling	Coupling Code	Description
		negative input of the PGIA. See the National Instruments user 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 boards use a hardware number scheme for differential mode channel pairs. xPC Target drivers use a software number scheme that adds a 1 to the hardware number, as shown in the following table:

Hardware Channel Number		Software Channel Number = Hardware Channel Number + 1	
+	-	+	-
0	8	1	9
1	9	2	10
2	10	3	11
3	11	4	12
4	12	5	13
5	13	6	14
6	14	7	15
7	15	8	16

The hardware uses a second input 8 channels higher than the first channel as the negative input of the pair. In the example above, the hardware would use the thirteenth channel as a differential input with the fifth channel.

If the board is set up for differential mode, you can read the data from either of the channels in the differential pair. For example, if you have a differential pair of 1 and 9, you can read the data from channel 1 or channel 9. However, you might want to read the lower channel number of the pair because it remains unchanged when you switch the input mode between single-ended and differential.

## Scan interval

If **Channel vector** has more than one channel in it, enter the time between sampling different channels. Because execution waits for data to be available, this value increases the time to execute the block. If you enter a shorter time, you get the minimum 2e-6. If you enter a time smaller than this value, the time will still be 4e-6, which is the fastest this board can acquire data. See the National Instruments user manual for a discussion about settling time and the need to control the time between sampling different channels depending on the impedance of the signal source.

## 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-6280 Digital Input

---

**Purpose** National Instruments PCI-6280 Digital Input block

**Library** xPC Target Library for National Instruments

**Note** Models can have multiple digital input blocks for one physical board. In this case, the xPC Target environment executes the blocks based on their sorted order as determined by the Simulink software.

Each board has one set of static digital I/O ports. You can independently configure each port to be input or output.

If a digital input block shares channels with a digital output block, the input block reads the last value written to the digital output block on the overlapping channels.

## Scaling of Input to Output

Hardware Input	Block Output Data Type	Scaling
TTL	Double	TTL low $\geq 0.0$ 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 the first eight 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.

### 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
```

# National Instruments PCI-6280 Digital Output

**Purpose** National Instruments PCI-6280 Digital Output block

**Library** xPC Target Library for National Instruments

**Note** Models can have multiple digital output blocks for one physical board. In this case, the xPC Target environment executes the blocks based on their sorted order as determined by the Simulink software.

Each board has one set of digital I/O ports. You can independently configure each port to be input or output.

If a digital input block shares channels with a digital output block, the input block reads the last value written to the digital output block on the overlapping channels.

The block compares the value written to each port with 0.5.

Value written	Block writes digital output with a value of...
$\geq 0.5$	1
$< 0.5$	0

## Scaling of Output to Input

Hardware Output	Block Input Data Type	Scaling
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 the first eight 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.

## **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-6280 Incremental Encoder

**Purpose** National Instruments PCI-6280 Incremental Encoder block

**Library** xPC Target Library for National Instruments

**Scaling Input to Output**

Hardware Input	Block Output Data Type	Scaling
TTL level quadrature encoder signals	Double	counts

## Block Parameters

### Channel

From the list, choose 0 or 1 for the channel counter you want to use. For information on input for these counters, see the National Instruments M Series user manual. The pin associations are:

Counter	Channel 0	Channel 1
A	PFI 8 (Pin 37)	PFI 3 (Pin 42)
B	PFI 10 (Pin 45)	PFI 11 (Pin 46)
Z	PFI 9 (Pin 3)	PFI 4 (Pin 41)

### Counting mode

From the list, select a counting mode:

Mode	Description
Quadrature Mode x1	Count on rising edges of the A signal
Quadrature Mode x2	Count on both rising and falling edges of the A signal
Quadrature Mode x4	Count on both rising and falling edges of both A and B signals

The level of the opposite signal when an edge occurs determines the count direction. For example, a rising edge of A when B is high counts one way, while an edge of A when B is low counts the other. For more information, see the National Instruments M Series user manual.

## Initial count

The initial count specifies the initial value for the counter. The block resets the count to this value under the following conditions:

- When the model starts
- When the following is true, which typically occurs once per revolution for a rotary encoder.
  - **Reload at index pulse** check box is selected
  - Index pulse (Z input) is true
  - The phases of A and B match the setting of **Index phase**

## Reload at index pulse

Select this check box to reset the counter to the value of **Initial count** at each index pulse. The reset occurs when:

- The encoder outputs a high level on its index output
- The phases of A and B match the setting of **Index phase**

## Index phase

If you have selected the **Reload at index pulse** check box, the **Index phase** parameter specifies the phase of the quadrature signals during which the count reloads with **Initial count**. Your choice of value primarily depends on the particular incremental encoder you use. From the list, select one:

- A low B low
- A low B high
- A high B low
- A high B high

# National Instruments PCI-6280 Incremental Encoder

---

To choose an **Index phase** for reset, use an oscilloscope, triggered on the index pulse, to observe the states of counters A and B. If your choice does not meet the encoder requirements, resets may be inconsistent.

## Filter

You can apply a digital debouncing filter to the input pins before processing.

From the list, select a filter value to ignore pulses that are shorter than the filter time. The value of `None` indicates that there is no filtering of the input.

- `None`
- `Minimum pulse width 125 nanoseconds`
- `Minimum pulse width 6.25 microseconds`
- `Minimum pulse width 1.25 milliseconds`

## Sample time

Model 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 locate the board automatically.

If two or more boards of this type are in the target PC, for the board associated with this driver block, enter:

- `Bus number`
- `PCI slot number`

Use the format `[BusNumber, SlotNumber]`. To determine the bus number and the PCI slot number, type

```
getxpcpci
```

# National Instruments PCI-6280 PWM Generate

**Purpose** National Instruments PCI-6280 PWM Generate block

**Library** xPC Target Library for National Instruments

**Description** Use this block to generate variable frequency and pulse width output. All M Series boards have two 32-bit counters that you can use to generate pulse trains. Inputs are in counts of the 80 MHz reference frequency on the onboard clock.

- H input is the number of counts that the output is high.
- L input is the number of counts that the output is low.

If the H input goes to 0, the output stays low. If the L input goes to 0, the output stays high. If both are 0, the output goes low.

## Block Parameters

### Counter

From the list, select the counter, 0 or 1, that you want to use. The output for these counters is on the following pins:

Counter	Pin
0	PFI 12 (Pin 2)
1	PFI 13 (Pin 40)

### Initial high count

Enter the initial H count to take effect between the time the Simulink engine calls the `md1Start` routine for this block and the first time it calls `md1Outputs`. This length of time depends on the number of blocks in the model and the sorted order as determined by the Simulink software.

If you enter a value of 0, the output stays low until the first time the Simulink engine calls `md1Outputs`.

# National Instruments PCI-6280 PWM Generate

---

## Initial low count

Enter the initial L count to take effect between the time the Simulink engine calls the `md1Start` routine for this block and the first time it calls `md1Outputs`. This length of time depends on the number of blocks in the model and the sorted order as determined by the Simulink software. If you enter a value of 0, the output stays high until the first time the Simulink engine calls `md1Outputs`, unless you have also set H to 0.

## Arm input

Select this check box to add a third input port to the block.

- If the input connected to this port is greater than 0.5, the block arms the counter. Output then occurs.
- If the input connected to this port is less than or equal to 0.5, the block disarms the counter and the output goes to the level of the value in the **Disarm level** parameter.

## Disarm level

From the list, select the disarm level `Disarm to LOW` or `Disarm to HIGH`. If the model disarms the block and the level of the value is **Arm input**, the output goes to the level you select here.

## Stop level

From the list, select the stop level `Stop to LOW` or `Stop to HIGH`. When model execution stops, the output goes to the level you choose here.

## Sample time

Model 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 locate the board automatically.

If two or more boards of this type are in the target PC, for the board associated with this driver block, enter:

- Bus number
- PCI slot number

Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```

# National Instruments PCI-6280 PFI Digital Input

**Purpose** National Instruments PCI-6280 PFI Digital Input block

**Library** xPC Target Library for National Instruments

**Description** A model can have more than one PFI digital input block for any given physical board. If a model has more than one digital input block, it executes the blocks in the order that the Simulink software has determined.

The two counters and the PFI digital input and output blocks share the 16 PFI digital I/O lines. Follow these rules when using these blocks:

You can read a PFI bit with this digital input block when...	You cannot read a PFI bit with this digital input block when...
PWM measure block is using the bit	PWM generate block is using the bit
Incremental encoder block is using the bit	PFI digital output block is using the bit

## Scaling of Input to Output

Hardware Input	Block Output Data Type	Scaling
TTL	Double	TTL low $\geq 0.0$ TTL high = 1.0

## Block Parameters

### PFI bit vector (0-15)

This block uses 0 based numbering to reflect the names of the bits described in the National Instruments M Series user manual.

Enter a vector that contains all bits you want to monitor. For example, to read bits PFI 0, PFI 5, and PFI 7, enter

[0, 5, 7]

## Sample time

Model 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 locate the board automatically.

If two or more boards of this type are in the target PC, for the board associated with this driver block, enter:

- Bus number
- PCI slot number

Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```

# National Instruments PCI-6280 PFI Digital Output

---

**Purpose** National Instruments PCI-6280 PFI Digital Output block

**Library** xPC Target Library for National Instruments

**Description** A model can have more than one PFI digital input block for any given physical board. If a model has more than one digital input block, it executes the blocks in the order that the Simulink software has determined.

The two counters and the PFI digital input and output blocks share the 16 PFI digital I/O lines. Follow these rules when using these blocks:

- Do not use a PFI bit as a digital output when the PDF digital input block is using the bit.
- Do not use a PFI bit as a digital output when the incremental encoder is using the bit.
- Do not use a PFI bit for digital output when the PWM generate block is using the bit.

## Scaling of Output to Input

Hardware Output	Block Input Data Type	Scaling
TTL	Double	< 0.5 = TTL low ≥ 0.5 = TTL high

## Block Parameters

### PFI bit vector (0-15)

This block uses 0 based numbering to reflect the names of the bits described in the National Instruments M Series user manual.

Enter a vector that contains all bits you want to control. For example, to write bits PFI 1, PFI 3, and PFI 9, enter

[1, 3, 9]

## Reset vector

The reset vector controls the behavior at model termination. If you specify a value of 1, the output is reset to the value specified in **Initial value vector**. Enter a scalar or a vector of values, one for each output. If you specify a scalar value, the driver uses that setting for all outputs. If you specify a value of 1, when the model execution stops, the corresponding output is reset to the value specified in **Initial value vector**. If you specify a value of 0, the output 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 block outputs. Enter a scalar or a vector of values, one for each output. If you enter a scalar, the driver uses that value for all outputs.

The block applies this value as follows:

- The model uses this value when you first download the model to the target PC.
- If you set **Reset vector** to 1, the block output is reset to this value when model execution stops. If the value is greater than 0.5, the block sets output to HIGH. If the value is less than or equal to 0.5, the block sets the output to LOW.

## Sample time

Model 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 locate the board automatically.

If two or more boards of this type are in the target PC, for the board associated with this driver block, enter:

# National Instruments PCI-6280 PFI Digital Output

---

- Bus number
- PCI slot number

Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```

# National Instruments PCI-6280 PWM Measure

**Purpose** National Instruments PCI-6280 PWM Measure block

**Library** xPC Target Library for National Instruments

**Description** Use this block to measure the pulse width or the period of a signal.  
To measure the pulse width of a signal, connect the signal to the gate, as follows:

Counter	Gate Pin
0	PFI 9 (Pin 3)
1	PFI 4 (Pin 41)

The input waveform must have fast rise and fall times and conform to TTL signal levels of [0,5] volts. Use external signal conditioning to achieve these levels. If you input a sine wave, the decision points are less definite and the measurement is noisy. In extreme cases, the gate can transition multiple times during a slow signal transition.

This block does not queue or buffer measurements. If the gate changes state, the driver holds the current count, overwriting any previous value.

This block returns a value that is always the length of the last gate period before the block executes.

To compute the duty cycle percentage, use two counters. Because the period of a cycle is H counter + L counter, you can measure any two of these counters and compute the third.

You cannot measure a 0 width pulse.

## Block Parameters

### Counter

From the list, select a counter, 0 or 1.

### Trigger mode

From the list, select the trigger mode to use:

# National Instruments PCI-6280 PWM Measure

Mode	Description
Width of Gate High	Returns the number of 80 MHz cycles that occur when a single cycle of the input is high.
Width of Gate Low	Returns the number of 80 MHz cycles that occur when a single cycle of the input is low.
Between Rising Edges	Returns the number of 80 MHz cycles that occur between rising edges of the input signal. This cycle is the full period of the input signal.
Between Falling Edges	Returns the number of 80 MHz cycles that occur between falling edges of the input signal. This cycle is the full period of the input signal.

If the input signal has both fast rise and fall edges, **Between Rising Edges** and **Between Falling Edges** return the same value. If the input is asymmetric, one of these options can give a better result than the other.

## Filter

You can apply a digital debouncing filter to the input pins before processing.

From the list, select a filter value to ignore pulses that are shorter than the filter time. The block ignores pulses shorter than the chosen duration. The value of **None** indicates that there is no filtering of the input

- None
- Minimum pulse width 125 nanoseconds
- Minimum pulse width 6.25 microseconds
- Minimum pulse width 1.25 milliseconds

# National Instruments PCI-6280 PWM Measure

---

## Sample time

Model 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 locate the board automatically.

If two or more boards of this type are in the target PC, for the board associated with this driver block, enter:

- Bus number
- PCI slot number

Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```

# National Instruments PCI-6281

---

**Purpose** Support for National Instruments PCI-6281 board

**Board** National Instruments PCI-6281

**General Description** The PCI-6281 is a data acquisition I/O board with up to 16 single-ended or 8 differential 18-bit analog inputs, two 16-bit analog outputs, 24 digital I/O lines, and two 32-bit 80 MHz counters.

- National Instruments PCI-6281 Analog Input
- National Instruments PCI-6281 Analog Output
- National Instruments PCI-6281 Digital Input
- National Instruments PCI-6281 Digital Output
- National Instruments PCI-6281 Incremental Encoder
- National Instruments PCI-6281 PFI Digital Input
- National Instruments PCI-6281 PFI Digital Output
- National Instruments PCI-6281 PWM Generate
- National Instruments PCI-6281 PWM Measure

The xPC Target block library supports this board with these driver blocks:

<b>Board Characteristics</b>	Board name	PCI-6281
	Manufacturer	National Instruments
	Bus type	PCI, PXI (use PCI version of blocks)
	Multiple block instance support	A/D: No, D/A: No, Digital I/O: Yes
	Multiple board support	Yes

# National Instruments PCI-6281 Analog Input

**Purpose** National Instruments PCI-6281 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 lines beginning with 1 even though the board manufacturer starts numbering the lines with 0.

### Range vector

Enter a range code for each of the channels in the channel vector. You can specify one value to be used for all channels, or specify one for each channel. This driver allows each channel to be different. This board works only in bipolar mode.

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-6281 Analog Input

Input Range (V)	Range Code
-0.2 to +0.2	0.2
-0.1 to +0.1	0.1

For example, if the first channel is -10 to +10 volts and the second and fifth channels are -1 to +1 volts, enter

[10,1,1]

If all channels use the same range, you can specify one entry.

## Input coupling vector

Enter a coupling code for each of the channels in the channel vector. You can specify one value for all channels, or specify one for each channel. 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 National Instruments user 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 National Instruments user manual.
DIFF	2	First analog input line connected to the positive input of the PGIA. Second analog input line connected to the

# National Instruments PCI-6281 Analog Input

Coupling	Coupling Code	Description
		negative input of the PGIA. See the National Instruments user 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 boards use a hardware number scheme for differential mode channel pairs. xPC Target drivers use a software number scheme that adds a 1 to the hardware number, as shown in the following table:

Hardware Channel Number		Software Channel Number = Hardware Channel Number + 1	
+	-	+	-
0	8	1	9
1	9	2	10
2	10	3	11
3	11	4	12
4	12	5	13
5	13	6	14
6	14	7	15
7	15	8	16

The hardware uses a second input 8 channels higher than the first channel as the negative input of the pair. In the example above, the hardware would use the thirteenth channel as a differential input with the fifth channel.

# National Instruments PCI-6281 Analog Input

---

If the board is set up for differential mode, you can read the data from either of the channels in the differential pair. For example, if you have a differential pair of 1 and 9, you can read the data from channel 1 or channel 9. However, you might want to read the lower channel number of the pair because it remains unchanged when you switch the input mode between single-ended and differential.

## Scan interval

If **Channel vector** has more than one channel in it, enter the time between sampling different channels. Because execution waits for data to be available, this value increases the time to execute the block. If you enter a shorter time, you get the minimum 2e-6. If you enter a time smaller than this value, the time will still be 4e-6, which is the fastest this board can acquire data. See the National Instruments user manual for a discussion about settling time and the need to control the time between sampling different channels depending on the impedance of the signal source.

## 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-6281 Analog Output

**Purpose** National Instruments PCI-6281 Analog Output block

**Library** xPC Target Library for National Instruments

## Scaling of Output to Input

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 though 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
$V_{ref} \times sig$ , where $-1 \leq sig \leq +1$	APFIO

# National Instruments PCI-6281 Analog Output

---

For example, if the first channel is -10 to +10 volts and the second channel is -5 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 or a multiple of the base sample 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-6281 Digital Input

**Purpose** National Instruments PCI-6281 Digital Input block

**Library** xPC Target Library for National Instruments

**Note** Models can have multiple digital input blocks for one physical board. In this case, the xPC Target environment executes the blocks based on their sorted order as determined by the Simulink software.

Each board has one set of static digital I/O ports. You can independently configure each port to be input or output.

If a digital input block shares channels with a digital output block, the input block reads the last value written to the digital output block on the overlapping channels.

## Scaling of Input to Output

Hardware Input	Block Output Data Type	Scaling
TTL	Double	TTL low $\geq$ 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 the first eight 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.

### 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

# National Instruments PCI-6281 Digital Input

---

- 1

to automatically locate the board.

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of 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-6281 Digital Output

**Purpose** National Instruments PCI-6281 Digital Output block

**Library** xPC Target Library for National Instruments

**Note** Models can have multiple digital output blocks for one physical board. In this case, the xPC Target environment executes the blocks based on their sorted order as determined by the Simulink software.

Each board has one set of digital I/O ports. You can independently configure each port to be input or output.

If a digital input block shares channels with a digital output block, the input block reads the last value written to the digital output block on the overlapping channels.

The block compares the value written to each port with 0.5.

Value written	Block writes digital output with a value of...
$\geq 0.5$	1
$< 0.5$	0

## Scaling of Output to Input

Hardware Output	Block Input Data Type	Scaling
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 the first eight digital outputs, enter

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

# National Instruments PCI-6281 Digital Output

---

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-6281 Incremental Encoder

**Purpose** National Instruments PCI-6281 Incremental Encoder block

**Library** xPC Target Library for National Instruments

**Scaling Input to Output**

Hardware Input	Block Output Data Type	Scaling
TTL level quadrature encoder signals	Double	counts

## Block Parameters

### Channel

From the list, choose 0 or 1 for the channel counter you want to use. For information on input for these counters, see the National Instruments M Series user manual. The pin associations are:

Counter	Channel 0	Channel 1
A	PFI 8 (Pin 37)	PFI 3 (Pin 42)
B	PFI 10 (Pin 45)	PFI 11 (Pin 46)
Z	PFI 9 (Pin 3)	PFI 4 (Pin 41)

### Counting mode

From the list, select a counting mode:

Mode	Description
Quadrature Mode x1	Count on rising edges of the A signal
Quadrature Mode x2	Count on both rising and falling edges of the A signal
Quadrature Mode x4	Count on both rising and falling edges of both A and B signals

# National Instruments PCI-6281 Incremental Encoder

---

The level of the opposite signal when an edge occurs determines the count direction. For example, a rising edge of A when B is high counts one way, while an edge of A when B is low counts the other. For more information, see the National Instruments M Series user manual.

## Initial count

The initial count specifies the initial value for the counter. The block resets the count to this value under the following conditions:

- When the model starts
- When the following is true, which typically occurs once per revolution for a rotary encoder.
  - **Reload at index pulse** check box is selected
  - Index pulse (Z input) is true
  - The phases of A and B match the setting of **Index phase**

## Reload at index pulse

Select this check box to reset the counter to the value of **Initial count** at each index pulse. The reset occurs when:

- The encoder outputs a high level on its index output
- The phases of A and B match the setting of **Index phase**

## Index phase

If you have selected the **Reload at index pulse** check box, the **Index phase** parameter specifies the phase of the quadrature signals during which the count reloads with **Initial count**. Your choice of value primarily depends on the particular incremental encoder you use. From the list, select one:

- A low B low
- A low B high
- A high B low
- A high B high

# National Instruments PCI-6281 Incremental Encoder

---

To choose an **Index phase** for reset, use an oscilloscope, triggered on the index pulse, to observe the states of counters A and B. If your choice does not meet the encoder requirements, resets may be inconsistent.

## Filter

You can apply a digital debouncing filter to the input pins before processing.

From the list, select a filter value to ignore pulses that are shorter than the filter time. The value of **None** indicates that there is no filtering of the input.

- None
- Minimum pulse width 125 nanoseconds
- Minimum pulse width 6.25 microseconds
- Minimum pulse width 1.25 milliseconds

## Sample time

Model 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 locate the board automatically.

If two or more boards of this type are in the target PC, for the board associated with this driver block, enter:

- Bus number
- PCI slot number

Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```

# National Instruments PCI-6281 PFI Digital Input

**Purpose** National Instruments PCI-6281 PFI Digital Input block

**Library** xPC Target Library for National Instruments

**Description** A model can have more than one PFI digital input block for any given physical board. If a model has more than one digital input block, it executes the blocks in the order that the Simulink software has determined.

The two counters and the PFI digital input and output blocks share the 16 PFI digital I/O lines. Follow these rules when using these blocks:

You can read a PFI bit with this digital input block when...	You cannot read a PFI bit with this digital input block when...
PWM measure block is using the bit	PWM generate block is using the bit
Incremental encoder block is using the bit	PFI digital output block is using the bit

## Scaling of Input to Output

Hardware Input	Block Output Data Type	Scaling
TTL	Double	TTL low $\geq 0.0$ TTL high = 1.0

## Block Parameters

### PFI bit vector (0-15)

This block uses 0 based numbering to reflect the names of the bits described in the National Instruments M Series user manual.

Enter a vector that contains all bits you want to monitor. For example, to read bits PFI 0, PFI 5, and PFI 7, enter

[0, 5, 7]

## Sample time

Model 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 locate the board automatically.

If two or more boards of this type are in the target PC, for the board associated with this driver block, enter:

- Bus number
- PCI slot number

Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```

# National Instruments PCI-6281 PFI Digital Output

---

**Purpose** National Instruments PCI-6281 PFI Digital Output block

**Library** xPC Target Library for National Instruments

**Description** A model can have more than one PFI digital input block for any given physical board. If a model has more than one digital input block, it executes the blocks in the order that the Simulink software has determined.

The two counters and the PFI digital input and output blocks share the 16 PFI digital I/O lines. Follow these rules when using these blocks:

- Do not use a PFI bit as a digital output when the PDF digital input block is using the bit.
- Do not use a PFI bit as a digital output when the incremental encoder is using the bit.
- Do not use a PFI bit for digital output when the PWM generate block is using the bit.

## Scaling of Output to Input

Hardware Output	Block Input Data Type	Scaling
TTL	Double	< 0.5 = TTL low ≥ 0.5 = TTL high

## Block Parameters

### PFI bit vector (0-15)

This block uses 0 based numbering to reflect the names of the bits described in the National Instruments M Series user manual.

Enter a vector that contains all bits you want to control. For example, to write bits PFI 1, PFI 3, and PFI 9, enter

[1, 3, 9]

## Reset vector

The reset vector controls the behavior at model termination. If you specify a value of 1, the output is reset to the value specified in **Initial value vector**. Enter a scalar or a vector of values, one for each output. If you specify a scalar value, the driver uses that setting for all outputs. If you specify a value of 1, when the model execution stops, the corresponding output is reset to the value specified in **Initial value vector**. If you specify a value of 0, the output 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 block outputs. Enter a scalar or a vector of values, one for each output. If you enter a scalar, the driver uses that value for all outputs.

The block applies this value as follows:

- The model uses this value when you first download the model to the target PC.
- If you set **Reset vector** to 1, the block output is reset to this value when model execution stops. If the value is greater than 0.5, the block sets output to HIGH. If the value is less than or equal to 0.5, the block sets the output to LOW.

## Sample time

Model 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 locate the board automatically.

If two or more boards of this type are in the target PC, for the board associated with this driver block, enter:

# National Instruments PCI-6281 PFI Digital Output

---

- Bus number
- PCI slot number

Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```

# National Instruments PCI-6281 PWM Generate

**Purpose** National Instruments PCI-6281 PWM Generate block

**Library** xPC Target Library for National Instruments

**Description** Use this block to generate variable frequency and pulse width output. All M Series boards have two 32-bit counters that you can use to generate pulse trains. Inputs are in counts of the 80 MHz reference frequency on the onboard clock.

- H input is the number of counts that the output is high.
- L input is the number of counts that the output is low.

If the H input goes to 0, the output stays low. If the L input goes to 0, the output stays high. If both are 0, the output goes low.

## Block Parameters

### Counter

From the list, select the counter, 0 or 1, that you want to use. The output for these counters is on the following pins:

Counter	Pin
0	PFI 12 (Pin 2)
1	PFI 13 (Pin 40)

### Initial high count

Enter the initial H count to take effect between the time the Simulink engine calls the `md1Start` routine for this block and the first time it calls `md1Outputs`. This length of time depends on the number of blocks in the model and the sorted order as determined by the Simulink software.

If you enter a value of 0, the output stays low until the first time the Simulink engine calls `md1Outputs`.

# National Instruments PCI-6281 PWM Generate

---

## Initial low count

Enter the initial L count to take effect between the time the Simulink engine calls the `md1Start` routine for this block and the first time it calls `md1Outputs`. This length of time depends on the number of blocks in the model and the sorted order as determined by the Simulink software. If you enter a value of 0, the output stays high until the first time the Simulink engine calls `md1Outputs`, unless you have also set H to 0.

## Arm input

Select this check box to add a third input port to the block.

- If the input connected to this port is greater than 0.5, the block arms the counter. Output then occurs.
- If the input connected to this port is less than or equal to 0.5, the block disarms the counter and the output goes to the level of the value in the **Disarm level** parameter.

## Disarm level

From the list, select the disarm level **Disarm to LOW** or **Disarm to HIGH**. If the model disarms the block and the level of the value is **Arm input**, the output goes to the level you select here.

## Stop level

From the list, select the stop level **Stop to LOW** or **Stop to HIGH**. When model execution stops, the output goes to the level you choose here.

## Sample time

Model 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 locate the board automatically.

If two or more boards of this type are in the target PC, for the board associated with this driver block, enter:

- Bus number
- PCI slot number

Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```

# National Instruments PCI-6281 PWM Measure

---

**Purpose** National Instruments PCI-6281 PWM Measure block

**Library** xPC Target Library for National Instruments

**Description** Use this block to measure the pulse width or the period of a signal.  
To measure the pulse width of a signal, connect the signal to the gate, as follows:

Counter	Gate Pin
0	PFI 9 (Pin 3)
1	PFI 4 (Pin 41)

The input waveform must have fast rise and fall times and conform to TTL signal levels of [0,5] volts. Use external signal conditioning to achieve these levels. If you input a sine wave, the decision points are less definite and the measurement is noisy. In extreme cases, the gate can transition multiple times during a slow signal transition.

This block does not queue or buffer measurements. If the gate changes state, the driver holds the current count, overwriting any previous value.

This block returns a value that is always the length of the last gate period before the block executes.

To compute the duty cycle percentage, use two counters. Because the period of a cycle is H counter + L counter, you can measure any two of these counters and compute the third.

You cannot measure a 0 width pulse.

## Block Parameters

### Counter

From the list, select a counter, 0 or 1.

### Trigger mode

From the list, select the gating mode to use:

# National Instruments PCI-6281 PWM Measure

Mode	Description
Width of Gate High	Returns the number of 80 MHz cycles that occur when a single cycle of the input is high.
Width of Gate Low	Returns the number of 80 MHz cycles that occur when a single cycle of the input is low.
Between Rising Edges	Returns the number of 80 MHz cycles that occur between rising edges of the input signal. This cycle is the full period of the input signal.
Between Falling Edges	Returns the number of 80 MHz cycles that occur between falling edges of the input signal. This cycle is the full period of the input signal.

If the input signal has both fast rise and fall edges, **Between Rising Edges** and **Between Falling Edges** return the same value. If the input is asymmetric, one of these options can give a better result than the other.

## Filter

You can apply a digital debouncing filter to the input pins before processing.

From the list, select a filter value to ignore pulses that are shorter than the filter time. The block ignores pulses shorter than the chosen duration. The value of **None** indicates that there is no filtering of the input

- None
- Minimum pulse width 125 nanoseconds
- Minimum pulse width 6.25 microseconds
- Minimum pulse width 1.25 milliseconds

# National Instruments PCI-6281 PWM Measure

---

## Sample time

Model 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 locate the board automatically.

If two or more boards of this type are in the target PC, for the board associated with this driver block, enter:

- Bus number
- PCI slot number

Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```

**Purpose** Support for National Instruments PCI-6289 board

**Board** National Instruments PCI-6289

**General Description** The PCI-6289 is a data acquisition I/O board with up to 32 single-ended or 16 differential 18-bit analog inputs, four 16-bit analog outputs, 48 digital I/O lines, and two 32-bit 80 MHz counters.

The xPC Target block library supports this board with these driver blocks:

- National Instruments PCI-6289 Analog Input
- National Instruments PCI-6289 Analog Output
- National Instruments PCI-6289 Digital Input
- National Instruments PCI-6289 Digital Output
- National Instruments PCI-6289 Incremental Encoder
- National Instruments PCI-6289 PFI Digital Input
- National Instruments PCI-6289 PFI Digital Output
- National Instruments PCI-6289 PWM Generate
- National Instruments PCI-6289 PWM Measure

## Board Characteristics

Board name	PCI-6289
Manufacturer	National Instruments
Bus type	PCI, PXI (use PCI version of blocks)
Multiple block instance support	A/D: No, D/A: No, Digital I/O: Yes
Multiple board support	Yes

# National Instruments PCI-6289 Analog Input

**Purpose** National Instruments PCI-6289 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 32. 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 lines beginning with 1 even though the board manufacturer starts numbering the lines with 0.

### **Range vector**

Enter a range code for each of the channels in the channel vector. You can specify one value to be used for all channels, or specify one for each channel. This driver allows each channel to be different. This board works only in bipolar mode.

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-6289 Analog Input

Input Range (V)	Range Code
-0.2 to +0.2	0.2
-0.1 to +0.1	0.1

For example, if the first channel is -10 to +10 volts and the second and fifth channels are -1 to +1 volts, enter

[10, 1, 1]

If all channels use the same range, you can specify one entry.

## Input coupling vector

Enter a coupling code for each of the channels in the channel vector. You can specify one value for all channels, or specify one for each channel. 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 National Instruments user 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 National Instruments user manual.
DIFF	2	First analog input line connected to the positive input of the PGIA. Second analog input line connected to the

# National Instruments PCI-6289 Analog Input

Coupling	Coupling Code	Description
		negative input of the PGIA. See the National Instruments user 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 boards use a hardware number scheme for differential mode channel pairs. xPC Target drivers use a software number scheme that adds a 1 to the hardware number, as shown in the following table:

Hardware Channel Number		Software Channel Number = Hardware Channel Number + 1	
+	-	+	-
0	8	1	9
1	9	2	10
2	10	3	11
3	11	4	12
4	12	5	13
5	13	6	14
6	14	7	15
7	15	8	16
16	24	17	25
17	25	18	26
18	26	19	27
19	27	20	28

# National Instruments PCI-6289 Analog Input

Hardware Channel Number		Software Channel Number = Hardware Channel Number + 1	
20	28	21	29
21	29	22	30
22	30	23	31
23	31	24	32

The hardware uses a second input 8 channels higher than the first channel as the negative input of the pair. In the example above, the hardware would use the thirteenth channel as a differential input with the fifth channel.

If the board is set up for differential mode, you can read the data from either of the channels in the differential pair. For example, if you have a differential pair of 1 and 9, you can read the data from channel 1 or channel 9. However, you might want to read the lower channel number of the pair because it remains unchanged when you switch the input mode between single-ended and differential.

## Scan interval

If **Channel vector** has more than one channel in it, enter the time between sampling different channels. Because execution waits for data to be available, this value increases the time to execute the block. If you enter a shorter time, you get the minimum 2e-6. If you enter a time smaller than this value, the time will still be 4e-6, which is the fastest this board can acquire data. See the National Instruments user manual for a discussion about settling time and the need to control the time between sampling different channels depending on the impedance of the signal source.

## Sample time

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

# National Instruments PCI-6289 Analog 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
```

# National Instruments PCI-6289 Analog Output

**Purpose** National Instruments PCI-6289 Analog Output block

**Library** xPC Target Library for National Instruments

## Scaling of Output to Input

Hardware Output	Block Input Data Type	Scaling
Volts	Double	1

## Block Parameters

### Channel vector

Enter 1 to 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 all analog output channels, enter

[1,2,3,4]

Number the channels begin with 1 even though 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

# National Instruments PCI-6289 Analog Output

Input Range (V)	Range Code
$V_{ref} \times sig$ , where $-1 \leq sig \leq +1$	APFI0
$V_{ref} \times sig$ , where $-1 \leq sig \leq +1$	APFI1

For example, if the first channel is -10 to +10 volts and the second channel is -5 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 or a multiple of the base sample time.

## PCI Slot (-1:autosearch)

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

-1

# National Instruments PCI-6289 Analog Output

---

to automatically locate the board.

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of 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-6289 Digital Input

---

**Purpose** National Instruments PCI-6289 Digital Input block

**Library** xPC Target Library for National Instruments

**Note** Models can have multiple digital input blocks for one physical board. In this case, the xPC Target environment executes the blocks based on their sorted order as determined by the Simulink software.

Each board has one set of static digital I/O ports. You can independently configure each port to be input or output.

If a digital input block shares channels with a digital output block, the input block reads the last value written to the digital output block on the overlapping channels.

## Scaling of Input to Output

Hardware Input	Block Output Data Type	Scaling
TTL	Double	TTL low $\geq 0.0$ TTL high = 1.0

## Block Parameters

### Channel vector

Enter numbers between 1 and 32 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 the first eight 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.

### 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
```

# National Instruments PCI-6289 Digital Output

**Purpose** National Instruments PCI-6289 Digital Output block

**Library** xPC Target Library for National Instruments

**Note** Models can have multiple digital output blocks for one physical board. In this case, the xPC Target environment executes the blocks based on their sorted order as determined by the Simulink software.

Each board has one set of digital I/O ports. You can independently configure each port to be input or output.

If a digital input block shares channels with a digital output block, the input block reads the last value written to the digital output block on the overlapping channels.

The block compares the value written to each port with 0.5.

Value written	Block writes digital output with a value of...
$\geq 0.5$	1
$< 0.5$	0

## Scaling of Output to Input

Hardware Output	Block Input Data Type	Scaling
TTL	Double	$< 0.5 = \text{TTL low}$ $\geq 0.5 = \text{TTL high}$

## Block Parameters

### Channel vector

Enter numbers between 1 and 32 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 the first eight 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.

## **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-6289 Incremental Encoder

**Purpose** National Instruments PCI-6289 Incremental Encoder block

**Library** xPC Target Library for National Instruments

**Scaling Input to Output**

Hardware Input	Block Output Data Type	Scaling
TTL level quadrature encoder signals	Double	counts

## Block Parameters

### Channel

From the list, choose 0 or 1 for the channel counter you want to use. For information on input for these counters, see the National Instruments M Series user manual. The pin associations are:

Counter	Channel 0	Channel 1
A	PFI 8 (Pin 37)	PFI 3 (Pin 42)
B	PFI 10 (Pin 45)	PFI 11 (Pin 46)
Z	PFI 9 (Pin 3)	PFI 4 (Pin 41)

### Counting mode

From the list, select a counting mode:

Mode	Description
Quadrature Mode x1	Count on rising edges of the A signal
Quadrature Mode x2	Count on both rising and falling edges of the A signal
Quadrature Mode x4	Count on both rising and falling edges of both A and B signals

The level of the opposite signal when an edge occurs determines the count direction. For example, a rising edge of A when B is high counts one way, while an edge of A when B is low counts the other. For more information, see the National Instruments M Series user manual.

## Initial count

The initial count specifies the initial value for the counter. The block resets the count to this value under the following conditions:

- When the model starts
- When the following is true, which typically occurs once per revolution for a rotary encoder.
  - **Reload at index pulse** check box is selected
  - Index pulse (Z input) is true
  - The phases of A and B match the setting of **Index phase**

## Reload at index pulse

Select this check box to reset the counter to the value of **Initial count** at each index pulse. The reset occurs when:

- The encoder outputs a high level on its index output
- The phases of A and B match the setting of **Index phase**

## Index phase

If you have selected the **Reload at index pulse** check box, the **Index phase** parameter specifies the phase of the quadrature signals during which the count reloads with **Initial count**. Your choice of value primarily depends on the particular incremental encoder you use. From the list, select one:

- A low B low
- A low B high
- A high B low
- A high B high

# National Instruments PCI-6289 Incremental Encoder

---

To choose an **Index phase** for reset, use an oscilloscope, triggered on the index pulse, to observe the states of counters A and B. If your choice does not meet the encoder requirements, resets may be inconsistent.

## Filter

You can apply a digital debouncing filter to the input pins before processing.

From the list, select a filter value to ignore pulses that are shorter than the filter time. The value of `None` indicates that there is no filtering of the input.

- `None`
- `Minimum pulse width 125 nanoseconds`
- `Minimum pulse width 6.25 microseconds`
- `Minimum pulse width 1.25 milliseconds`

## Sample time

Model 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 locate the board automatically.

If two or more boards of this type are in the target PC, for the board associated with this driver block, enter:

- `Bus number`
- `PCI slot number`

Use the format `[BusNumber, SlotNumber]`. To determine the bus number and the PCI slot number, type

```
getxpcpci
```

# National Instruments PCI-6289 PFI Digital Input

**Purpose** National Instruments PCI-6289 PFI Digital Input block

**Library** xPC Target Library for National Instruments

**Description** A model can have more than one PFI digital input block for any given physical board. If a model has more than one digital input block, it executes the blocks in the order that the Simulink software has determined.

The two counters and the PFI digital input and output blocks share the 16 PFI digital I/O lines. Follow these rules when using these blocks:

<b>You can read a PFI bit with this digital input block when...</b>	<b>You cannot read a PFI bit with this digital input block when...</b>
PWM measure block is using the bit	PWM generate block is using the bit
Incremental encoder block is using the bit	PFI digital output block is using the bit

## Scaling of Input to Output

<b>Hardware Input</b>	<b>Block Output Data Type</b>	<b>Scaling</b>
TTL	Double	TTL low $\geq 0.0$ TTL high = 1.0

## Block Parameters

### PFI bit vector (0-15)

This block uses 0 based numbering to reflect the names of the bits described in the National Instruments M Series user manual.

Enter a vector that contains all bits you want to monitor. For example, to read bits PFI 0, PFI 5, and PFI 7, enter

[0, 5, 7]

# National Instruments PCI-6289 PFI Digital Input

---

## Sample time

Model 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 locate the board automatically.

If two or more boards of this type are in the target PC, for the board associated with this driver block, enter:

- Bus number
- PCI slot number

Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```

# National Instruments PCI-6289 PFI Digital Output

**Purpose** National Instruments PCI-6289 PFI Digital Output block

**Library** xPC Target Library for National Instruments

**Description** A model can have more than one PFI digital input block for any given physical board. If a model has more than one digital input block, it executes the blocks in the order that the Simulink software has determined.

The two counters and the PFI digital input and output blocks share the 16 PFI digital I/O lines. Follow these rules when using these blocks:

- Do not use a PFI bit as a digital output when the PDF digital input block is using the bit.
- Do not use a PFI bit as a digital output when the incremental encoder is using the bit.
- Do not use a PFI bit for digital output when the PWM generate block is using the bit.

## Scaling of Output to Input

Hardware Output	Block Input Data Type	Scaling
TTL	Double	< 0.5 = TTL low ≥ 0.5 = TTL high

## Block Parameters

### PFI bit vector (0-15)

This block uses 0 based numbering to reflect the names of the bits described in the National Instruments M Series user manual.

Enter a vector that contains all bits you want to control. For example, to write bits PFI 1, PFI 3, and PFI 9, enter

[1, 3, 9]

# National Instruments PCI-6289 PFI Digital Output

---

## Reset vector

The reset vector controls the behavior at model termination. If you specify a value of 1, the output is reset to the value specified in **Initial value vector**. Enter a scalar or a vector of values, one for each output. If you specify a scalar value, the driver uses that setting for all outputs. If you specify a value of 1, when the model execution stops, the corresponding output is reset to the value specified in **Initial value vector**. If you specify a value of 0, the output 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 block outputs. Enter a scalar or a vector of values, one for each output. If you enter a scalar, the driver uses that value for all outputs.

The block applies this value as follows:

- The model uses this value when you first download the model to the target PC.
- If you set **Reset vector** to 1, the block output is reset to this value when model execution stops. If the value is greater than 0.5, the block sets output to HIGH. If the value is less than or equal to 0.5, the block sets the output to LOW.

## Sample time

Model 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 locate the board automatically.

If two or more boards of this type are in the target PC, for the board associated with this driver block, enter:

# National Instruments PCI-6289 PFI Digital Output

---

- Bus number
- PCI slot number

Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```

# National Instruments PCI-6289 PWM Generate

---

**Purpose** National Instruments PCI-6289 PWM Generate block

**Library** xPC Target Library for National Instruments

**Description** Use this block to generate variable frequency and pulse width output. All M Series boards have two 32-bit counters that you can use to generate pulse trains. Inputs are in counts of the 80 MHz reference frequency on the onboard clock.

- H input is the number of counts that the output is high.
- L input is the number of counts that the output is low.

If the H input goes to 0, the output stays low. If the L input goes to 0, the output stays high. If both are 0, the output goes low.

## Block Parameters

### Counter

From the list, select the counter, 0 or 1, that you want to use. The output for these counters is on the following pins:

Counter	Pin
0	PFI 12 (Pin 2)
1	PFI 13 (Pin 40)

### Initial high count

Enter the initial H count to take effect between the time the Simulink engine calls the `md1Start` routine for this block and the first time it calls `md1Outputs`. This length of time depends on the number of blocks in the model and the sorted order as determined by the Simulink software.

If you enter a value of 0, the output stays low until the first time the Simulink engine calls `md1Outputs`.

## Initial low count

Enter the initial L count to take effect between the time the Simulink engine calls the `mdlStart` routine for this block and the first time it calls `mdlOutputs`. This length of time depends on the number of blocks in the model and the sorted order as determined by the Simulink software. If you enter a value of 0, the output stays high until the first time the Simulink engine calls `mdlOutputs`, unless you have also set H to 0.

## Arm input

Select this check box to add a third input port to the block.

- If the input connected to this port is greater than 0.5, the block arms the counter. Output then occurs.
- If the input connected to this port is less than or equal to 0.5, the block disarms the counter and the output goes to the level of the value in the **Disarm level** parameter.

## Disarm level

From the list, select the disarm level **Disarm to LOW** or **Disarm to HIGH**. If the model disarms the block and the level of the value is **Arm input**, the output goes to the level you select here.

## Stop level

From the list, select the stop level **Stop to LOW** or **Stop to HIGH**. When model execution stops, the output goes to the level you choose here.

## Sample time

Model 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 locate the board automatically.

If two or more boards of this type are in the target PC, for the board associated with this driver block, enter:

# National Instruments PCI-6289 PWM Generate

---

- Bus number
- PCI slot number

Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```

# National Instruments PCI-6289 PWM Measure

**Purpose** National Instruments PCI-6289 PWM Measure block

**Library** xPC Target Library for National Instruments

**Description** Use this block to measure the pulse width or the period of a signal.  
To measure the pulse width of a signal, connect the signal to the gate, as follows:

Counter	Gate Pin
0	PFI 9 (Pin 3)
1	PFI 4 (Pin 41)

The input waveform must have fast rise and fall times and conform to TTL signal levels of [0,5] volts. Use external signal conditioning to achieve these levels. If you input a sine wave, the decision points are less definite and the measurement is noisy. In extreme cases, the gate can transition multiple times during a slow signal transition.

This block does not queue or buffer measurements. If the gate changes state, the driver holds the current count, overwriting any previous value.

This block returns a value that is always the length of the last gate period before the block executes.

To compute the duty cycle percentage, use two counters. Because the period of a cycle is H counter + L counter, you can measure any two of these counters and compute the third.

You cannot measure a 0 width pulse.

## Block Parameters

### Counter

From the list, select a counter, 0 or 1.

### Trigger mode

From the list, select the gating mode to use:

# National Instruments PCI-6289 PWM Measure

Mode	Description
Width of Gate High	Returns the number of 80 MHz cycles that occur when a single cycle of the input is high.
Width of Gate Low	Returns the number of 80 MHz cycles that occur when a single cycle of the input is low.
Between Rising Edges	Returns the number of 80 MHz cycles that occur between rising edges of the input signal. This cycle is the full period of the input signal.
Between Falling Edges	Returns the number of 80 MHz cycles that occur between falling edges of the input signal. This cycle is the full period of the input signal.

If the input signal has both fast rise and fall edges, **Between Rising Edges** and **Between Falling Edges** return the same value. If the input is asymmetric, one of these options can give a better result than the other.

## Filter

You can apply a digital debouncing filter to the input pins before processing.

From the list, select a filter value to ignore pulses that are shorter than the filter time. The block ignores pulses shorter than the chosen duration. The value of **None** indicates that there is no filtering of the input

- None
- Minimum pulse width 125 nanoseconds
- Minimum pulse width 6.25 microseconds
- Minimum pulse width 1.25 milliseconds

## Sample time

Model 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 locate the board automatically.

If two or more boards of this type are in the target PC, for the board associated with this driver block, enter:

- Bus number
- PCI slot number

Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

```
getxpcpci
```

# National Instruments PCI-6503

---

**Purpose** Support for National Instruments PCI-6503 board

**Board** National Instruments PCI-6503

**General Description** The PCI-6503 is an I/O board with 24 digital input and output lines. The xPC Target block library 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** National Instruments 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 an 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 required 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** National Instruments 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 an 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 required 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
```

**Purpose** Support for National Instruments PCI-6527 board

**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 turn input filtering on or off individually for each of the 24 input lines. However, there is only one filter time of all of the input lines that have filtering enabled.

The xPC Target block library supports this board with these driver blocks:

- National Instruments PCI-6527 Digital Input
- National Instruments PCI-6527 Digital Output

<b>Board Characteristics</b>	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** National Instruments 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 required 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.

## 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 required 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
```

**Purpose** Support for National Instruments PCI-6528 board

**Board** National Instruments PCI-6528

**General Description** The PCI-6528 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.

The xPC Target block library supports this board with these driver blocks:

- National Instruments PCI-6527 Digital Input
- National Instruments PCI-6527 Digital Output

<b>Board Characteristics</b>	Board name	PCI-6528
	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-6528 Digital Input

---

**Purpose** PCI-6528 Digital Input block

**Library** xPC Target Library for National Instruments

**Note** The PCI-6528 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 required for each port.

# National Instruments PCI-6528 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  $\mu$ s.

## **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-6528 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-6528 Digital Output

**Purpose** PCI-6528 Digital Output block

**Library** xPC Target Library for National Instruments

**Note** The PCI-6528 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 required for each port.

# National Instruments PCI-6528 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
```

**Purpose** Support for National Instruments PCI-6601 board

**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.

The xPC Target block library 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
- National Instruments PCI-6601 Event Counter

The xPC Target software does not support the interrupt or timer functionality of the board.

<b>Board Characteristics</b>	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

---

<b>Purpose</b>	PCI-6601 Digital Output block
<b>Library</b>	xPC Target Library for National Instruments
<b>Note</b>	Only one output block can exist in a model for any given board.
<b>Block Parameters</b>	<p><b>Channel</b></p> <p>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.</p> <p>For example, to use the last four ports as digital output channels, enter</p> <p>[5,6,7,8]</p> <p>Number the lines beginning with 1 even though the board manufacturer starts numbering the ports with 0.</p>

---

**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 Parameters** **Channel**  
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 a 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:

- 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 **Initial count**. The

# National Instruments PCI-6601 Incremental Encoder

---

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 required 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 required 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 required 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
```

# National Instruments PCI-6601 Event Counter

---

<b>Purpose</b>	PCI-6601 Event Counter block
<b>Library</b>	xPC Target Library for National Instruments
<b>Description</b>	The Event Counter is used for such tasks as measuring the frequency of a pulse train and counting binary events, such as switch closures and photocell interruptions (see “Examples” on page 38-427).
<b>Block Parameters</b>	<p><b>Channel</b></p> <p>From the list, choose one of the following:</p> <ul style="list-style-type: none"><li>• 1</li><li>• 2</li><li>• 3</li><li>• 4</li></ul> <p>These parameters specify the counter to be used with this driver block (hardware channels 0 to 3). In each case, one block is required for each counter.</p> <p><b>Counting Mode</b></p> <p>Select one of:</p> <ul style="list-style-type: none"><li>• <b>Cumulative counter</b></li></ul> <p>Accumulates count of edges of the source signal. Counter is reset according to <b>Reset Mode</b>. Block returns the current count with no latching.</p> <li>• <b>HW gated cumulative counter</b></li> <p>Accumulates count of edges of the source signal when hardware gate condition is true. Counter is reset according to <b>Reset Mode</b> (the gate end does <i>not</i> reset the count). Block returns the current count with no latching.</p>

# National Instruments PCI-6601 Event Counter

---

---

**Tip** Use **Gate Trigger Mode** values `Level Active High` or `Level Active Low` in this mode. Edge gating produces a simple cumulative counter that starts with the first edge after arming the counter.

---

- HW gated latched counter

Accumulates count of edges of the source signal when the hardware gate condition is true. Gate end latches the count and resets the count to the reset value for the next gate interval.

**Reset Mode** is ignored.

## Arm Control Input

Select this box to add arm input A to the block. If the arm input is high ( $> 0.5$ ), the counter is turned on. If the input is low ( $\leq 0.5$ ), the counter is turned off. The arm state is changed at the end of block execution and continues until the next time step.

The arm input serves as a software control for the counter, but does not latch the counter when it goes low. The block will output the most recently latched value if `HW gated latched counter` mode is also selected.

## Source

Select the counter input pin. `Default Source Pin` selects the pin documented in the 6601/6602 User Manual. The `PFI` values select the input pin for other counters. The `Timebase` values select either of two different standard frequencies.

Select one of:

- `Default Source Pin`
- `PFI 39 ( Pin 2 )`
- `PFI 35 ( Pin 7 )`
- `PFI 31 ( Pin 34 )`

# National Instruments PCI-6601 Event Counter

---

- PFI 27 ( Pin 31 )
- PFI 23 ( Pin 28 )
- PFI 19 ( Pin 25 )
- PFI 15 ( Pin 22 )
- PFI 11 ( Pin 52 )
- Timebase 1 (20 MHz)
- Timebase 2 (100 kHz)

## Source Polarity

Choose the signal edge that will increment the counter. Using the fastest edge will give the most accurate count. Select one of:

- Rising Edge
- Falling Edge

## Filter

Choose a filter to remove signal pulses shorter than a specified duration. High-frequency noise and slow signal edges can cause false counting. Select one of:

- None
- Synchronize input to Timebase 1 (20MHz)
- 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

## HW Gate Input

Select the PFI input pin to be used as gate when **Counting Mode** is set to a hardware gated mode. **Default Gate Pin** selects the pin shown in the 6601/6602 user manual. To gate two or more

# National Instruments PCI-6601 Event Counter

---

counters from the same signal, select the same input for all of the counters. Select one of:

- Default Source Pin
- Default Gate Pin
- PFI 38 ( Pin 3 )
- PFI 34 ( Pin 8 )
- PFI 30 ( Pin 67 )
- PFI 26 ( Pin 64 )
- PFI 22 ( Pin 61 )
- PFI 18 ( Pin 58 )
- PFI 14 ( Pin 21 )
- PFI 10 ( Pin 51 )

## Gate Trigger Mode

Select the trigger mode to be used when **Counting Mode** is set to a HW gated cumulative counter or HW gated latched counter. The choice of trigger edge depends on the rise time or fall time of the gate signal. Using the fastest edge will give the most accurate count. Select one of:

- Level Active High

Count source edges when the gate input is high. Latch the current count when the gate input goes from high to low and reload the counter with the reset value. The block outputs the most recently latched value.

- Level Active Low

Count source edges when the gate input is low. Latch the current count when the gate input goes from low to high and reload the counter with the reset value. The block outputs the most recently latched value.

- Rising Edge

Latch the current count on a rising edge, reload the counter with the reset value, and resume counting. The block outputs the most recently latched value.

- **Falling Edge**

Latch the current count on a falling edge, reload the counter with the reset value, and resume counting. The block outputs the most recently latched value.

## **Reset Mode**

Select the reset mode to be used when **Counting Mode** is set to Cumulative counter or HW gated cumulative counter.

- **No Reset**

Accumulate source input edges without resetting the counter to the reset value. Sample and return the accumulated value in the counter.

- **Reset after read**

Return the accumulated value in the counter and load the counter with the reset value.

- **SW Reset**

Select this value to add reset input R. During block execution, the block returns the accumulated value and checks input R. If R is nonzero, the block loads the counter with the reset value.

# National Instruments PCI-6601 Event Counter

---

---

## Note

- **Reset Mode** is ignored if **Counting Mode** is HW gated latched counter.
  - The counter value may become very large in No Reset mode. The value rolls over to 0 when it exceeds 0xFFFFFFFF (4294967295).
  - If R is always nonzero, SW Reset behaves like Reset after read mode.
- 

## Initial Load and Reset Value

Load the counter with this value on initial model load and again when the reset condition is met. When **Counting Mode** is HW gated latched counter, load the counter with this value when the gate turns off and latch the latest counter value for reading.

## Sample time

-1 to inherit or a time in seconds

## PCI slot (-1:autosearch)

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

1

to automatically locate the board.

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

```
getxpcpci
```

at the MATLAB prompt.

## Examples

You can configure this block in many different ways, including:

### Frequency Counter

Measures the frequency of a signal. Configure a second counter as a pulse generator and feed the pulse output to the gate input for this channel. Set the pulse generator H and L inputs to set your chosen measurement interval. Set the event counter block as follows:

<b>Block Parameter</b>	<b>Value</b>
Counting Mode	HW gated latched counter
Arm Control Input	Not checked
Source	Default Source Pin
Source Polarity	Rising Edge or Falling Edge (choose the cleaner edge)
Filter	None, or any choice faster than the signal being measured
HW Gate Input	Default Gate Pin, or any other if sharing a gate with another channel
Gate Trigger Mode	Rising Edge or Falling Edge. Because the pulse generator produces fast edges, rising and falling give the same result.
Reset Mode	Any (ignored)

---

**Note** If you use a **Gate Trigger Mode** of **Level Active High** or **Level Active Low** for a frequency counter, the gate must be both on and off for longer than the period of the signal being measured. If it is not on for long enough, it may not latch an edge. If it is not off for long enough, a source edge may not occur while the gate is off, the gate end condition will be missed, and the counter will add the next interval.

---

# National Instruments PCI-6601 Event Counter

---

## Events Since Last Execution

Counts the number of external binary events, such as switch closures and photocell interruptions, since the last sample time. Set the event counter block as follows:

<b>Block Parameter</b>	<b>Value</b>
<b>Counting Mode</b>	Cumulative counter
<b>Arm Control Input</b>	Checked or unchecked.  If checked, use arm input A in the model to control when to accept events.
<b>Source</b>	Default Source Pin, or any other source pin
<b>Source Polarity</b>	Rising Edge or Falling Edge (choose the cleaner edge)
<b>Filter</b>	None, or a pulse width large enough to eliminate switch bounce and similar artifacts
<b>HW Gate Input</b>	Any (ignored)
<b>Gate Trigger Mode</b>	Any (ignored)
<b>Reset Mode</b>	Reset after read  Use No Reset to create a running total of external events.  Use SW Reset and reset input R to reset the count according to some condition in your model.

**Purpose** Support for National Instruments PCI/PXI-6602 board

**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.

The xPC Target block library 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
- National Instruments PCI/PXI-6602 Event Counter

The xPC Target software does not support the interrupt or timer functionality of the board.

<b>Board Characteristics</b>	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

---

<b>Purpose</b>	PCI/PXI-6602 Digital Output block
<b>Library</b>	xPC Target Library for National Instruments
<b>Note</b>	Only one output block can exist in a model for any given board.
<b>Block Parameters</b>	<b>Channel</b> 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.

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

---

**Note** If you want to attach a 3-wire encoder, do so by connecting the channel A signal to a 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. See the Choose one of the following:

- Normal
- Quadrature Mode X1
- Quadrature Mode X2
- Quadrature Mode X4
- Two-pulse mode
- Synchronous Source Mode

This parameter specifies the counting mode for the board. See the National Instruments PCI/PXI-6602 user manual documentation for details and definitions of these modes.

**Initial count**

The initial count specifies the initial value for the counter. Enter a nonnegative integer.

# National Instruments PCI/PXI-6602 Incremental Encoder

---

## 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 **Initial count**. 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

# National Instruments PCI/PXI-6602 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
```

# 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 required 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 required 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 required 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/PXI-6602 Event Counter

---

<b>Purpose</b>	PCI-6602 Event Counter block
<b>Library</b>	xPC Target Library for National Instruments
<b>Description</b>	The Event Counter is used for such tasks as measuring the frequency of a pulse train and counting binary events, such as switch closures and photocell interruptions (see “Examples” on page 38-451).

## 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 (hardware channels 0 to 7). In each case, one block is required for each counter.

### Counting Mode

Select one of:

- Cumulative counter

Accumulates count of edges of the source signal. Counter is reset according to **Reset Mode**. Block returns the current count with no latching.

- HW gated cumulative counter

# National Instruments PCI/PXI-6602 Event Counter

---

Accumulates count of edges of the source signal when hardware gate condition is true. Counter is reset according to **Reset Mode** (the gate end does *not* reset the count). Block returns the current count with no latching.

---

**Tip** Use **Gate Trigger Mode** values `Level Active High` or `Level Active Low` in this mode. Edge gating produces a simple cumulative counter that starts with the first edge after arming the counter.

---

- HW gated latched counter

Accumulates count of edges of the source signal when the hardware gate condition is true. Gate end latches the count and resets the count to the reset value for the next gate interval. **Reset Mode** is ignored.

## Arm Control Input

Select this box to add arm input A to the block. If the arm input is high ( $> 0.5$ ), the counter is turned on. If the input is low ( $\leq 0.5$ ), the counter is turned off. The arm state is changed at the end of block execution and continues until the next time step.

The arm input serves as a software control for the counter, but does not latch the counter when it goes low. The block will output the most recently latched value if `HW gated latched counter` mode is also selected.

## Source

Select the counter input pin. `Default Source Pin` selects the pin documented in the 6601/6602 User Manual. The `PFI` values select the input pin for other counters. The `Timebase` values select any of three different standard frequencies.

- `Default Source Pin`
- `PFI 39 ( Pin 2 )`

# National Instruments PCI/PXI-6602 Event Counter

---

- PFI 35 ( Pin 7 )
- PFI 31 ( Pin 34 )
- PFI 27 ( Pin 31 )
- PFI 23 ( Pin 28 )
- PFI 19 ( Pin 25 )
- PFI 15 ( Pin 22 )
- PFI 11 ( Pin 52 )
- Timebase 1 (20.0 MHz)
- Timebase 2 (100 kHz)
- Timebase 3 (80 MHz)

## Source Polarity

Choose the signal edge that will increment the counter. Using the fastest edge will give the most accurate count. Select one of:

- Rising Edge
- Falling Edge

## Filter

Choose a filter to remove signal pulses shorter than a specified duration. High-frequency noise and slow signal edges can cause false counting. Select one of:

- None
- Synchronize input to Timebase 1 (20MHz)
- 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

# National Instruments PCI/PXI-6602 Event Counter

---

## HW Gate Input

Select the PFI input pin to be used as gate when **Counting Mode** is set to a hardware gated mode. **Default Gate Pin** selects the pin shown in the 6601/6602 user manual. To gate two or more counters from the same signal, select the same input for all of the counters. Select one of:

- Default Source Pin
- Default Gate Pin
- PFI 38 ( Pin 3 )
- PFI 34 ( Pin 8 )
- PFI 30 ( Pin 67 )
- PFI 26 ( Pin 64 )
- PFI 22 ( Pin 61 )
- PFI 18 ( Pin 58 )
- PFI 14 ( Pin 21 )
- PFI 10 ( Pin 51 )

## Gate Trigger Mode

Select the trigger mode to be used when **Counting Mode** is set to a HW gated cumulative counter or HW gated latched counter. The choice of trigger edge depends on the rise time or fall time of the gate signal. Using the fastest edge will give the most accurate count. Select one of:

- Level Active High

Count source edges when the gate input is high. Latch the current count when the gate input goes from high to low and reload the counter with the reset value. The block outputs the most recently latched value.

- Level Active Low

Count source edges when the gate input is low. Latch the current count when the gate input goes from low to high and reload the counter with the reset value. The block outputs the most recently latched value.

- **Rising Edge**

Latch the current count on a rising edge, reload the counter with the reset value, and resume counting. The block outputs the most recently latched value.

- **Falling Edge**

Latch the current count on a falling edge, reload the counter with the reset value, and resume counting. The block outputs the most recently latched value.

## **Reset Mode**

Select the reset mode to be used when **Counting Mode** is set to Cumulative counter or HW gated cumulative counter.

- **No Reset**

Accumulate source input edges without resetting the counter to the reset value. Sample and return the accumulated value in the counter.

- **Reset after read**

Return the accumulated value in the counter and load the counter with the reset value.

- **SW Reset**

Select this value to add reset input R. During block execution, the block returns the accumulated value and checks input R. If R is nonzero, the block loads the counter with the reset value.

# National Instruments PCI/PXI-6602 Event Counter

---

---

## Note

- **Reset Mode** is ignored if **Counting Mode** is HW gated latched counter.
  - The counter value may become very large in No Reset mode. The value rolls over to 0 when it exceeds 0xFFFFFFFF (4294967295).
  - If R is always nonzero, SW Reset behaves like Reset after read mode.
- 

## Initial Load and Reset Value

Load the counter with this value on initial model load and again when the reset condition is met. When **Counting Mode** is HW gated latched counter, load the counter with this value when the gate turns off and latch the latest counter value for reading.

## Sample time

-1 to inherit or a time in seconds

## PCI slot (-1:autosearch)

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

1

to automatically locate the board.

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

```
getxpcpci
```

at the MATLAB prompt.

## Examples

You can configure this block in many different ways, including:

### Frequency Counter

Measures the frequency of a signal. Configure a second counter as a pulse generator and feed the pulse output to the gate input for this channel. Set the pulse generator H and L inputs to set your chosen measurement interval. Set the event counter block as follows:

<b>Block Parameter</b>	<b>Value</b>
Counting Mode	HW gated latched counter
Arm Control Input	Not checked
Source	Default Source Pin
Source Polarity	Rising Edge or Falling Edge (choose the cleaner edge)
Filter	None, or any choice faster than the signal being measured
HW Gate Input	Default Gate Pin, or any other if sharing a gate with another channel
Gate Trigger Mode	Rising Edge or Falling Edge. Because the pulse generator produces fast edges, rising and falling give the same result.
Reset Mode	Any (ignored)

---

**Note** If you use a **Gate Trigger Mode** of **Level Active High** or **Level Active Low** for a frequency counter, the gate must be both on and off for longer than the period of the signal being measured. If it is not on for long enough, it may not latch an edge. If it is not off for long enough, a source edge may not occur while the gate is off, the gate end condition will be missed, and the counter will add the next interval.

---

# National Instruments PCI/PXI-6602 Event Counter

---

## Events Since Last Execution

Counts the number of external binary events, such as switch closures and photocell interruptions, since the last sample time. Set the event counter block as follows:

<b>Block Parameter</b>	<b>Value</b>
<b>Counting Mode</b>	Cumulative counter
<b>Arm Control Input</b>	Checked or unchecked.  If checked, use arm input A in the model to control when to accept events.
<b>Source</b>	Default Source Pin, or any other source pin
<b>Source Polarity</b>	Rising Edge or Falling Edge (choose the cleaner edge)
<b>Filter</b>	None, or a pulse width required to eliminate switch bounce and similar artifacts
<b>HW Gate Input</b>	Any (ignored)
<b>Gate Trigger Mode</b>	Any (ignored)
<b>Reset Mode</b>	Reset after read  Use No Reset to create a running total of external events.  Use SW Reset and reset input R to reset the count according to some condition in your model.

**Purpose** Support for National Instruments PCI-6703 board

**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.

The xPC Target block library supports this board with this driver block:

- National Instruments PCI-6703 Analog Output (D/A)

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
```

**Purpose** Support for National Instruments PCI-6704 board

**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.

The xPC Target block library supports this board with this driver block:

- National Instruments PCI-6704 Analog Output (D/A)

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.

<b>Board Characteristics</b>	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
	Milliamperes — 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
```

# National Instruments PCI/PXI-6711

---

**Purpose** Support for National Instruments PCI/PXI-6711 board

**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.

The xPC Target block library 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.

# National Instruments PCI/PXI-6711 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.

## **PCI Slot (-1:autosearch)**

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

-1

to automatically locate the board.

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of 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 required 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
```

# National Instruments PCI/PXI-6713

---

**Purpose** Support for National Instruments PCI/PXI-6713 board

**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.

The xPC Target block library 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.

## Scaling Input to Output

Hardware Output	Block Input Data Type	Scaling
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

## Scaling Input to Output

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

## 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.

# National Instruments PCI/PXI-6713 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.

## **PCI Slot (-1:autosearch)**

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

-1

to automatically locate the board.

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of 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 required 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.

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-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
```

# National Instruments PCI/PXI-6731

---

**Purpose** Support for National Instruments PCI/PXI-6731 board

**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.

The xPC Target block library 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

## 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 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

## 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 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.

# National Instruments PCI/PXI-6731 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.

## **PCI Slot (-1:autosearch)**

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

-1

to automatically locate the board.

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of 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 required 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.

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-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
```

# National Instruments PCI/PXI-6733

---

**Purpose** Support for National Instruments PCI/PXI-6733 board

**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.

The xPC Target block library 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 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-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

## 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 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.

# National Instruments PCI/PXI-6733 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.

## **PCI Slot (-1:autosearch)**

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

-1

to automatically locate the board.

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of 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 required 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.

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-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
```

# National Instruments PCI-DIO-96

---

**Purpose** Support for National Instruments PCI-DIO-96 board

**Board** National Instruments PCI-DIO-96

**General Description** The PC-DIO-96 is an I/O board with 96 digital input and output lines. The xPC Target block library 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

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 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 of 8 digital lines that can be configured as inputs or

# National Instruments PCI-DIO-96 Digital Input

---

outputs depending on which driver block is chosen. In each case, one block is required 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.

## **PCI Slot (-1:autosearch)**

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

-1

to automatically locate the board.

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of 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-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 an 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 required 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.

## **PCI Slot (-1:autosearch)**

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

-1

to automatically locate the board.

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of 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** Support for National Instruments PXI-6040E board

**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.

The xPC Target block library 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; Pulse Width/Period measurement: Yes; Pulse Train 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)

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.

If the board is set up for differential mode, you can read the data from either of the channels in the differential pair. For example, if you have a differential pair of 1 and 9, you can read the data from channel 1 or channel 9. However, you might want to read the lower channel number of the pair because it remains unchanged when you switch the input mode between single-ended and differential.

# National Instruments PXI-6040E Analog Input (A/D)

---

## **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 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]

## **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

<b>Scaling Input to Output</b>	<b>Hardware Output</b>	<b>Block Input 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 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.

## **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 required 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
```

**Purpose** Support for National Instruments PXI-6070E board

**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.

The xPC Target block library 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

<b>Board Characteristics</b>	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; Pulse Width/Period measurement: Yes; Pulse Train 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.

If the board is set up for differential mode, you can read the data from either of the channels in the differential pair. For example, if you have a differential pair of 1 and 9, you can read the data from channel 1 or channel 9. However, you might want to read the lower channel number of the pair because it remains unchanged when you switch the input mode between single-ended and differential.

# National Instruments PXI-6070E Analog Input (A/D)

---

## **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 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.

<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]

## **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

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 required 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
```

**Purpose** Support for National Instruments PXI-6071E board

**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.

The xPC Target block library 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; Pulse Width/Period measurement: Yes; Pulse Train 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)

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.

If the board is set up for differential mode, you can read the data from either of the channels in the differential pair. For example, if you have a differential pair of 1 and 9, you can read the data from channel 1 or channel 9. However, you might want to read the lower channel number of the pair because it remains unchanged when you switch the input mode between single-ended and differential.

# National Instruments PXI-6071E Analog Input (A/D)

---

## **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 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.

<b>Input Range (V)</b>	<b>Range Code</b>
-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 this driver block. Use the format [BusNumber, SlotNumber]. To determine the bus number and the PCI slot number, type

getxpci

# 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.

## **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 required 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
```

**Purpose** Support for National Instruments PXI-6508 board

**Board** National Instruments PXI-6508

**General Description** The PXI-6508 is an I/O board with 96 digital input and output lines. The xPC Target block library 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 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 an 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 required 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.

**PCI Slot (-1:autosearch)**

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

-1

to automatically locate the board.

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of 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 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 an 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 required 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.

## **PCI Slot (-1:autosearch)**

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

-1

to automatically locate the board.

If two or more boards of this type are in the target PC, enter the bus number and the PCI slot number of 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-6527

---

**Purpose** Support for National Instruments PXI-6527 board

**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.

The xPC Target block library 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 required 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

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 required 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
```

# National Instruments PXI-6704

---

**Purpose** Support for National Instruments PXI-6704 board

**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.

The xPC Target block library 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

# National Instruments PXI-6704 Analog Output (D/A)

---

# North Atlantic Industries, Inc.

---

This topic describes the North Atlantic Industries, Inc. (NAII, formerly Apex) I/O boards supported by the xPC Target product (<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  
39-3

Description of NAII driver blocks.

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

# NAII PC-12SD (PC-77SD1)

---

**Purpose** Support for NAII PC-12SD (PC-77SD1) positioning sensor I/O board-

**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.

The xPC Target block library 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

## Scaling Input to Output

<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 from a channel. This vector has to have the same length as the **Channel vector**. Choose values to give the most accurate velocity readings.

## **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

---

**Purpose** Support for NAII (Apex) 73LD3 positioning sensor I/O board-

**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.

The xPC Target block library supports this board with this driver:

- NAII 73LD3 LVDT/RVDT Converter

The xPC Target product does not support the digital I/O functionality of this board.

<b>Board Characteristics</b>	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 1
  - Signal status, weight of 2
  - Reference status, weight of 4

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.

# NAII 73LD3 LVDT/RVDT Converter

---

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

---

**Purpose** Support for NAII (Apex) 73SD3 positioning sensor I/O board-

**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.

The xPC Target block library 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 1
  - Signal status, weight of 2
  - Reference status, weight of 4

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 latch all channels before reading them. All channels are sampled at 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

**Purpose** Support for NAII (Apex) 76LD1 positioning sensor I/O board-

**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.

The xPC Target block library supports this board with one driver block:

- NAII 76LD1 L/D

<b>Board Characteristics</b>	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** NAII 76LD1 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 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 1
  - Signal status, weight of 2
  - Reference status, weight of 4

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 latch all channels before reading them. All channels are sampled at 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 computer, enter

-1

to automatically locate the board.

If two or more boards of this type are in the target computer, enter the bus number and the PCI slot number of 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.

**Purpose** Support for NAII (Apex) 76CL1 positioning sensor I/O board-

**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.

The xPC Target block library 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

---

**Purpose** Support for NAI 76CL1 L/D positioning sensor I/O board-

**Board** NAI 76CL1 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 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 1
  - Signal status, weight of 2
  - Reference status, weight of 4

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 latch all channels before reading them. All channels are sampled at the same time. Note that selecting this option incurs a small additional resource overhead.

**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 computer, enter

-1

to automatically locate the board.

If two or more boards of this type are in the target computer, enter the bus number and the PCI slot number of 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 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 only the A subchannels (B is implied) in **Channel vector** as *n*. The block input expects a vector that contains both channels. For example, if all channels are configured as 2-wire, the **Channel vector** value

[1, 2, 3, 4, 5, 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

If a channel is 2-wire, you can specify the A subchannels (B is implied) (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 channels 5A and 5B. 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 5AB when you click **Apply** or **OK**.

## 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 computer, enter

-1

to automatically locate the board.

If two or more boards of this type are in the target computer, enter the bus number and the PCI slot number of 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 76CL1 board is 7651 in hexadecimal.

# NAII (Apex) 76CS1

---

**Purpose** Support for NAII (Apex) 76CS1 positioning sensor I/O board-

**Board** NAII (Apex) 76CS1

**General 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.

The xPC Target block library supports this board with two driver blocks:

- NAII 76CS1 S/D
- NAII 76CS1 D/S

<b>Board Characteristics</b>	Board name	76CS1
	Manufacturer	NAII
	Bus type	PCI
	Access method	Memory mapped
	Multiple block instance support	No
	Multiple board support	Yes

**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 1
  - Signal status, weight of 2
  - Reference status, weight of 4

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 latch all channels before reading them. All channels are sampled at 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 computer screen.

## **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 computer, enter

-1

to automatically locate the board.

If two or more boards of this type are in the target computer, enter the bus number and the PCI slot number of 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.

# NAII 76CS1 D/S

---

**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 computer, enter

-1

to automatically locate the board.

If two or more boards of this type are in the target computer, enter the bus number and the PCI slot number of 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 topic describes the Quanser I/O board supported by the xPC Target product (<http://www.quanser.com>).

## Quanser Q4

Multifunction measurement and control board with four 14-bit single-ended A/D converters, four 12-bit D/A voltage outputs, 16 digital communication channels, four 24-bit encoder inputs, and two 32-bit counter/timers.

## 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 – Alphabetical list**

**Purpose** Support for the Quanser Q4 multifunction measurement and control board.

**Board** Quanser Q4

**General Description** The Quanser Q4 is a multifunction measurement and control board. It has four 14-bit singled-ended A/D converters, four 12-bit D/A voltage outputs, 16 digital communication channels (each of which can be used for digital input or digital output), four 24-bit encoder inputs, and two 32-bit counter/timers.

The xPC Target block library supports this board with these driver blocks:

- Quanser Q4 Analog Input
- Quanser Q4 Analog Output
- Quanser Q4 Digital Input
- Quanser Q4 Digital Output
- Quanser Q4 Incremental Encoder
- Quanser Q4 Counter

## Board Characteristics

Board name	Q4
Manufacturer	Quanser
Bus type	PCI
Access method	Memory mapped
Multiple block instance support	No
Multiple board support	Yes

# Quanser Q4 Analog Input

---

**Purpose** Q4 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 4. This driver allows you to enter channel numbers in any order. For example, to use the first, second and fourth channels, enter

[1,2,4]

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.

### 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
```

**Purpose** Q4 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 4. This driver allows you to enter channel numbers in any order. For example, to use the first, second, and fourth channels, enter

[1,2,4]

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 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

# Quanser Q4 Analog Output

---

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 set the outputs to update simultaneously (nontransparent 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.

## **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
```

# Quanser Q4 Digital Input

---

**Purpose** Q4 Digital Input block

**Library** xPC Target Library for Quanser

## Scaling of 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 between 1 and 16. 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 Q4 board has 16 digital channels. You can use each of these digital channels for input or output.

---

**Note** If you have a Q4 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.

## 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
```

# Quanser Q4 Digital Output

---

**Purpose** Q4 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 16. 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 Q4 board has 16 digital channels. You can use each of these digital channels for input or output.

---

**Note** If you have a Q4 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.

## **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
```

# Quanser Q4 Incremental Encoder

---

**Purpose** Q4 Incremental Encoder block

**Library** xPC Target Library for Quanser

**Block Parameters** **Channel vector**

Enter a vector of numbers between 1 and 4. This driver allows you to enter channel numbers in any order. For example, to use the first two odd encoder input channels, enter

[1,3]

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.

---

## **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.

## 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

# Quanser Q4 Incremental Encoder

---

- 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 Q4 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
```

# Quanser Q4 Counter

---

## Purpose

Q4 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.

# Quanser Q4 Counter

---

## **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
```

**Purpose** Support for the Quanser Q8 multifunction measurement and control board.

**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.

The xPC Target block library 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 set all outputs to update simultaneously (nontransparent 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

Hardware Input	Block Output Data Type	Scaling
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.

---

**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.

## 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.

# Quanser Q8 Counter

---

# Real Time Devices

---

This topic describes Real Time Devices I/O boards supported by the xPC Target product (<http://www.rtd.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.
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 41-3

Description of block parameters for Real Time Devices driver blocks.

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

# Real Time Devices DM6420

---

**Purpose** Support for the RTD DM6420 I/O board.

**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).

The xPC Target block library 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** The xPC Target software 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.

# Real Time Devices DM6420 Analog Output (D/A)

---

Input Range (V)	Range Code
-5 to +5	-5
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 Real Time Devices

**Note** The DAS1601/16 has an 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 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. One block is required for each port 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

# Real Time Devices DM6420 Digital Output

---

**Purpose** DM6420 Digital Output block

**Library** xPC Target Library for Real Time Devices

**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. One block is required for each port used.

# 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

---

**Purpose** Support for the RTD DM6430 I/O board.

**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).

The xPC Target block library 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** The xPC Target software 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 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.

### **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
4	-2.5 to 2.5
8	-1.25 to 1.25

# Real Time Devices DM6430 Analog Input (A/D)

---

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

```
[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.

## **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 Analog Output (D/A)

**Purpose** DM6430 Analog Output block

**Library** xPC Target Library for Real Time Devices

<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 Real Time Devices

**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. One block is required for each port used.

# 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 Real Time Devices

**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 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. One block is required for each port used.

# 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

**Purpose** Support for the RTD DM6604 I/O board.

**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.

The xPC Target block library 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

<b>Board Characteristics</b>	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 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
-10 to +10	-10
-5 to +5	-5
0 to +10	10
0 to +5	5

# Real Time Devices DM6604 Analog Output (D/A)

---

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

# Real Time Devices DM6604 Digital Input

---

**Purpose** DM6604 Digital Input block

**Library** xPC Target Library for Real Time Devices

**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. One block is required for each port 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
```

# Real Time Devices DM6604 Digital Output

---

**Purpose** DM6604 Digital Output block

**Library** xPC Target Library for Real Time Devices

## **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. One block is required for each port 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 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

**Purpose** Support for the RTD DM6804 I/O board.

**Board** Real Time Devices DM6804

**General Description** The DM6804 is an ISA PC/104 I/O board with 24 digital I/O lines. It contains one 8255 chip with 3 digital I/O ports. The xPC Target block library supports this board with these driver blocks:

- Real Time Devices DM6804 Digital Input
- Real Time Devices DM6804 Digital Output

<b>Board Characteristics</b>	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 Real Time Devices

**Note** The DM6804 has an 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. One block is required for each port 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
```

# Real Time Devices DM6804 Digital Output

---

**Purpose** DM6804 Digital Output block

**Library** xPC Target Library for Real Time Devices

**Note** The DM6804 has an 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. One block is required for each port 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.

# 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

**Purpose** Support for the RTD DM6814 I/O board.

**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.

The xPC Target block library 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** The xPC Target software 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 Real Time Devices

<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 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 Real Time Devices

**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.

## 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

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 Real Time Devices

**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.

## 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 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

**Purpose** Support for the RTD DM6816 I/O board.

**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.

The xPC Target block library 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

# Real Time Devices DM6816 PWM

---

**Purpose** DM6816 PWM block

**Library** xPC Target Library for Real Time Devices

**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.

## **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

# Real Time Devices DM7420

---

**Purpose** Support for the RTD DM7420 I/O board.

**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.

The xPC Target block library 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** The xPC Target software 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 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.

### 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

# 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 Real Time Devices

## **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 Real Time Devices

**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

# Sensoray

---

This topic describes the Sensoray I/O boards supported by the xPC Target product (<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.

“Boards and Blocks — Alphabetical List” on page 42-2

Description of block parameters for Sensoray driver blocks.

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

**Purpose** Support for the Sensoray 526 measurement and control board

**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.

The xPC Target software supports this board with these driver blocks:

- Sensoray526 AD
- Sensoray526 Dual AD
- Sensoray526 DA
- Sensoray526 Dual DA
- Sensoray526 DI
- Sensoray526 DO
- Sensoray526 Encoder Input

<b>Board Characteristics</b>	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 (i.e. 0x2c0)

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** Sensoray526 DA block

**Library** xPC Target Library for Sensoray

**Scaling of Input to Output**

Hardware Output	Block Input Data Type	Scaling
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.

# Sensoray526 Dual DA

---

## **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**

Hardware Input	Block Output Data Type	Scaling
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 $\geq$ 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.

# Sensaray526 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.

# Sensoray526 Encoder Input

---

# Speedgoat

---

This topic describes the Speedgoat (<http://www.speedgoat.ch>) FPGA I/O boards supported by xPC Target.

See “Workflow” for a description of the development workflow for FPGA I/O boards. You do not use these blocks outside of HDL Coder™ HDL Workflow Advisor.

---

**Note** These Speedgoat FPGA I/O board block descriptions are for informational purposes only. The HDL Coder HDL Workflow Advisor uses these blocks to generate an xPC Target interface subsystem. The subsystem mask controls all of the block parameters.

---

Speedgoat I/O FPGA boards are sold as part of xPC Target Turnkey systems. For xPC Target Turnkey hardware, see <http://www.mathworks.com/products/xpctarget/supported-hardware/index.html>.

## Speedgoat IO301

The Speedgoat IO301 is a field programmable gate array (FPGA) board that provides 64 bidirectional TTL I/O lines. This board is based on a Xilinx® Virtex-II chip with 6912 logic cells.

## Speedgoat IO302

The Speedgoat IO302 is a field programmable gate array (FPGA) board that provides 32 bidirectional RS-422 I/O lines. This board is based on a Xilinx Virtex-II chip with 6912 logic cells.

Speedgoat IO303

The Speedgoat IO303 is a field programmable gate array (FPGA) board that provides 16 bidirectional TTL and 24 bidirectional RS-422 I/O lines. This board is based on a Xilinx Virtex-II chip with 6912 logic cells.

Speedgoat IO311

The Speedgoat IO311 is a field programmable gate array (FPGA) board that provides 64 bidirectional TTL I/O lines. This board is based on a Xilinx Virtex-II chip with 24192 logic cells.

Speedgoat IO312

The Speedgoat IO312 is a field programmable gate array (FPGA) board that provides 32 bidirectional RS-422 I/O lines. This board is based on a Xilinx Virtex-II chip with 24192 logic cells.

Speedgoat IO313

The Speedgoat IO313 is a field programmable gate array (FPGA) board that provides 16 TTL and 24 RS-422 I/O lines. This board is based on a Xilinx Virtex-II chip with 24192 logic cells.

Speedgoat IO314

The Speedgoat IO314 is a field programmable gate array (FPGA) board that provides 32 bidirectional LVDS I/O lines. This board is based on a Xilinx Virtex-II chip with 24192 logic cells.

---

Speedgoat IO325

The Speedgoat IO325 is a field programmable gate array (FPGA) board that provides 64 bidirectional LVC MOS or 32 bidirectional LVDS (four are input only) I/O lines and two 16-bit 105 MHz analog input channels. This board is based on a Xilinx Virtex-4 chip with 41472 logic cells.

“Blocks — Alphabetical List” on page 43-4

List of blocks for Speedgoat FPGA I/O boards.

## **Blocks – Alphabetical List**

**Purpose** Support for the Speedgoat IO301 FPGA board

**Board** Speedgoat IO301

**General Description** The Speedgoat IO301 is a field programmable gate array (FPGA) board that provides 64 bidirectional TTL I/O lines. This board is based on a Xilinx Virtex-II chip with 6912 logic cells. Design tools supporting this chip include the freeware Xilinx ISE Webpack (<http://www.xilinx.com/tools/webpack.htm>) and the Xilinx ISE Design Suite (<http://www.xilinx.com/tools/designtools.htm>), version 10.1 required.

The clock rate for this board is 33 MHz.

The xPC Target block library supports this board with these driver blocks:

- Speedgoat IO301 PCI Read
- Speedgoat IO301 PCI Setup
- Speedgoat IO301 PCI Write

---

**Note** The Speedgoat FPGA I/O board block descriptions in this section are for informational purposes only. The HDL Coder HDL Workflow Advisor uses these blocks to generate an xPC Target interface subsystem. The subsystem mask controls all of the block parameters. Typically, you do not need to directly control these blocks.

---

The Speedgoat IO301 68 Pin Connector Map on page 43-5 lists the I/O channel to connector pin assignments.

### Speedgoat IO301 68 Pin Connector Map

Connector Pin	I/O Channel	Direction	Transceiver
1	0	Input/Output	TTL
2	1		

# Speedgoat IO301

**Speedgoat IO301 68 Pin Connector Map (Continued)**

Connector Pin	I/O Channel	Direction	Transceiver
3	2	Input/Output	TTL
4	3		
5	4	Input/Output	TTL
6	5		
7	6	Input/Output	TTL
8	7		
9	8	Input/Output	TTL
10	9		
11	10	Input/Output	TTL
12	11		
13	12	Input/Output	TTL
14	13		
15	14	Input/Output	TTL
16	15		
17	Ground		
18	Ground		
19	16	Input/Output	TTL
20	17		
21	18	Input/Output	TTL
22	19		
23	20	Input/Output	TTL
24	21		

## Speedgoat IO301 68 Pin Connector Map (Continued)

Connector Pin	I/O Channel	Direction	Transceiver
25	22	Input/Output	TTL
26	23		
27	24	Input/Output	TTL
28	25		
29	26	Input/Output	TTL
30	27		
31	28	Input/Output	TTL
32	29		
33	30	Input/Output	TTL
34	31		
35	Ground		
36	Ground		
37	32	Input/Output	TTL
38	33		
39	34	Input/Output	TTL
40	35		
41	36	Input/Output	TTL
42	37		
43	38	Input/Output	TTL
44	39		
45	40	Input/Output	TTL
46	41		

# Speedgoat IO301

**Speedgoat IO301 68 Pin Connector Map (Continued)**

Connector Pin	I/O Channel	Direction	Transceiver
47	42	Input/Output	TTL
48	43		
49	44	Input/Output	TTL
50	45		
51	46	Input/Output	TTL
52	47		
53	48	Input/Output	TTL
54	49		
55	50	Input/Output	TTL
56	51		
57	52	Input/Output	TTL
58	53		
59	54	Input/Output	TTL
60	55		
61	56	Input/Output	TTL
62	57		
63	58	Input/Output	TTL
64	59		
65	60	Input/Output	TTL
66	61		
67	62	Input/Output	TTL
68	63		

## **Board Characteristics**

Board name	IO301
Clock rate	33 MHz
Manufacturer	Speedgoat
Bus type	PCI
Multiple block instance support	Yes
Multiple board support	Yes

# Speedgoat IO301 PCI Read

---

**Purpose** Speedgoat IO301 PCI Read block

**Description** The Speedgoat IO301 PCI Read block reads a 32-bit data value from a PCI port offset address. Output is a `uint32` data type.

**Library** xPC Target Library for Speedgoat

## Block Parameters

---

**Note** The Speedgoat FPGA I/O board block descriptions in this section are for informational purposes only. The HDL Coder HDL Workflow Advisor uses these blocks to generate an xPC Target interface subsystem. The subsystem mask controls all of the block parameters. Typically, you do not need to directly control these blocks.

---

### Device index

Enter a number between 0 and 7. The maximum number of allowed boards is 8.

### Port name

Enter the FPGA output port name.

### Port offset

Enter the PCI register offset address for the ports being read.

Valid offset addresses begin at `0x8000`. Each PCI address must be unique and 32-bit aligned, such as `hex2dec('8000')`, `hex2dec('8004')`, and so on.

### Port type

Enter the data type of the ports being read (`uint32` (fixed)).

### Port dimension

Enter the dimension of the ports being read (*number of addresses to read*). If **Port dimension** is greater than 1, you must use a strobe.

## **Strobe offset**

Enter the PCI register offset address for the signal used to latch data from the FPGA into the ports being read. If **Strobe offset** is 0, no strobe is used.

Valid offset addresses begin at 0x8000. Each PCI address must be unique and 32-bit aligned, such as `hex2dec('8000')`, `hex2dec('8004')`, and so on.

## **Sample time**

Enter a base sample time or a multiple of the base sample time (-1 means sample time is inherited).

# Speedgoat IO301 PCI Setup

---

**Purpose** Speedgoat IO301 PCI Setup block

**Description** The Speedgoat IO301 PCI Setup block performs the setup and configuration required to use the IO301 board.

**Library** xPC Target Library for Speedgoat

## Block Parameters

---

**Note** The Speedgoat FPGA I/O board block descriptions in this section are for informational purposes only. The HDL Coder HDL Workflow Advisor uses these blocks to generate an xPC Target interface subsystem. The subsystem mask controls all of the block parameters. Typically, you do not need to directly control these blocks.

---

### FPGA model/subsystem name

Enter a string used to identify the model and subsystem containing the Setup block.

### Run FPGA only when xPC Target runs

Select this check box to enable the FPGA to execute only when the target application is running.

Clear this check box to enable the FPGA to start immediately after the target application is loaded.

### Reset FPGA states on start

Select this check box to cause the FPGA to reset the initial state of all variables and registers to the default state on start. Takes effect only when the check box **Run FPGA only when xPC Target runs** is selected.

Clear this check box to allow the FPGA to retain the end state of the previous execution. To return to the initial state, you must download the model to the target again and restart the simulation.

The default setting is Selected.

## **Device index**

Enter a number between 0 and 7. The maximum number of allowed boards is 8.

## **PCI slot (-1:autosearch)**

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

-1

to automatically locate the board.

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

```
getxpcpci
```

# Speedgoat IO301 PCI Write

---

**Purpose** Speedgoat IO301 PCI Write block

**Description** The Speedgoat IO301 PCI Write block writes a 32-bit data value to a PCI port offset address. Input must be a uint32 data type.

**Library** xPC Target Library for Speedgoat

## Block Parameters

---

**Note** The Speedgoat FPGA I/O board block descriptions in this section are for informational purposes only. The HDL Coder HDL Workflow Advisor uses these blocks to generate an xPC Target interface subsystem. The subsystem mask controls all of the block parameters. Typically, you do not need to directly control these blocks.

---

### Device index

Enter a number between 0 and 7. The maximum number of allowed boards is 8.

### Port name

Enter the FPGA input port name.

### Port offset

Enter the PCI register offset address for the ports being written.

Valid offset addresses begin at 0x8000. Each PCI address must be unique and 32-bit aligned, such as `hex2dec('8000')`, `hex2dec('8004')`, and so on.

### Port type

Enter the data type of the ports being written (uint32 (fixed)).

### Port dimension

Enter the dimension of the ports being written (*number of addresses to write*). If **Port dimension** is greater than 1, you must use a strobe.

## **Strobe offset**

Enter the PCI register offset address for the signal used to latch data into the FPGA from the ports being written. If **Strobe offset** is 0, no strobe is used.

Valid offset addresses begin at 0x8000. Each PCI address must be unique and 32-bit aligned, such as `hex2dec('8000')`, `hex2dec('8004')`, and so on.

## **Sample time**

Enter a base sample time or a multiple of the base sample time (-1 means sample time is inherited).

# Speedgoat IO302

---

**Purpose** Support for the Speedgoat IO302 FPGA board

**Board** Speedgoat IO302

**General Description** The Speedgoat IO302 is a field programmable gate array (FPGA) board that provides 32 bidirectional RS-422 I/O lines. This board is based on a Xilinx Virtex-II chip with 6912 logic cells. Design tools supporting this chip include the freeware Xilinx ISE Webpack (<http://www.xilinx.com/tools/webpack.htm>) and the Xilinx ISE Design Suite (<http://www.xilinx.com/tools/designtools.htm>), version 10.1 required.

The clock rate for this board is 33 MHz.

The xPC Target block library supports this board with these driver blocks:

- Speedgoat IO302 PCI Read
- Speedgoat IO302 PCI Setup
- Speedgoat IO302 PCI Write

---

**Note** The Speedgoat FPGA I/O board block descriptions in this section are for informational purposes only. The HDL Coder HDL Workflow Advisor uses these blocks to generate an xPC Target interface subsystem. The subsystem mask controls all of the block parameters. Typically, you do not need to directly control these blocks.

---

The Speedgoat IO302 68 Pin Connector Map on page 43-17 lists the I/O channel to connector pin assignments.

## Speedgoat IO302 68 Pin Connector Map

Connector Pin	I/O Channel	Direction	Transceiver
1	0(+)	Input/Output	RS-422
2	0(-)		
3	1(+)	Input/Output	RS-422
4	1(-)		
5	2(+)	Input/Output	RS-422
6	2(-)		
7	3(+)	Input/Output	RS-422
8	3(-)		
9	4(+)	Input/Output	RS-422
10	4(-)		
11	5(+)	Input/Output	RS-422
12	5(-)		
13	6(+)	Input/Output	RS-422
14	6(-)		
15	7(+)	Input/Output	RS-422
16	7(-)		
17	Ground		
18	Ground		
19	8(+)	Input/Output	RS-422
20	8(-)		
21	9(+)	Input/Output	RS-422
22	9(-)		

# Speedgoat IO302

**Speedgoat IO302 68 Pin Connector Map (Continued)**

Connector Pin	I/O Channel	Direction	Transceiver
23	10(+)	Input/Output	RS-422
24	10(-)		
25	11(+)	Input/Output	RS-422
26	11(-)		
27	12(+)	Input/Output	RS-422
28	12(-)		
29	13(+)	Input/Output	RS-422
30	13(-)		
31	14(+)	Input/Output	RS-422
32	14(-)		
33	15(+)	Input/Output	RS-422
34	15(-)		
35	Ground		
36	Ground		
37	16(+)	Input/Output	RS-422
38	16(-)		
39	17(+)	Input/Output	RS-422
40	17(-)		
41	18(+)	Input/Output	RS-422
42	18(-)		
43	19(+)	Input/Output	RS-422
44	19(-)		

## Speedgoat IO302 68 Pin Connector Map (Continued)

Connector Pin	I/O Channel	Direction	Transceiver
45	20(+)	Input/Output	RS-422
46	20(-)		
47	21(+)	Input/Output	RS-422
48	21(-)		
49	22(+)	Input/Output	RS-422
50	22(-)		
51	23(+)	Input/Output	RS-422
52	23(-)		
53	24(+)	Input/Output	RS-422
54	24(-)		
55	25(+)	Input/Output	RS-422
56	25(-)		
57	26(+)	Input/Output	RS-422
58	26(-)		
59	27(+)	Input/Output	RS-422
60	27(-)		
61	28(+)	Input/Output	RS-422
62	28(-)		
63	29(+)	Input/Output	RS-422
64	29(-)		
65	30(+)	Input/Output	RS-422
66	30(-)		

# Speedgoat IO302

---

## Speedgoat IO302 68 Pin Connector Map (Continued)

Connector Pin	I/O Channel	Direction	Transceiver
67	31(+)	Input/Output	RS-422
68	31(-)		

### Board Characteristics

Board name	IO302
Clock rate	33 MHz
Manufacturer	Speedgoat
Bus type	PCI
Multiple block instance support	Yes
Multiple board support	Yes

**Purpose** Speedgoat IO302 PCI Read block

**Description** The Speedgoat IO302 PCI Read block reads a 32-bit data value from a PCI port offset address. Output is a `uint32` data type.

**Library** xPC Target Library for Speedgoat

## Block Parameters

---

**Note** The Speedgoat FPGA I/O board block descriptions in this section are for informational purposes only. The HDL Coder HDL Workflow Advisor uses these blocks to generate an xPC Target interface subsystem. The subsystem mask controls all of the block parameters. Typically, you do not need to directly control these blocks.

---

### Device index

Enter a number between 0 and 7. The maximum number of allowed boards is 8.

### Port name

Enter the FPGA output port name.

### Port offset

Enter the PCI register offset address for the ports being read.

Valid offset addresses begin at 0x8000. Each PCI address must be unique and 32-bit aligned, such as `hex2dec('8000')`, `hex2dec('8004')`, and so on.

### Port type

Enter the data type of the ports being read (`uint32` (fixed)).

### Port dimension

Enter the dimension of the ports being read (*number of addresses to read*). If **Port dimension** is greater than 1, you must use a strobe.

# Speedgoat IO302 PCI Read

---

## **Strobe offset**

Enter the PCI register offset address for the signal used to latch data from the FPGA into the ports being read. If **Strobe offset** is 0, no strobe is used.

Valid offset addresses begin at 0x8000. Each PCI address must be unique and 32-bit aligned, such as `hex2dec('8000')`, `hex2dec('8004')`, and so on.

## **Sample time**

Enter a base sample time or a multiple of the base sample time (-1 means sample time is inherited).

**Purpose** Speedgoat IO302 PCI Setup block

**Description** The Speedgoat IO302 PCI Setup block performs the setup and configuration required to use the IO302 board.

**Library** xPC Target Library for Speedgoat

## Block Parameters

---

**Note** The Speedgoat FPGA I/O board block descriptions in this section are for informational purposes only. The HDL Coder HDL Workflow Advisor uses these blocks to generate an xPC Target interface subsystem. The subsystem mask controls all of the block parameters. Typically, you do not need to directly control these blocks.

---

### FPGA model/subsystem name

Enter a string used to identify the model and subsystem containing the Setup block.

### Run FPGA only when xPC Target runs

Select this check box to enable the FPGA to execute only when the target application is running.

Clear this check box to enable the FPGA to start immediately after the target application is loaded.

### Reset FPGA states on start

Select this check box to cause the FPGA to reset the initial state of all variables and registers to the default state on start. Takes effect only when the check box **Run FPGA only when xPC Target runs** is selected.

Clear this check box to allow the FPGA to retain the end state of the previous execution. To return to the initial state, you must download the model to the target again and restart the simulation.

The default setting is Selected.

# Speedgoat IO302 PCI Setup

---

## **Device index**

Enter a number between 0 and 7. The maximum number of allowed boards is 8.

## **PCI slot (-1:autosearch)**

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

-1

to automatically locate the board.

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

```
getxpcpci
```

<b>Purpose</b>	Speedgoat IO302 PCI Write block
<b>Description</b>	The Speedgoat IO302 PCI Write block writes a 32-bit data value to a PCI port offset address. Input must be a uint32 data type.
<b>Library</b>	xPC Target Library for Speedgoat

## Block Parameters

---

**Note** The Speedgoat FPGA I/O board block descriptions in this section are for informational purposes only. The HDL Coder HDL Workflow Advisor uses these blocks to generate an xPC Target interface subsystem. The subsystem mask controls all of the block parameters. Typically, you do not need to directly control these blocks.

---

### Device index

Enter a number between 0 and 7. The maximum number of allowed boards is 8.

### Port name

Enter the FPGA input port name.

### Port offset

Enter the PCI register offset address for the ports being written.

Valid offset addresses begin at 0x8000. Each PCI address must be unique and 32-bit aligned, such as `hex2dec('8000')`, `hex2dec('8004')`, and so on.

### Port type

Enter the data type of the ports being written (uint32 (fixed)).

### Port dimension

Enter the dimension of the ports being written (*number of addresses to write*). If **Port dimension** is greater than 1, you must use a strobe.

# Speedgoat IO302 PCI Write

---

## **Strobe offset**

Enter the PCI register offset address for the signal used to latch data into the FPGA from the ports being written. If **Strobe offset** is 0, no strobe is used.

Valid offset addresses begin at 0x8000. Each PCI address must be unique and 32-bit aligned, such as `hex2dec('8000')`, `hex2dec('8004')`, and so on.

## **Sample time**

Enter a base sample time or a multiple of the base sample time (-1 means sample time is inherited).

**Purpose** Support for the Speedgoat IO303 FPGA board

**Board** Speedgoat IO303

**General  
Description**

The Speedgoat IO303 is a field programmable gate array (FPGA) board that provides 16 bidirectional TTL and 24 bidirectional RS-422 I/O lines. This board is based on a Xilinx Virtex-II chip with 6912 logic cells. Design tools supporting this chip include the freeware Xilinx ISE Webpack (<http://www.xilinx.com/tools/webpack.htm>) and the Xilinx ISE Design Suite (<http://www.xilinx.com/tools/designtools.htm>), version 10.1 required.

The clock rate for this board is 33 MHz.

The xPC Target block library supports this board with these driver blocks:

- Speedgoat IO303 PCI Read
- Speedgoat IO303 PCI Setup
- Speedgoat IO303 PCI Write

---

**Note** The Speedgoat FPGA I/O board block descriptions in this section are for informational purposes only. The HDL Coder HDL Workflow Advisor uses these blocks to generate an xPC Target interface subsystem. The subsystem mask controls all of the block parameters. Typically, you do not need to directly control these blocks.

---

The Speedgoat IO303 68 Pin Connector Map on page 43-28 lists the I/O channel to connector pin assignments.

# Speedgoat IO303

## Speedgoat IO303 68 Pin Connector Map

Connector Pin	I/O Channel	Direction	Transceiver
1	0	Input/Output	TTL
2	1		
3	2	Input/Output	TTL
4	3		
5	4	Input/Output	TTL
6	5		
7	6	Input/Output	TTL
8	7		
9	8	Input/Output	TTL
10	9		
11	10	Input/Output	TTL
12	11		
13	12	Input/Output	TTL
14	13		
15	14	Input/Output	TTL
16	15		
17	Ground		
18	Ground		
19	0(+)	Input/Output	RS-422
20	0(-)		
21	1(+)	Input/Output	RS-422
22	1(-)		

## Speedgoat IO303 68 Pin Connector Map (Continued)

Connector Pin	I/O Channel	Direction	Transceiver
23	2(+)	Input/Output	RS-422
24	2(-)		
25	3(+)	Input/Output	RS-422
26	3(-)		
27	4(+)	Input/Output	RS-422
28	4(-)		
29	5(+)	Input/Output	RS-422
30	5(-)		
31	6(+)	Input/Output	RS-422
32	6(-)		
33	7(+)	Input/Output	RS-422
34	7(-)		
35	Ground		
36	Ground		
37	8(+)	Input/Output	RS-422
38	8(-)		
39	9(+)	Input/Output	RS-422
40	9(-)		
41	10(+)	Input/Output	RS-422
42	10(-)		
43	11(+)	Input/Output	RS-422
44	11(-)		

# Speedgoat IO303

**Speedgoat IO303 68 Pin Connector Map (Continued)**

Connector Pin	I/O Channel	Direction	Transceiver
45	12(+)	Input/Output	RS-422
46	12(-)		
47	13(+)	Input/Output	RS-422
48	13(-)		
49	14(+)	Input/Output	RS-422
50	14(-)		
51	15(+)	Input/Output	RS-422
52	15(-)		
53	16(+)	Input/Output	RS-422
54	16(-)		
55	17(+)	Input/Output	RS-422
56	17(-)		
57	18(+)	Input/Output	RS-422
58	18(-)		
59	19(+)	Input/Output	RS-422
60	19(-)		
61	20(+)	Input/Output	RS-422
62	20(-)		
63	21(+)	Input/Output	RS-422
64	21(-)		
65	22(+)	Input/Output	RS-422
66	22(-)		

## Speedgoat IO303 68 Pin Connector Map (Continued)

Connector Pin	I/O Channel	Direction	Transceiver
67	23(+)	Input/Output	RS-422
68	23(-)		

### Board Characteristics

Board name	IO303
Clock rate	33 MHz
Manufacturer	Speedgoat
Bus type	PCI
Multiple block instance support	Yes
Multiple board support	Yes

# Speedgoat IO303 PCI Read

---

**Purpose** Speedgoat IO303 PCI Read block

**Description** The Speedgoat IO303 PCI Read block reads a 32-bit data value from a PCI port offset address. Output is a `uint32` data type.

**Library** xPC Target Library for Speedgoat

## Block Parameters

---

**Note** The Speedgoat FPGA I/O board block descriptions in this section are for informational purposes only. The HDL Coder HDL Workflow Advisor uses these blocks to generate an xPC Target interface subsystem. The subsystem mask controls all of the block parameters. Typically, you do not need to directly control these blocks.

---

### Device index

Enter a number between 0 and 7. The maximum number of allowed boards is 8.

### Port name

Enter the FPGA output port name.

### Port offset

Enter the PCI register offset address for the ports being read.

Valid offset addresses begin at `0x8000`. Each PCI address must be unique and 32-bit aligned, such as `hex2dec('8000')`, `hex2dec('8004')`, and so on.

### Port type

Enter the data type of the ports being read (`uint32` (fixed)).

### Port dimension

Enter the dimension of the ports being read (*number of addresses to read*). If **Port dimension** is greater than 1, you must use a strobe.

## **Strobe offset**

Enter the PCI register offset address for the signal used to latch data from the FPGA into the ports being read. If **Strobe offset** is 0, no strobe is used.

Valid offset addresses begin at 0x8000. Each PCI address must be unique and 32-bit aligned, such as `hex2dec('8000')`, `hex2dec('8004')`, and so on.

## **Sample time**

Enter a base sample time or a multiple of the base sample time (-1 means sample time is inherited).

# Speedgoat IO303 PCI Setup

---

**Purpose** Speedgoat IO303 PCI Setup block

**Description** The Speedgoat IO303 PCI Setup block performs the setup and configuration required to use the IO303 board.

**Library** xPC Target Library for Speedgoat

## Block Parameters

---

**Note** The Speedgoat FPGA I/O board block descriptions in this section are for informational purposes only. The HDL Coder HDL Workflow Advisor uses these blocks to generate an xPC Target interface subsystem. The subsystem mask controls all of the block parameters. Typically, you do not need to directly control these blocks.

---

### FPGA model/subsystem name

Enter a string used to identify the model and subsystem containing the Setup block.

### Run FPGA only when xPC Target runs

Select this check box to enable the FPGA to execute only when the target application is running.

Clear this check box to enable the FPGA to start immediately after the target application is loaded.

### Reset FPGA states on start

Select this check box to cause the FPGA to reset the initial state of all variables and registers to the default state on start. Takes effect only when the check box **Run FPGA only when xPC Target runs** is selected.

Clear this check box to allow the FPGA to retain the end state of the previous execution. To return to the initial state, you must download the model to the target again and restart the simulation.

The default setting is Selected.

## **Device index**

Enter a number between 0 and 7. The maximum number of allowed boards is 8.

## **PCI slot (-1:autosearch)**

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

-1

to automatically locate the board.

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

```
getxpcpci
```

# Speedgoat IO303 PCI Write

---

**Purpose** Speedgoat IO303 PCI Write block

**Description** The Speedgoat IO303 PCI Write block writes a 32-bit data value to a PCI port offset address. Input must be a uint32 data type.

**Library** xPC Target Library for Speedgoat

## Block Parameters

---

**Note** The Speedgoat FPGA I/O board block descriptions in this section are for informational purposes only. The HDL Coder HDL Workflow Advisor uses these blocks to generate an xPC Target interface subsystem. The subsystem mask controls all of the block parameters. Typically, you do not need to directly control these blocks.

---

### Device index

Enter a number between 0 and 7. The maximum number of allowed boards is 8.

### Port name

Enter the FPGA input port name.

### Port offset

Enter the PCI register offset address for the ports being written.

Valid offset addresses begin at 0x8000. Each PCI address must be unique and 32-bit aligned, such as `hex2dec('8000')`, `hex2dec('8004')`, and so on.

### Port type

Enter the data type of the ports being written (uint32 (fixed)).

### Port dimension

Enter the dimension of the ports being written (*number of addresses to write*). If **Port dimension** is greater than 1, you must use a strobe.

## **Strobe offset**

Enter the PCI register offset address for the signal used to latch data into the FPGA from the ports being written. If **Strobe offset** is 0, no strobe is used.

Valid offset addresses begin at 0x8000. Each PCI address must be unique and 32-bit aligned, such as `hex2dec('8000')`, `hex2dec('8004')`, and so on.

## **Sample time**

Enter a base sample time or a multiple of the base sample time (-1 means sample time is inherited).

# Speedgoat IO311

---

**Purpose** Support for the Speedgoat IO311 FPGA board

**Board** Speedgoat IO311

**General Description** The Speedgoat IO311 is a field programmable gate array (FPGA) board that provides 64 bidirectional TTL I/O lines. This board is based on a Xilinx Virtex-II chip with 24192 logic cells. Design tools supporting this chip include the Xilinx ISE Design Suite (<http://www.xilinx.com/tools/designtools.htm>), version 10.1 required.

The clock rate for this board is 33 MHz.

The xPC Target block library supports this board with these driver blocks:

- Speedgoat IO311 PCI Read
- Speedgoat IO311 PCI Setup
- Speedgoat IO311 PCI Write

---

**Note** The Speedgoat FPGA I/O board block descriptions in this section are for informational purposes only. The HDL Coder HDL Workflow Advisor uses these blocks to generate an xPC Target interface subsystem. The subsystem mask controls all of the block parameters. Typically, you do not need to directly control these blocks.

---

The Speedgoat IO311 68 Pin Connector Map on page 43-38 lists the I/O channel to connector pin assignments.

## Speedgoat IO311 68 Pin Connector Map

Connector Pin	I/O Channel	Direction	Transceiver
1	0	Input/Output	TTL
2	1		

## Speedgoat IO311 68 Pin Connector Map (Continued)

Connector Pin	I/O Channel	Direction	Transceiver
3	2	Input/Output	TTL
4	3		
5	4	Input/Output	TTL
6	5		
7	6	Input/Output	TTL
8	7		
9	8	Input/Output	TTL
10	9		
11	10	Input/Output	TTL
12	11		
13	12	Input/Output	TTL
14	13		
15	14	Input/Output	TTL
16	15		
17	Ground		
18	Ground		
19	16	Input/Output	TTL
20	17		
21	18	Input/Output	TTL
22	19		
23	20	Input/Output	TTL
24	21		

# Speedgoat IO311

**Speedgoat IO311 68 Pin Connector Map (Continued)**

Connector Pin	I/O Channel	Direction	Transceiver
25	22	Input/Output	TTL
26	23		
27	24	Input/Output	TTL
28	25		
29	26	Input/Output	TTL
30	27		
31	28	Input/Output	TTL
32	29		
33	30	Input/Output	TTL
34	31		
35	Ground		
36	Ground		
37	32	Input/Output	TTL
38	33		
39	34	Input/Output	TTL
40	35		
41	36	Input/Output	TTL
42	37		
43	38	Input/Output	TTL
44	39		
45	40	Input/Output	TTL
46	41		

## Speedgoat IO311 68 Pin Connector Map (Continued)

Connector Pin	I/O Channel	Direction	Transceiver
47	42	Input/Output	TTL
48	43		
49	44	Input/Output	TTL
50	45		
51	46	Input/Output	TTL
52	47		
53	48	Input/Output	TTL
54	49		
55	50	Input/Output	TTL
56	51		
57	52	Input/Output	TTL
58	53		
59	54	Input/Output	TTL
60	55		
61	56	Input/Output	TTL
62	57		
63	58	Input/Output	TTL
64	59		
65	60	Input/Output	TTL
66	61		
67	62	Input/Output	TTL
68	63		

# Speedgoat IO311

---

## **Board Characteristics**

Board name	IO311
Clock rate	33 MHz
Manufacturer	Speedgoat
Bus type	PCI
Multiple block instance support	Yes
Multiple board support	Yes

**Purpose** Speedgoat IO311 PCI Read block

**Description** The Speedgoat IO311 PCI Read block reads a 32-bit data value from a PCI port offset address. Output is a `uint32` data type.

**Library** xPC Target Library for Speedgoat

## Block Parameters

---

**Note** The Speedgoat FPGA I/O board block descriptions in this section are for informational purposes only. The HDL Coder HDL Workflow Advisor uses these blocks to generate an xPC Target interface subsystem. The subsystem mask controls all of the block parameters. Typically, you do not need to directly control these blocks.

---

### Device index

Enter a number between 0 and 7. The maximum number of allowed boards is 8.

### Port name

Enter the FPGA output port name.

### Port offset

Enter the PCI register offset address for the ports being read.

Valid offset addresses begin at 0x8000. Each PCI address must be unique and 32-bit aligned, such as `hex2dec('8000')`, `hex2dec('8004')`, and so on.

### Port type

Enter the data type of the ports being read (`uint32` (fixed)).

### Port dimension

Enter the dimension of the ports being read (*number of addresses to read*). If **Port dimension** is greater than 1, you must use a strobe.

# Speedgoat IO311 PCI Read

---

## **Strobe offset**

Enter the PCI register offset address for the signal used to latch data from the FPGA into the ports being read. If **Strobe offset** is 0, no strobe is used.

Valid offset addresses begin at 0x8000. Each PCI address must be unique and 32-bit aligned, such as `hex2dec('8000')`, `hex2dec('8004')`, and so on.

## **Sample time**

Enter a base sample time or a multiple of the base sample time (-1 means sample time is inherited).

**Purpose** Speedgoat IO311 PCI Setup block

**Description** The Speedgoat IO311 PCI Setup block performs the setup and configuration required to use the IO311 board.

**Library** xPC Target Library for Speedgoat

## Block Parameters

---

**Note** The Speedgoat FPGA I/O board block descriptions in this section are for informational purposes only. The HDL Coder HDL Workflow Advisor uses these blocks to generate an xPC Target interface subsystem. The subsystem mask controls all of the block parameters. Typically, you do not need to directly control these blocks.

---

### FPGA model/subsystem name

Enter a string used to identify the model and subsystem containing the Setup block.

### Run FPGA only when xPC Target runs

Select this check box to enable the FPGA to execute only when the target application is running.

Clear this check box to enable the FPGA to start immediately after the target application is loaded.

### Reset FPGA states on start

Select this check box to cause the FPGA to reset the initial state of all variables and registers to the default state on start. Takes effect only when the check box **Run FPGA only when xPC Target runs** is selected.

Clear this check box to allow the FPGA to retain the end state of the previous execution. To return to the initial state, you must download the model to the target again and restart the simulation.

The default setting is Selected.

# Speedgoat IO311 PCI Setup

---

## **Device index**

Enter a number between 0 and 7. The maximum number of allowed boards is 8.

## **PCI slot (-1:autosearch)**

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

-1

to automatically locate the board.

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

```
getxpcpci
```

<b>Purpose</b>	Speedgoat IO311 PCI Write block
<b>Description</b>	The Speedgoat IO311 PCI Write block writes a 32-bit data value to a PCI port offset address. Input must be a uint32 data type.
<b>Library</b>	xPC Target Library for Speedgoat

## Block Parameters

---

**Note** The Speedgoat FPGA I/O board block descriptions in this section are for informational purposes only. The HDL Coder HDL Workflow Advisor uses these blocks to generate an xPC Target interface subsystem. The subsystem mask controls all of the block parameters. Typically, you do not need to directly control these blocks.

---

### Device index

Enter a number between 0 and 7. The maximum number of allowed boards is 8.

### Port name

Enter the FPGA input port name.

### Port offset

Enter the PCI register offset address for the ports being written.

Valid offset addresses begin at 0x8000. Each PCI address must be unique and 32-bit aligned, such as `hex2dec('8000')`, `hex2dec('8004')`, and so on.

### Port type

Enter the data type of the ports being written (uint32 (fixed)).

### Port dimension

Enter the dimension of the ports being written (*number of addresses to write*). If **Port dimension** is greater than 1, you must use a strobe.

# Speedgoat IO311 PCI Write

---

## **Strobe offset**

Enter the PCI register offset address for the signal used to latch data into the FPGA from the ports being written. If **Strobe offset** is 0, no strobe is used.

Valid offset addresses begin at 0x8000. Each PCI address must be unique and 32-bit aligned, such as `hex2dec('8000')`, `hex2dec('8004')`, and so on.

## **Sample time**

Enter a base sample time or a multiple of the base sample time (-1 means sample time is inherited).

**Purpose** Support for the Speedgoat IO312 FPGA board

**Board** Speedgoat IO312

**General Description** The Speedgoat IO312 is a field programmable gate array (FPGA) board that provides 32 bidirectional RS-422 I/O lines. This board is based on a Xilinx Virtex-II chip with 24192 logic cells. Design tools supporting this chip include the Xilinx ISE Design Suite (<http://www.xilinx.com/tools/designtools.htm>), version 10.1 required.

The clock rate for this board is 33 MHz.

The xPC Target block library supports this board with these driver blocks:

- Speedgoat IO312 PCI Read
- Speedgoat IO312 PCI Setup
- Speedgoat IO312 PCI Write

---

**Note** The Speedgoat FPGA I/O board block descriptions in this section are for informational purposes only. The HDL Coder HDL Workflow Advisor uses these blocks to generate an xPC Target interface subsystem. The subsystem mask controls all of the block parameters. Typically, you do not need to directly control these blocks.

---

The Speedgoat IO312 68 Pin Connector Map on page 43-49 lists the I/O channel to connector pin assignments.

### Speedgoat IO312 68 Pin Connector Map

Connector Pin	I/O Channel	Direction	Transceiver
1	0(+)	Input/Output	RS-422
2	0(-)		

# Speedgoat IO312

**Speedgoat IO312 68 Pin Connector Map (Continued)**

Connector Pin	I/O Channel	Direction	Transceiver
3	1(+)	Input/Output	RS-422
4	1(-)		
5	2(+)	Input/Output	RS-422
6	2(-)		
7	3(+)	Input/Output	RS-422
8	3(-)		
9	4(+)	Input/Output	RS-422
10	4(-)		
11	5(+)	Input/Output	RS-422
12	5(-)		
13	6(+)	Input/Output	RS-422
14	6(-)		
15	7(+)	Input/Output	RS-422
16	7(-)		
17	Ground		
18	Ground		
19	8(+)	Input/Output	RS-422
20	8(-)		
21	9(+)	Input/Output	RS-422
22	9(-)		
23	10(+)	Input/Output	RS-422
24	10(-)		

## Speedgoat IO312 68 Pin Connector Map (Continued)

Connector Pin	I/O Channel	Direction	Transceiver
25	11(+)	Input/Output	RS-422
26	11(-)		
27	12(+)	Input/Output	RS-422
28	12(-)		
29	13(+)	Input/Output	RS-422
30	13(-)		
31	14(+)	Input/Output	RS-422
32	14(-)		
33	15(+)	Input/Output	RS-422
34	15(-)		
35	Ground		
36	Ground		
37	16(+)	Input/Output	RS-422
38	16(-)		
39	17(+)	Input/Output	RS-422
40	17(-)		
41	18(+)	Input/Output	RS-422
42	18(-)		
43	19(+)	Input/Output	RS-422
44	19(-)		
45	20(+)	Input/Output	RS-422
46	20(-)		

# Speedgoat IO312

**Speedgoat IO312 68 Pin Connector Map (Continued)**

Connector Pin	I/O Channel	Direction	Transceiver
47	21(+)	Input/Output	RS-422
48	21(-)		
49	22(+)	Input/Output	RS-422
50	22(-)		
51	23(+)	Input/Output	RS-422
52	23(-)		
53	24(+)	Input/Output	RS-422
54	24(-)		
55	25(+)	Input/Output	RS-422
56	25(-)		
57	26(+)	Input/Output	RS-422
58	26(-)		
59	27(+)	Input/Output	RS-422
60	27(-)		
61	28(+)	Input/Output	RS-422
62	28(-)		
63	29(+)	Input/Output	RS-422
64	29(-)		
65	30(+)	Input/Output	RS-422
66	30(-)		
67	31(+)	Input/Output	RS-422
68	31(-)		

## **Board Characteristics**

Board name	IO312
Clock rate	33 MHz
Manufacturer	Speedgoat
Bus type	PCI
Multiple block instance support	Yes
Multiple board support	Yes

# Speedgoat IO312 PCI Read

---

**Purpose** Speedgoat IO312 PCI Read block

**Description** The Speedgoat IO312 PCI Read block reads a 32-bit data value from a PCI port offset address. Output is a `uint32` data type.

**Library** xPC Target Library for Speedgoat

## Block Parameters

---

**Note** The Speedgoat FPGA I/O board block descriptions in this section are for informational purposes only. The HDL Coder HDL Workflow Advisor uses these blocks to generate an xPC Target interface subsystem. The subsystem mask controls all of the block parameters. Typically, you do not need to directly control these blocks.

---

### Device index

Enter a number between 0 and 7. The maximum number of allowed boards is 8.

### Port name

Enter the FPGA output port name.

### Port offset

Enter the PCI register offset address for the ports being read.

Valid offset addresses begin at `0x8000`. Each PCI address must be unique and 32-bit aligned, such as `hex2dec('8000')`, `hex2dec('8004')`, and so on.

### Port type

Enter the data type of the ports being read (`uint32` (fixed)).

### Port dimension

Enter the dimension of the ports being read (*number of addresses to read*). If **Port dimension** is greater than 1, you must use a strobe.

## **Strobe offset**

Enter the PCI register offset address for the signal used to latch data from the FPGA into the ports being read. If **Strobe offset** is 0, no strobe is used.

Valid offset addresses begin at 0x8000. Each PCI address must be unique and 32-bit aligned, such as `hex2dec('8000')`, `hex2dec('8004')`, and so on.

## **Sample time**

Enter a base sample time or a multiple of the base sample time (-1 means sample time is inherited).

# Speedgoat IO312 PCI Setup

---

**Purpose** Speedgoat IO312 PCI Setup block

**Description** The Speedgoat IO312 PCI Setup block performs the setup and configuration required to use the IO312 board.

**Library** xPC Target Library for Speedgoat

## Block Parameters

---

**Note** The Speedgoat FPGA I/O board block descriptions in this section are for informational purposes only. The HDL Coder HDL Workflow Advisor uses these blocks to generate an xPC Target interface subsystem. The subsystem mask controls all of the block parameters. Typically, you do not need to directly control these blocks.

---

### FPGA model/subsystem name

Enter a string used to identify the model and subsystem containing the Setup block.

### Run FPGA only when xPC Target runs

Select this check box to enable the FPGA to execute only when the target application is running.

Clear this check box to enable the FPGA to start immediately after the target application is loaded.

### Reset FPGA states on start

Select this check box to cause the FPGA to reset the initial state of all variables and registers to the default state on start. Takes effect only when the check box **Run FPGA only when xPC Target runs** is selected.

Clear this check box to allow the FPGA to retain the end state of the previous execution. To return to the initial state, you must download the model to the target again and restart the simulation.

The default setting is Selected.

## **Device index**

Enter a number between 0 and 7. The maximum number of allowed boards is 8.

## **PCI slot (-1:autosearch)**

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

-1

to automatically locate the board.

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

```
getxpcpci
```

# Speedgoat IO312 PCI Write

---

**Purpose** Speedgoat IO312 PCI Write block

**Description** The Speedgoat IO312 PCI Write block writes a 32-bit data value to a PCI port offset address. Input must be a uint32 data type.

**Library** xPC Target Library for Speedgoat

## Block Parameters

---

**Note** The Speedgoat FPGA I/O board block descriptions in this section are for informational purposes only. The HDL Coder HDL Workflow Advisor uses these blocks to generate an xPC Target interface subsystem. The subsystem mask controls all of the block parameters. Typically, you do not need to directly control these blocks.

---

### Device index

Enter a number between 0 and 7. The maximum number of allowed boards is 8.

### Port name

Enter the FPGA input port name.

### Port offset

Enter the PCI register offset address for the ports being written.

Valid offset addresses begin at 0x8000. Each PCI address must be unique and 32-bit aligned, such as `hex2dec('8000')`, `hex2dec('8004')`, and so on.

### Port type

Enter the data type of the ports being written (uint32 (fixed)).

### Port dimension

Enter the dimension of the ports being written (*number of addresses to write*). If **Port dimension** is greater than 1, you must use a strobe.

## **Strobe offset**

Enter the PCI register offset address for the signal used to latch data into the FPGA from the ports being written. If **Strobe offset** is 0, no strobe is used.

Valid offset addresses begin at 0x8000. Each PCI address must be unique and 32-bit aligned, such as `hex2dec('8000')`, `hex2dec('8004')`, and so on.

## **Sample time**

Enter a base sample time or a multiple of the base sample time (-1 means sample time is inherited).

# Speedgoat IO313

---

**Purpose** Support for the Speedgoat IO313 FPGA board

**Board** Speedgoat IO313

## General Description

The Speedgoat IO313 is a field programmable gate array (FPGA) board that provides 16 bidirectional TTL and 24 bidirectional RS-422 I/O lines. This board is based on a Xilinx Virtex-II chip with 24192 logic cells. Design tools supporting this chip include the Xilinx ISE Design Suite (<http://www.xilinx.com/tools/designtools.htm>), version 10.1 required.

The clock rate for this board is 33 MHz.

The xPC Target block library supports this board with these driver blocks:

- Speedgoat IO313 PCI Read
- Speedgoat IO313 PCI Setup
- Speedgoat IO313 PCI Write

---

**Note** The Speedgoat FPGA I/O board block descriptions in this section are for informational purposes only. The HDL Coder HDL Workflow Advisor uses these blocks to generate an xPC Target interface subsystem. The subsystem mask controls all of the block parameters. Typically, you do not need to directly control these blocks.

---

The Speedgoat IO313 68 Pin Connector Map on page 43-60 lists the I/O channel to connector pin assignments.

### Speedgoat IO313 68 Pin Connector Map

Connector Pin	I/O Channel	Direction	Transceiver
1	0	Input/Output	TTL
2	1		

## Speedgoat IO313 68 Pin Connector Map (Continued)

Connector Pin	I/O Channel	Direction	Transceiver
3	2	Input/Output	TTL
4	3		
5	4	Input/Output	TTL
6	5		
7	6	Input/Output	TTL
8	7		
9	8	Input/Output	TTL
10	9		
11	10	Input/Output	TTL
12	11		
13	12	Input/Output	TTL
14	13		
15	14	Input/Output	TTL
16	15		
17	Ground		
18	Ground		
19	0(+)	Input/Output	RS-422
20	0(-)		
21	1(+)	Input/Output	RS-422
22	1(-)		
23	2(+)	Input/Output	RS-422
24	2(-)		

# Speedgoat IO313

**Speedgoat IO313 68 Pin Connector Map (Continued)**

Connector Pin	I/O Channel	Direction	Transceiver
25	3(+)	Input/Output	RS-422
26	3(-)		
27	4(+)	Input/Output	RS-422
28	4(-)		
29	5(+)	Input/Output	RS-422
30	5(-)		
31	6(+)	Input/Output	RS-422
32	6(-)		
33	7(+)	Input/Output	RS-422
34	7(-)		
35	Ground		
36	Ground		
37	8(+)	Input/Output	RS-422
38	8(-)		
39	9(+)	Input/Output	RS-422
40	9(-)		
41	10(+)	Input/Output	RS-422
42	10(-)		
43	11(+)	Input/Output	RS-422
44	11(-)		
45	12(+)	Input/Output	RS-422
46	12(-)		

## Speedgoat IO313 68 Pin Connector Map (Continued)

Connector Pin	I/O Channel	Direction	Transceiver
47	13(+)	Input/Output	RS-422
48	13(-)		
49	14(+)	Input/Output	RS-422
50	14(-)		
51	15(+)	Input/Output	RS-422
52	15(-)		
53	16(+)	Input/Output	RS-422
54	16(-)		
55	17(+)	Input/Output	RS-422
56	17(-)		
57	18(+)	Input/Output	RS-422
58	18(-)		
59	19(+)	Input/Output	RS-422
60	19(-)		
61	20(+)	Input/Output	RS-422
62	20(-)		
63	21(+)	Input/Output	RS-422
64	21(-)		
65	22(+)	Input/Output	RS-422
66	22(-)		
67	23(+)	Input/Output	RS-422
68	23(-)		

# Speedgoat IO313

---

## **Board Characteristics**

Board name	IO313
Clock rate	33 MHz
Manufacturer	Speedgoat
Bus type	PCI
Multiple block instance support	Yes
Multiple board support	Yes

**Purpose** Speedgoat IO313 PCI Read block

**Description** The Speedgoat IO313 PCI Read block reads a 32-bit data value from a PCI port offset address. Output is a `uint32` data type.

**Library** xPC Target Library for Speedgoat

## Block Parameters

---

**Note** The Speedgoat FPGA I/O board block descriptions in this section are for informational purposes only. The HDL Coder HDL Workflow Advisor uses these blocks to generate an xPC Target interface subsystem. The subsystem mask controls all of the block parameters. Typically, you do not need to directly control these blocks.

---

### Device index

Enter a number between 0 and 7. The maximum number of allowed boards is 8.

### Port name

Enter the FPGA output port name.

### Port offset

Enter the PCI register offset address for the ports being read.

Valid offset addresses begin at 0x8000. Each PCI address must be unique and 32-bit aligned, such as `hex2dec('8000')`, `hex2dec('8004')`, and so on.

### Port type

Enter the data type of the ports being read (`uint32` (fixed)).

### Port dimension

Enter the dimension of the ports being read (*number of addresses to read*). If **Port dimension** is greater than 1, you must use a strobe.

# Speedgoat IO313 PCI Read

---

## **Strobe offset**

Enter the PCI register offset address for the signal used to latch data from the FPGA into the ports being read. If **Strobe offset** is 0, no strobe is used.

Valid offset addresses begin at 0x8000. Each PCI address must be unique and 32-bit aligned, such as `hex2dec('8000')`, `hex2dec('8004')`, and so on.

## **Sample time**

Enter a base sample time or a multiple of the base sample time (-1 means sample time is inherited).

**Purpose** Speedgoat IO313 PCI Setup block

**Description** The Speedgoat IO313 PCI Setup block performs the setup and configuration required to use the IO313 board.

**Library** xPC Target Library for Speedgoat

## Block Parameters

---

**Note** The Speedgoat FPGA I/O board block descriptions in this section are for informational purposes only. The HDL Coder HDL Workflow Advisor uses these blocks to generate an xPC Target interface subsystem. The subsystem mask controls all of the block parameters. Typically, you do not need to directly control these blocks.

---

### FPGA model/subsystem name

Enter a string used to identify the model and subsystem containing the Setup block.

### Run FPGA only when xPC Target runs

Select this check box to enable the FPGA to execute only when the target application is running.

Clear this check box to enable the FPGA to start immediately after the target application is loaded.

### Reset FPGA states on start

Select this check box to cause the FPGA to reset the initial state of all variables and registers to the default state on start. Takes effect only when the check box **Run FPGA only when xPC Target runs** is selected.

Clear this check box to allow the FPGA to retain the end state of the previous execution. To return to the initial state, you must download the model to the target again and restart the simulation.

The default setting is Selected.

# Speedgoat IO313 PCI Setup

---

## **Device index**

Enter a number between 0 and 7. The maximum number of allowed boards is 8.

## **PCI slot (-1:autosearch)**

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

-1

to automatically locate the board.

If two or more boards of this type are in the target computer, enter the bus number and the PCI slot number of 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** Speedgoat IO313 PCI Write block

**Description** The Speedgoat IO313 PCI Write block writes a 32-bit data value to a PCI port offset address. Input must be a uint32 data type.

**Library** xPC Target Library for Speedgoat

## Block Parameters

---

**Note** The Speedgoat FPGA I/O board block descriptions in this section are for informational purposes only. The HDL Coder HDL Workflow Advisor uses these blocks to generate an xPC Target interface subsystem. The subsystem mask controls all of the block parameters. Typically, you do not need to directly control these blocks.

---

### Device index

Enter a number between 0 and 7. The maximum number of allowed boards is 8.

### Port name

Enter the FPGA input port name.

### Port offset

Enter the PCI register offset address for the ports being written.

Valid offset addresses begin at 0x8000. Each PCI address must be unique and 32-bit aligned, such as `hex2dec('8000')`, `hex2dec('8004')`, and so on.

### Port type

Enter the data type of the ports being written (uint32 (fixed)).

### Port dimension

Enter the dimension of the ports being written (*number of addresses to write*). If **Port dimension** is greater than 1, you must use a strobe.

# Speedgoat IO313 PCI Write

---

## **Strobe offset**

Enter the PCI register offset address for the signal used to latch data into the FPGA from the ports being written. If **Strobe offset** is 0, no strobe is used.

Valid offset addresses begin at 0x8000. Each PCI address must be unique and 32-bit aligned, such as `hex2dec('8000')`, `hex2dec('8004')`, and so on.

## **Sample time**

Enter a base sample time or a multiple of the base sample time (-1 means sample time is inherited).

**Purpose** Support for the Speedgoat IO314 FPGA board

**Board** Speedgoat IO314

**General Description** The Speedgoat IO314 is a field programmable gate array (FPGA) board that provides 32 bidirectional LVDS I/O lines. This board is based on a Xilinx Virtex-II chip with 24192 logic cells. Design tools supporting this chip include the Xilinx ISE Design Suite (<http://www.xilinx.com/tools/designtools.htm>), version 10.1 required.

The clock rate for this board is 33MHz.

The xPC Target block library supports this board with these driver blocks:

- Speedgoat IO314 PCI Read
- Speedgoat IO314 PCI Setup
- Speedgoat IO314 PCI Write

---

**Note** The Speedgoat FPGA I/O board block descriptions in this section are for informational purposes only. The HDL Coder HDL Workflow Advisor uses these blocks to generate an xPC Target interface subsystem. The subsystem mask controls all of the block parameters. Typically, you do not need to directly control these blocks.

---

The Speedgoat IO314 provides up to 32 channels of 2.5 volt LVDS. See Speedgoat IO325 68 Pin Connector Map (LVDS) on page 43-86 for more information on channel/pin assignment.

# Speedgoat IO314

**Speedgoat IO314 68 Pin Connector Map (LVDS)**

Connector Pin	I/O Channel	Direction	Transceiver
1	0(+)	Input/Output	LVDS
35	0(-)		
2	1(+)	Input/Output	LVDS
36	1(-)		
3	2(+)	Input/Output	LVDS
37	2(-)		
4	3(+)	Input/Output	LVDS
38	3(-)		
5	4(+)	Input/Output	LVDS
39	4(-)		
6	5(+)	Input/Output	LVDS
40	5(-)		
7	6(+)	Input/Output	LVDS
41	6(-)		
8	7(+)	Input/Output	LVDS
42	7(-)		
9	8(+)	Input/Output	LVDS
43	8(-)		
10	9(+)	Input/Output	LVDS
44	9(-)		
11	10(+)	Input/Output	LVDS
45	10(-)		
12	Ground		

## Speedgoat IO314 68 Pin Connector Map (LVDS) (Continued)

Connector Pin	I/O Channel	Direction	Transceiver
46	Ground		
13	11(+)	Input/Output	LVDS
47	11(-)		
14	12(+)	Input/Output	LVDS
48	12(-)		
15	13(+)	Input/Output	LVDS
49	13(-)		
16	14(+)	Input/Output	LVDS
50	14(-)		
17	15(+)	Input/Output	LVDS
51	15(-)		
18	16(+)	Input/Output	LVDS
52	16(-)		
19	17(+)	Input/Output	LVDS
53	17(-)		
20	18(+)	Input/Output	LVDS
54	18(-)		
21	19(+)	Input/Output	LVDS
55	19(-)		
22	20(+)	Input/Output	LVDS
56	20(-)		
23	Ground		
57	Ground		

# Speedgoat IO314

**Speedgoat IO314 68 Pin Connector Map (LVDS) (Continued)**

Connector Pin	I/O Channel	Direction	Transceiver
24	21(+)	Input/Output	LVDS
58	21(-)		
25	22(+)	Input/Output	LVDS
59	22(-)		
26	23(+)	Input/Output	LVDS
60	23(-)		
27	24(+)	Input/Output	LVDS
61	24(-)		
28	25(+)	Input/Output	LVDS
62	25(-)		
29	26(+)	Input/Output	LVDS
63	26(-)		
30	27(+)	Input/Output	LVDS
64	27(-)		
31	28(+)	Input/Output	LVDS
65	28(-)		
32	29(+)	Input/Output	LVDS
66	29(-)		
33	30(+)	Input/Output	LVDS
67	30(-)		
34	31(+)	Input/Output	LVDS
68	31(-)		

## **Board Characteristics**

Board name	IO314
Clock rate	33MHz
Manufacturer	Speedgoat
Bus type	PCI
Multiple block instance support	Yes
Multiple board support	Yes

# Speedgoat IO314 PCI Read

---

**Purpose** Speedgoat IO314 PCI Read block

**Description** The Speedgoat IO314 PCI Read block reads a 32-bit data value from a PCI port offset address. Output is a `uint32` data type.

**Library** xPC Target Library for Speedgoat

## Block Parameters

---

**Note** The Speedgoat FPGA I/O board block descriptions in this section are for informational purposes only. The HDL Coder HDL Workflow Advisor uses these blocks to generate an xPC Target interface subsystem. The subsystem mask controls all of the block parameters. Typically, you do not need to directly control these blocks.

---

### Device index

Enter a number between 0 and 7. The maximum number of allowed boards is 8.

### Port name

Enter the FPGA output port name.

### Port offset

Enter the PCI register offset address for the ports being read.

Valid offset addresses begin at `0x8000`. Each PCI address must be unique and 32-bit aligned, such as `hex2dec('8000')`, `hex2dec('8004')`, and so on.

### Port type

Enter the data type of the ports being read (`uint32` (fixed)).

### Port dimension

Enter the dimension of the ports being read (*number of addresses to read*). If **Port dimension** is greater than 1, you must use a strobe.

## **Strobe offset**

Enter the PCI register offset address for the signal used to latch data from the FPGA into the ports being read. If **Strobe offset** is 0, no strobe is used.

Valid offset addresses begin at 0x8000. Each PCI address must be unique and 32-bit aligned, such as `hex2dec('8000')`, `hex2dec('8004')`, and so on.

## **Sample time**

Enter a base sample time or a multiple of the base sample time (-1 means sample time is inherited).

# Speedgoat IO314 PCI Setup

---

**Purpose** Speedgoat IO314 PCI Setup block

**Description** The Speedgoat IO314 PCI Setup block performs the setup and configuration required to use the IO325 board.

**Library** xPC Target Library for Speedgoat

## Block Parameters

---

**Note** The Speedgoat FPGA I/O board block descriptions in this section are for informational purposes only. The HDL Coder HDL Workflow Advisor uses these blocks to generate an xPC Target interface subsystem. The subsystem mask controls all of the block parameters. Typically, you do not need to directly control these blocks.

---

### FPGA model/subsystem name

Enter a string used to identify the model and subsystem containing the Setup block.

### Run FPGA only when xPC Target runs

Select this check box to enable the FPGA to execute only when the target application is running.

Clear this check box to enable the FPGA to start immediately after the target application is loaded.

### Reset FPGA states on start

Select this check box to cause the FPGA to reset the initial state of all variables and registers to the default state on start. Takes effect only when the check box **Run FPGA only when xPC Target runs** is selected.

Clear this check box to allow the FPGA to retain the end state of the previous execution. To return to the initial state, you must download the model to the target again and restart the simulation.

The default setting is Selected.

## **Device index**

Enter a number between 0 and 7. The maximum number of allowed boards is 8.

## **PCI slot (-1:autosearch)**

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

-1

to automatically locate the board.

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

```
getxpcpci
```

# Speedgoat IO314 PCI Write

---

**Purpose** Speedgoat IO314 PCI Write block

**Description** The Speedgoat IO314 PCI Write block writes a 32-bit value to a PCI port offset address. Input must be a `uint32` data type.

**Library** xPC Target Library for Speedgoat

## Block Parameters

---

**Note** The Speedgoat FPGA I/O board block descriptions in this section are for informational purposes only. The HDL Coder HDL Workflow Advisor uses these blocks to generate an xPC Target interface subsystem. The subsystem mask controls all of the block parameters. Typically, you do not need to directly control these blocks.

---

### Device index

Enter a number between 0 and 7. The maximum number of allowed boards is 8.

### Port name

Enter the FPGA input port name.

### Port offset

Enter the PCI register offset address for the ports being written.

Valid offset addresses begin at `0x8000`. Each PCI address must be unique and 32-bit aligned, such as `hex2dec('8000')`, `hex2dec('8004')`, and so on.

### Port type

Enter the data type of the ports being written (`uint32` (fixed)).

### Port dimension

Enter the dimension of the ports being written (*number of addresses to write*). If **Port dimension** is greater than 1, you must use a strobe.

## **Strobe offset**

Enter the PCI register offset address for the signal used to latch data into the FPGA from the ports being written. If **Strobe offset** is 0, no strobe is used.

Valid offset addresses begin at 0x8000. Each PCI address must be unique and 32-bit aligned, such as `hex2dec('8000')`, `hex2dec('8004')`, and so on.

## **Sample time**

Enter a base sample time or a multiple of the base sample time (-1 means sample time is inherited).

# Speedgoat IO325

---

**Purpose** Support for the Speedgoat IO325 FPGA board

**Board** Speedgoat IO325

**General Description** The Speedgoat IO325 is a field programmable gate array (FPGA) board that provides 64 bidirectional LVC MOS or 32 bidirectional LVDS (four are input only) I/O lines and two 16-bit 105 MHz analog input channels. This board is based on a Xilinx Virtex-4 chip with 41472 logic cells. Design tools supporting this chip include the Xilinx ISE Design Suite (<http://www.xilinx.com/tools/designtools.htm>), version 10.1 required.

The clock rate for this board is 66 MHz.

The xPC Target block library supports this board with these driver blocks:

- Speedgoat IO325 A/D Calibration
- Speedgoat IO325 PCI Read
- Speedgoat IO325 PCI Setup
- Speedgoat IO325 PCI Write

---

**Note** The Speedgoat FPGA I/O board block descriptions in this section are for informational purposes only. The HDL Coder HDL Workflow Advisor uses these blocks to generate an xPC Target interface subsystem. The subsystem mask controls all of the block parameters. Typically, you do not need to directly control these blocks.

---

The Speedgoat IO325 provides up to 64 channels of 2.5 volt LVC MOS and up to 32 channels of 2.5 volt LVDS. You can use both LVC MOS and LVDS channels in a single application; however, channel selection must uniquely map to connector pins. For example, you cannot concurrently use the same connector pins for both LVC MOS and LVDS. See Speedgoat IO325 68 Pin Connector Map (LVC MOS) on page 43-83

and Speedgoat IO325 68 Pin Connector Map (LVDS) on page 43-86 for more information on channel/pin assignment.

The Speedgoat IO325 provides two analog input ports connected to a 16-bit A/D converter. See “Speedgoat IO325 A/D Input Ports” on page 43-90 for more information.

When defining...	Use Table
LVC MOS channels	Speedgoat IO325 68 Pin Connector Map (LVC MOS) on page 43-83
LVDS channels	Speedgoat IO325 68 Pin Connector Map (LVDS) on page 43-86

### Speedgoat IO325 68 Pin Connector Map (LVC MOS)

Connector Pin	I/O Channel	Direction	Transceiver
1	0	Input/Output	LVC MOS
35	1	Input/Output	LVC MOS
2	2	Input/Output	LVC MOS
36	3	Input/Output	LVC MOS
3	4	Input/Output	LVC MOS
37	5	Input/Output	LVC MOS
4	6	Input/Output	LVC MOS
38	7	Input/Output	LVC MOS
5	8	Input/Output	LVC MOS
39	9	Input/Output	LVC MOS
6	10	Input/Output	LVC MOS
40	11	Input/Output	LVC MOS
7	12	Input/Output	LVC MOS

# Speedgoat IO325

**Speedgoat IO325 68 Pin Connector Map (LVCMOS) (Continued)**

Connector Pin	I/O Channel	Direction	Transceiver
41	13	Input/Output	LVCMOS
8	14	Input/Output	LVCMOS
42	15	Input/Output	LVCMOS
9	16	Input/Output	LVCMOS
43	17	Input/Output	LVCMOS
10	18	Input/Output	LVCMOS
44	19	Input/Output	LVCMOS
11	20	Input/Output	LVCMOS
45	21	Input/Output	LVCMOS
12	22	Input/Output	LVCMOS
46	23	Input/Output	LVCMOS
13	24	Input/Output	LVCMOS
47	25	Input/Output	LVCMOS
14	26	Input/Output	LVCMOS
48	27	Input/Output	LVCMOS
15	28	Input/Output	LVCMOS
49	29	Input/Output	LVCMOS
16	30	Input/Output	LVCMOS
50	31	Input/Output	LVCMOS
17	32	Input/Output	LVCMOS
51	33	Input/Output	LVCMOS
18	34	Input/Output	LVCMOS
52	35	Input/Output	LVCMOS

## Speedgoat IO325 68 Pin Connector Map (LVCMOS) (Continued)

Connector Pin	I/O Channel	Direction	Transceiver
19	36	Input/Output	LVCMOS
53	37	Input/Output	LVCMOS
20	38	Input/Output	LVCMOS
54	39	Input/Output	LVCMOS
21	40	Input/Output	LVCMOS
55	41	Input/Output	LVCMOS
22	42	Input/Output	LVCMOS
56	43	Input/Output	LVCMOS
23	44	Input/Output	LVCMOS
57	45	Input/Output	LVCMOS
24	46	Input/Output	LVCMOS
58	47	Input/Output	LVCMOS
25	48	Input/Output	LVCMOS
59	49	Input/Output	LVCMOS
26	50	Input/Output	LVCMOS
60	51	Input/Output	LVCMOS
27	52	Input/Output	LVCMOS
61	53	Input/Output	LVCMOS
28	54	Input/Output	LVCMOS
62	55	Input/Output	LVCMOS
29	56	Input/Output	LVCMOS
63	57	Input/Output	LVCMOS
30	58	Input/Output	LVCMOS

# Speedgoat IO325

**Speedgoat IO325 68 Pin Connector Map (LVCMOS) (Continued)**

Connector Pin	I/O Channel	Direction	Transceiver
64	59	Input/Output	LVCMOS
31	60	Input/Output	LVCMOS
65	61	Input/Output	LVCMOS
32	62	Input/Output	LVCMOS
66	63	Input/Output	LVCMOS
33	Ground		
67	Ground		
34	Ground		
68	Ground		

**Speedgoat IO325 68 Pin Connector Map (LVDS)**

Connector Pin	I/O Channel	Direction	Transceiver
1	0(+)	Input	LVDS
35	0(-)		
2	1(+)	Input/Output	LVDS
36	1(-)		
3	2(+)	Input/Output	LVDS
37	2(-)		
4	3(+)	Input/Output	LVDS
38	3(-)		
5	4(+)	Input/Output	LVDS
39	4(-)		

## Speedgoat IO325 68 Pin Connector Map (LVDS) (Continued)

Connector Pin	I/O Channel	Direction	Transceiver
6	5(+)	Input	LVDS
40	5(-)		
7	6(+)	Input/Output	LVDS
41	6(-)		
8	7(+)	Input/Output	LVDS
42	7(-)		
9	8(+)	Input/Output	LVDS
43	8(-)		
10	9(+)	Input/Output	LVDS
44	9(-)		
11	10(+)	Input/Output	LVDS
45	10(-)		
12	11(+)	Input/Output	LVDS
46	11(-)		
13	12(+)	Input/Output	LVDS
47	12(-)		
14	13(+)	Input/Output	LVDS
48	13(-)		
15	14(+)	Input/Output	LVDS
49	14(-)		
16	15(+)	Input	LVDS
50	15(-)		

# Speedgoat IO325

**Speedgoat IO325 68 Pin Connector Map (LVDS) (Continued)**

Connector Pin	I/O Channel	Direction	Transceiver
17	16(+)	Input/Output	LVDS
51	16(-)		
18	17(+)	Input/Output	LVDS
52	17(-)		
19	18(+)	Input/Output	LVDS
53	18(-)		
20	19(+)	Input/Output	LVDS
54	19(-)		
21	20(+)	Input/Output	LVDS
55	20(-)		
22	21(+)	Input/Output	LVDS
56	21(-)		
23	22(+)	Input/Output	LVDS
57	22(-)		
24	23(+)	Input	LVDS
58	23(-)		
25	24(+)	Input/Output	LVDS
59	24(-)		
26	25(+)	Input/Output	LVDS
60	25(-)		
27	26(+)	Input/Output	LVDS
61	26(-)		

## Speedgoat IO325 68 Pin Connector Map (LVDS) (Continued)

Connector Pin	I/O Channel	Direction	Transceiver
28	27(+)	Input/Output	LVDS
62	27(-)		
29	28(+)	Input/Output	LVDS
63	28(-)		
30	29(+)	Input/Output	LVDS
64	29(-)		
31	30(+)	Input/Output	LVDS
65	30(-)		
32	31(+)	Input/Output	LVDS
66	31(-)		
33	Ground		
67	Ground		
34	Ground		
68	Ground		

### Board Characteristics

Board name	IO325
Clock rate	66 MHz
Manufacturer	Speedgoat
Bus type	PCI
Multiple block instance support	Yes
Multiple board support	Yes

# Speedgoat IO325

---

## Speedgoat IO325 A/D Input Ports

The Speedgoat IO325 A/D supports two analog input ports and transmits their measured voltage to the FPGA as signed 16-bit fixed-point integers, where +32767 represents +1.6 V, 0 represents 0 V, and -32767 represents -1.6 V. Voltages scale linearly.

When you assign an input port to the A/D in HDL Workflow Advisor, the software automatically adds calibration parameters to the interface block. For more on preparing a Speedgoat FPGA using HDL Workflow Advisor, see “xPC Target Interface Generation for Speedgoat Boards”.

### Calibration Procedure

Calibrate the IO325 A/D port assembly using the following procedure.

---

**Note** Calibration requires a precision voltage source capable of generating the required calibration voltages into a 50  $\Omega$  load to the desired accuracy and precision.

---

Apply the following procedure to Ch 1 and Ch 2:

**1**

Initialize parameter  $\text{Gain}_{\text{Ch}}$  to 1.0 in the subsystem interface block mask.

---

**Note** See Speedgoat IO325 A/D Calibration for information on the Gain and Offset data entry format.

---

**2**

Connect the voltage source to channel Ch.

**3**

Apply 0 V to Ch and read the measured value from the A/D several times, recording your measurements. (The measured value will be noisy, with a typical uncertainty of  $\pm 30$  units.)

**4**

Pick an average value.

**5**

Open the subsystem interface block mask and enter the negative of the average value into the **Offset** parameter for channel Ch, leaving  $\text{Gain}_{\text{Ch}}$  set to 1.0.

**6**

Rebuild your model and check the results.

**7**

If the average value is acceptably close to 0, continue with step 8. Otherwise, repeat steps 1–6 until you have refined  $\text{Offset}_{\text{Ch}}$  to the desired accuracy and precision.

**8**

Choose a calibration voltage and calculate its expected value depending upon the desired scaling. For the case where you want 1.6 V to give 32767 units and the calibration voltage is 1.0 V:

$$1.6 / 32767 = 1.0 / \textit{ExpectedValue}$$

$$\textit{ExpectedValue} = 1.0 / 1.6 * 32767$$

$$\textit{ExpectedValue} = 20479$$

**9**

Apply the calibration voltage (1 V) to the channel and read the measured value from the A/D. Calculate  $\text{Gain}_{\text{Ch}}$  using the following equation:

$$\textit{Gain} = \textit{ExpectedValue} / \textit{MeasuredValue}$$

# Speedgoat IO325

---

## 10

Open the subsystem interface block mask and enter the calculated  $Gain_{Ch}$  value into the block parameters for channel Ch.

## 11

Rebuild your model and check the results.

## 12

Repeat steps 1–11 for the other channel.

During execution, the A/D applies the calibration factors to each channel using the following equation:

$$CalibratedOutput_{Ch} = (MeasuredValue_{Ch} + Offset_{Ch}) * Gain_{Ch}$$

For instance, suppose the Ch1 calibrates as follows:

<b>Expected Value</b>	<b>Measured Value</b>	<b>Offset</b>	<b>Gain</b>
0 (~0 V)	425	-425	—
20479 (~1 V)	20000	—	1.024

During execution, if the measured value at Ch 1 is 10000, the calibrated value is:

$$CalibratedOutput_{Ch1} = (10000 + (-425)) * 1.024 = 9805$$

or approximately 0.48 V.

## Port Characteristics

Name	AXM-A30
Input configuration	Two differential channels using two Analog Devices™ AD9460 A/D converters

Manufacturer	Acromag
A/D resolution	16 bits
Input range	3.4 V peak-to-peak, centered at 0V, into 50 $\Omega$ load
Operating Temperature	0 – 70 °C
Signal-to-noise ratio	69 dB (25 °C) typical
Signal-to-noise and distortion	67dB (25 °C) typical
Clock rate	66 MHz (fixed)
Maximum throughput rate (1 and 2 channel)	9.5 nS (105 MHz)
A/D trigger	FPGA controlled
Connectors	Four SMA PCB jack receptacle connectors (non-isolated)

## Speedgoat IO325 A/D Input Port Connector Map

<b>SMA Connector</b>	<b>Connector Description</b>
1	Analog Input Ch1
2	External Clock (not used)
3	External Trigger (not used)
4	Analog Input Ch2

# Speedgoat IO325 A/D Calibration

---

**Purpose** Speedgoat IO325 AXM-A30 Calibration block factors

**Description** The Speedgoat IO325 A/D Calibration block is used to generate A/D gain and offset corrections for the IO325 A/D Input Port Assembly.

**Library** xPC Target Library for Speedgoat

## **Block Parameters**

---

**Note** The Speedgoat FPGA I/O board block descriptions in this section are for informational purposes only. The HDL Coder HDL Workflow Advisor uses these blocks to generate an xPC Target interface subsystem. The subsystem mask controls all of the block parameters. Typically, you do not need to directly control these blocks.

---

### **Device index**

Enter a number between 0 and 7. The maximum number of allowed boards is 8.

### **A/D Gain Corrections in the range [0:1.999]**

Enter pairs of the form [Ch1Gain, Ch2Gain], where Ch1Gain is the gain correction (double) for channel 1 and Ch2Gain is the gain correction (double) for channel 2.

### **A/D Offset Corrections**

Enter pairs of the form [Ch1offset, Ch2offset], where Ch1offset is the offset correction (int16) for channel 1 and Ch2offset is the offset correction (int16) for channel 2.

For more on how to calculate these factors, see “Speedgoat IO325 A/D Input Ports” on page 43-90.

**Purpose** Speedgoat IO325 PCI Read block

**Description** The Speedgoat IO325 PCI Read block reads a 32-bit data value from a PCI port offset address. Output is a `uint32` data type.

**Library** xPC Target Library for Speedgoat

## Block Parameters

---

**Note** The Speedgoat FPGA I/O board block descriptions in this section are for informational purposes only. The HDL Coder HDL Workflow Advisor uses these blocks to generate an xPC Target interface subsystem. The subsystem mask controls all of the block parameters. Typically, you do not need to directly control these blocks.

---

### Device index

Enter a number between 0 and 7. The maximum number of allowed boards is 8.

### Port name

Enter the FPGA output port name.

### Port offset

Enter the PCI register offset address for the ports being read.

Valid offset addresses begin at 0x8000. Each PCI address must be unique and 32-bit aligned, such as `hex2dec('8000')`, `hex2dec('8004')`, and so on.

### Port type

Enter the data type of the ports being read (`uint32` (fixed)).

### Port dimension

Enter the dimension of the ports being read (*number of addresses to read*). If **Port dimension** is greater than 1, you must use a strobe.

# Speedgoat IO325 PCI Read

---

## **Strobe offset**

Enter the PCI register offset address for the signal used to latch data from the FPGA into the ports being read. If **Strobe offset** is 0, no strobe is used.

Valid offset addresses begin at 0x8000. Each PCI address must be unique and 32-bit aligned, such as `hex2dec('8000')`, `hex2dec('8004')`, and so on.

## **Sample time**

Enter a base sample time or a multiple of the base sample time (-1 means sample time is inherited).

<b>Purpose</b>	Speedgoat IO325 PCI Setup block
<b>Description</b>	The Speedgoat IO325 PCI Setup block performs the setup and configuration required to use the IO325 board.
<b>Library</b>	xPC Target Library for Speedgoat

## Block Parameters

---

**Note** The Speedgoat FPGA I/O board block descriptions in this section are for informational purposes only. The HDL Coder HDL Workflow Advisor uses these blocks to generate an xPC Target interface subsystem. The subsystem mask controls all of the block parameters. Typically, you do not need to directly control these blocks.

---

### FPGA model/subsystem name

Enter a string used to identify the model and subsystem containing the Setup block.

### Run FPGA only when xPC Target runs

Select this check box to enable the FPGA to execute only when the target application is running.

Clear this check box to enable the FPGA to start immediately after the target application is loaded.

### Reset FPGA states on start

Select this check box to cause the FPGA to reset the initial state of all variables and registers to the default state on start. Takes effect only when the check box **Run FPGA only when xPC Target runs** is selected.

Clear this check box to allow the FPGA to retain the end state of the previous execution. To return to the initial state, you must download the model to the target again and restart the simulation.

The default setting is Selected.

# Speedgoat IO325 PCI Setup

---

## **Device index**

Enter a number between 0 and 7. The maximum number of allowed boards is 8.

## **PCI slot (-1:autosearch)**

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

-1

to automatically locate the board.

If two or more boards of this type are in the target computer, enter the bus number and the PCI slot number of 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** Speedgoat IO325 PCI Write block

**Description** The Speedgoat IO325 PCI Write block writes a 32-bit value to a PCI port offset address. Input must be a `uint32` data type.

**Library** xPC Target Library for Speedgoat

## Block Parameters

---

**Note** The Speedgoat FPGA I/O board block descriptions in this section are for informational purposes only. The HDL Coder HDL Workflow Advisor uses these blocks to generate an xPC Target interface subsystem. The subsystem mask controls all of the block parameters. Typically, you do not need to directly control these blocks.

---

### Device index

Enter a number between 0 and 7. The maximum number of allowed boards is 8.

### Port name

Enter the FPGA input port name.

### Port offset

Enter the PCI register offset address for the ports being written.

Valid offset addresses begin at 0x8000. Each PCI address must be unique and 32-bit aligned, such as `hex2dec('8000')`, `hex2dec('8004')`, and so on.

### Port type

Enter the data type of the ports being written (`uint32` (fixed)).

### Port dimension

Enter the dimension of the ports being written (*number of addresses to write*). If **Port dimension** is greater than 1, you must use a strobe.

# Speedgoat IO325 PCI Write

---

## **Strobe offset**

Enter the PCI register offset address for the signal used to latch data into the FPGA from the ports being written. If **Strobe offset** is 0, no strobe is used.

Valid offset addresses begin at 0x8000. Each PCI address must be unique and 32-bit aligned, such as `hex2dec('8000')`, `hex2dec('8004')`, and so on.

## **Sample time**

Enter a base sample time or a multiple of the base sample time (-1 means sample time is inherited).

# Texas Instruments

---

This topic describes the Texas Instruments™ I/O board supported by the xPC Target product (<http://www.ti.com>).

Texas Instruments DM642 EVM

The DM642 is an evaluation module that the xPC Target product works with in conjunction with Embedded Coder® and Simulink Coder, available from The MathWorks.

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

<b>Purpose</b>	Evaluation Module for the Texas Instruments DM642 I/O board
<b>Board</b>	Texas Instruments DM642 EVM
<b>General Description</b>	<p>You can use the xPC Target and Texas Instruments DM642 Evaluation Module (EVM) hardware with the xPC Target , Simulink Coder, and Embedded Coder software to develop a hybrid application. This enables you to build and download an xPC Target application to a target computer that contains a DM642 EVM board. This application can then communicate and exchange data with the DM642 EVM through a model that contains blocks that enable communication with the xPC Target software.</p> <p>The xPC Target block library supports this board with these driver blocks:</p> <ul style="list-style-type: none"><li>• TI DM642 Init</li><li>• TI DM642 RX</li><li>• TI DM642 TX</li></ul> <p>To access these blocks, type <code>xpctexasinstrumentslib</code> at the MATLAB Command Window. If an xPC Target model includes a TI DM642 RX or TI DM642 TX block, you must include a TI DM642 Init block.</p> <p>For an example of this data exchange, see the xPC Target example, <i>Exchanging Data with the TI DM642 EVM (TI dm642xPC)</i>, and its counterpart, <i>PCI Bus Data Exchange Data with xPC (c6000pcixpc)</i>.</p> <p>To use these examples, see the following procedure. It assumes that you are familiar with how to build and download Embedded Coder models. See the Embedded Coder documentation for details. It also assumes that you are familiar with how to build and download an xPC Target model to a target computer. Note that these examples have been configured to work together; you will not need to make additional changes to them.</p> <ol style="list-style-type: none"><li>1 Install the DM642 EVM in the target computer.</li></ol>

# Texas Instruments DM642 EVM

---

- 2** Install the Embedded Coder product on the host computer. You must install this product to use the xPC Target DM642 blocks.
- 3** From the MATLAB Command Window, run the example model, PCI Bus Data Exchange Data with xPC (c6000pciipc). Build and download this model to the DM642 EVM. For example, press **Ctrl+B** in model.
- 4** From the MATLAB Command Window, run the xPC Target example model, Exchanging Data with the TI DM642 EVM.
- 5** Build and download this model to the target computer in which you have installed the DM642 EVM. For example, press **Ctrl+B** in model.

**Purpose** TI DM642 initialization block

**Library** xPC Target Library for Texas Instruments

**Block Parameters** **Page**

Specify a value, in hex, for the page to write into the DSP page register. For example, to select the DM642 memory address 0x80000000, enter a value of 0x200. This is the default value.

Modify this value only if you want the DM642 to use a new value.

---

**Note** This page must match the one accessed by the Embedded Coder model.

---

### **PCI Slot (-1:autosearch)**

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

-1

to automatically locate the board.

If two or more boards of this type are in the target computer, enter the bus number and the PCI slot number of 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** TI DM642 receive block

**Library** xPC Target Library for Texas Instruments

**Block Parameters** **Region**  
Enter a unique number (starting with 1) to identify the memory region for this block.

---

**Note** Region numbers must be the same regions as those identified in the Embedded Coder DM642 model.

---

### Data type and size (cell array)

Enter a cell array of character strings and port widths, in the form of `{'datatype[port width]'}`, for each port. Valid data types are:

- 'double'
- 'single'
- 'int8'
- 'uint8'
- 'int16'
- 'uint16'
- 'int32'
- 'uint32'
- 'boolean'

For example, a value of `{'double[1]'}` specifies a port data type of double and a width of 1.

### New port

Select this check box to create an output port labeled new. This is a port of type boolean to indicate if the port data is new. This port

does not indicate that data has changed, only that the DM642 has written data since the xPC Target software last read data. You might want to use this check box if this block is reading data asynchronously.

**Block until data is available**

Select this check box to read data only when data is ready to be read. The block is in a busy wait until then. Be careful to avoid deadlock situations when selecting this block.

**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 computer, enter

-1

to automatically locate the board.

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

```
getxpcpci
```

<b>Purpose</b>	TI DM642 transmit block
<b>Library</b>	xPC Target Library for Texas Instruments
<b>Block Parameters</b>	<b>Region</b> Enter a unique number (starting with 1) to identify the memory region for this block.

---

**Note** Region numbers must match the regions specified in the DM642 model.

---

### Data type and size (cell array)

Enter a cell array of character strings and port widths, in the form of `{'datatype[port width]'}`, for each port. Valid data types are:

- 'double'
- 'single'
- 'int8'
- 'uint8'
- 'int16'
- 'uint16'
- 'int32'
- 'uint32'
- 'boolean'

For example, a value of `{'double[1]'}` specifies a port data type of double and a width of 1.

### Block until data has been read

Select this box to write data only after the device has read existing data. The block is in a busy wait state until the data has been

read. Be careful to avoid deadlock situations when selecting this block.

### **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 computer, enter

-1

to automatically locate the board.

If two or more boards of this type are in the target computer, enter the bus number and the PCI slot number of 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 topic describes the UEI groups of boards supported by the xPC Target product (<http://www.ueidaq.com>).

“Grouping the UEI Boards” on page 45-5	Explanation of the grouping of the UEI boards.
“Analog Input Frame Driver Blocks” on page 45-7	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 opto-isolated digital input and 64 opto-isolated 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 45-13	Description of block parameters for UEI driver blocks.

## Grouping the UEI Boards

### In this section...

“Introduction” on page 45-5

“Changing the Board Associated with the Block” on page 45-6

“Getting Information on a Specific Board” on page 45-6

### Introduction

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 the xPC Target product 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

In this section...
“Introduction” on page 45-7
“Notes on Master and Slave Boards” on page 45-7
“Interrupt Numbers” on page 45-8
“IRQ Source Block” on page 45-10
“Example Models” on page 45-11

### Introduction

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 the xPC Target software. The xPC Target model is executed when each frame completes on the board. To use the UEI frame driver blocks, note the IRQ values for each UEI board in the model and configure those boards with the model Configuration Parameters dialog box.

---

**Note** To use the UEI PowerDAQ board series with the xPC Target Analog Input frame driver blocks, you need to upgrade the UEI PowerDAQ boards. Contact your UEI representative and request the xPC Target upgrade for these boards.

---

### 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 **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 cables. This connects the acquisition clock and scan clock from the master board 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 45-9 — 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 45-9 — 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 allow the PCI BIOS to set up the plugged-in PCI cards, disable (set to No) the Plug-and-Play (PnP) operating system feature. The xPC Target kernel is not a PnP operating system; you must disable this feature or PCI devices will not work on the xPC Target software.

---

## 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 On the host computer, open the model that you want to use with the UEI board.
- 4 In the **Simulink > Model Configuration Parameters** dialog box, expand node **Code Generation** and select the **xPC Target options** node.
- 5 In the Real-time interrupt source parameter, select **Auto (PCI only)**.
- 6 From the **I/O board generating the interrupt** list, select the value **UEI-MFx**. This specifies that the UEI MFx board generates the interrupt.
- 7 For the **PCI slot (-1: autosearch) or ISA base address** parameter, enter the same PCI address as for the UEI MFx Frame block **PCI slot** parameter. Enter -1 (for autodetection) if this is the only UEI board in the target system.
- 8 Click **OK** and save the model.

## Guidelines for Multiple UEI Boards

If you have two or more UEI boards in a master/slave configuration, use 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** On the host computer, open the model that you want to use with the multiple UEI boards.
- 4** In the **Simulink > Model Configuration Parameters** dialog box, expand node **Code Generation** and select the **xPC Target options** node.
- 5** In the Real-time interrupt source parameter, select **Auto (PCI only)**.
- 6** From the **I/O board generating the interrupt** list, select the value **UEI-MFx**. This specifies that the UEI MFx board generates the interrupt.
- 7** From a MATLAB Command Window running on the host computer, type  

```
getxpcpci all
```
- 8** Note the slot numbers for the UEI boards.
- 9** Determine the slot number for the master board.
- 10** For the **PCI slot (-1: autosearch) or ISA base address** parameter, enter the bus and the slot number of the master board. Use the format [bus, slot].
- 11** 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 **Auto (PCI only)**.
- 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.
- 4** For the **PCI slot (-1: autosearch) or ISA base address** parameter, enter the same PCI address as for the UEI MFx Frame block **PCI slot (-1: autosearch)** 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. To operate these models requires you to set the required interrupt source in the simulation parameters dialog box.

Note that these examples use the UEI PD2-MF-16-333/16H driver block. Replace the block as required by your model.

- `xpcUEIframe` — 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` — A master/slave frame-based model.
- `xpcUEIasync` — 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` — 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` — 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**

# UEI PD2-MF 12-Bit Series

---

**Purpose** Support for the UEI PD2-MF 12-Bit Series I/O boards

**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. The xPC Target software does not support the counter/timers on these boards.

The xPC Target block library 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

<b>Board Characteristics</b>	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 computer, enter

-1

to automatically locate the board.

If two or more boards of this type are in the target computer, enter the bus number and the PCI slot number of 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

# UEI PD2-MF 12-Bit Series Frame Analog Input

---

corresponding channel has a longer settling time. To get a more 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 resulting from setting the **Mux settling time vector (slow bit)** might cause an acquisition overrun that the xPC Target software does not detect. For example, on a PD2-MF-xx-333/16H board, each acquisition takes 3 microseconds if the slow bit is 0. If the slow bit is 1, that same acquisition takes more than 10 microseconds. With the xPC Target software, 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.

## 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

$$\text{Frame time} = \text{Frame size} \times \text{Scan time}.$$

After you specify two of the parameters, specify -1 for the third parameter and the equation determines the third parameter.

---

**Note** Because the xPC Target software 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 **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 computer, 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 computer, enter the bus number and the PCI slot number of 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 computer, enter

-1

to automatically locate the board.

If two or more boards of this type are in the target computer, enter the bus number and the PCI slot number of 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 computer, enter

-1

to automatically locate the board.

If two or more boards of this type are in the target computer, enter the bus number and the PCI slot number of the board associated

# UEI PD2-MF 12-Bit Series Digital Input

---

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 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

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 computer, enter

-1

to automatically locate the board.

If two or more boards of this type are in the target computer, enter the bus number and the PCI slot number of 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

---

**Purpose** Support for the UEI PD2-MF 14-Bit Series I/O boards

**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. The xPC Target software does not support the counter/timers on these boards.

The xPC Target block library 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

<b>Board Characteristics</b>	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 computer, enter

-1

to automatically locate the board.

If two or more boards of this type are in the target computer, enter the bus number and the PCI slot number of 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

# UEI PD2-MF 14-Bit Series Frame Analog Input

---

corresponding channel has a longer settling time. To get a more 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 resulting from setting the **Mux settling time vector (slow bit)** value might cause an acquisition overrun that the xPC Target software does not detect. For example, on a PD2-MF-xx-333/16H board, each acquisition takes 3 microseconds if the slow bit is 0. If the slow bit is 1, that same acquisition takes more than 10 microseconds. With xPC Target software, 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.

## 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

$$\text{Frame time} = \text{Frame size} \times \text{Scan time}.$$

After you specify two of the parameters, specify -1 for the third parameter and the equation determines the third parameter.

---

**Note** Because the xPC Target software 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 **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 computer, 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 computer, enter the bus number and the PCI slot number of 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 computer, enter

-1

to automatically locate the board.

If two or more boards of this type are in the target computer, enter the bus number and the PCI slot number of 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 computer, enter

-1

to automatically locate the board.

If two or more boards of this type are in the target computer, enter the bus number and the PCI slot number of the board associated

## UEI PD2-MF 14-Bit Series Digital Input

---

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 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 computer, enter

-1

to automatically locate the board.

If two or more boards of this type are in the target computer, enter the bus number and the PCI slot number of 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

---

**Purpose** Support for the UEI PD2-MF 16-Bit Series I/O boards

**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. The xPC Target software does not support the counter/timers on these boards.

The xPC Target block library 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

<b>Board Characteristics</b>	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

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 computer, enter

-1

to automatically locate the board.

If two or more boards of this type are in the target computer, enter the bus number and the PCI slot number of 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

# UEI PD2-MF 16-Bit Series Frame Analog Input

---

corresponding channel has a longer settling time. To get a more 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 resulting from setting the **Mux settling time vector (slow bit)** value might cause an acquisition overrun that the xPC Target software does not detect. For example, on a PD2-MF-xx-333/16H board, each acquisition takes 3 microseconds if the slow bit is 0. If the slow bit is 1, that same acquisition takes more than 10 microseconds. With the xPC Target software, 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.

## 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

$$\text{Frame time} = \text{Frame size} \times \text{Scan time}.$$

After you specify two of the parameters, specify -1 for the third parameter and the equation determines the third parameter.

---

**Note** Because the xPC Target software 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 **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 computer, 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 computer, enter the bus number and the PCI slot number of 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 computer, enter

-1

to automatically locate the board.

If two or more boards of this type are in the target computer, enter the bus number and the PCI slot number of 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

<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 computer, enter

-1

to automatically locate the board.

If two or more boards of this type are in the target computer, enter the bus number and the PCI slot number of the board associated

## UEI PD2-MF 16-Bit Series Digital Input

---

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 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 computer, enter

-1

to automatically locate the board.

If two or more boards of this type are in the target computer, enter the bus number and the PCI slot number of 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

---

**Purpose** Support for the UEI PD2-MFS 12-Bit Series I/O boards

**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. The xPC Target software does not support the counter/timers on these boards.

The xPC Target block library 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

<b>Board Characteristics</b>	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 computer, enter

-1

to automatically locate the board.

If two or more boards of this type are in the target computer, enter the bus number and the PCI slot number of 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

# UEI PD2-MFS 12-Bit Series Frame Analog Input

---

corresponding channel has a longer settling time. To get a more 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 resulting from setting the **Mux settling time vector (slow bit)** value might cause an acquisition overrun that the xPC Target software does not detect. For example, on a PD2-MF-xx-333/16H board, each acquisition takes 3 microseconds if the slow bit is 0. If the slow bit is 1, that same acquisition takes more than 10 microseconds. With the xPC Target software, 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 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

$$\text{Frametime} = \text{Framesize} \times \text{Scantime}.$$

After you specify two of the parameters, specify -1 for the third parameter and the equation determines the third parameter.

---

**Note** Because the xPC Target software 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 **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 computer, 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 computer, enter the bus number and the PCI slot number of 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 computer, enter

-1

to automatically locate the board.

If two or more boards of this type are in the target computer, enter the bus number and the PCI slot number of 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 computer, enter

-1

to automatically locate the board.

If two or more boards of this type are in the target computer, enter the bus number and the PCI slot number of the board associated

## UEI PD2-MFS 12-Bit Series Digital Input

---

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 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 computer, enter

-1

to automatically locate the board.

If two or more boards of this type are in the target computer, enter the bus number and the PCI slot number of 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

Support for the EI PD2-MFS 14-Bit Series I/O boards

## 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. The xPC Target software does not support the counter/timers on these boards.

The xPC Target block library 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)

## UEI PD2-MFS 14-Bit Series

---

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 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 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 computer, enter

-1

to automatically locate the board.

If two or more boards of this type are in the target computer, enter the bus number and the PCI slot number of 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

# UEI PD2-MFS 14-Bit Series Frame Analog Input

---

corresponding channel has a longer settling time. To get a more 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 resulting from setting the **Mux settling time vector (slow bit)** value might cause an acquisition overrun that the xPC Target software does not detect. For example, on a PD2-MF-xx-333/16H board, each acquisition takes 3 microseconds if the slow bit is 0. If the slow bit is 1, that same acquisition takes more than 10 microseconds. With the xPC Target software, 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
- 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 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

$$\text{Frametime} = \text{Framesize} \times \text{Scantime}.$$

After you specify two of the parameters, specify -1 for the third parameter and the equation determines the third parameter.

---

**Note** Because the xPC Target software 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 **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 computer, enter

-1

to automatically locate the board.

# UEI PD2-MFS 14-Bit Series Frame Analog Input

---

If two or more boards of this type are in the target computer, enter the bus number and the PCI slot number of 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 computer, enter

-1

to automatically locate the board.

If two or more boards of this type are in the target computer, enter the bus number and the PCI slot number of 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

## 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 computer, enter

-1

to automatically locate the board.

If two or more boards of this type are in the target computer, enter the bus number and the PCI slot number of the board associated

# UEI PD2-MFS 14-Bit Series Digital Input

---

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 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 computer, enter

-1

to automatically locate the board.

If two or more boards of this type are in the target computer, enter the bus number and the PCI slot number of 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

Support for the UEI PD2-MFS 16-Bit Series I/O boards

## 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. The xPC Target software does not support the counter/timers on these boards.

The xPC Target block library 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

## UEI PD2-MFS 16-Bit Series

---

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 computer, enter

-1

to automatically locate the board.

If two or more boards of this type are in the target computer, enter the bus number and the PCI slot number of 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

# UEI PD2-MFS 16-Bit Series Frame Analog Input

---

corresponding channel has a longer settling time. To get a more 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 resulting from setting the **Mux settling time vector (slow bit)** value might cause an acquisition overrun that the xPC Target software does not detect. For example, on a PD2-MF-xx-333/16H board, each acquisition takes 3 microseconds if the slow bit is 0. If the slow bit is 1, that same acquisition takes more than 10 microseconds. With the xPC Target software, 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
- 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

$$\text{Frametime} = \text{Framesize} \times \text{Scantime}.$$

After you specify two of the parameters, specify -1 for the third parameter and the equation determines the third parameter.

---

**Note** Because the xPC Target software 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 **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 computer, 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 computer, enter the bus number and the PCI slot number of 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

## Scaling Input to Output

Hardware Output	Block Input Data Type	Scaling
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 computer, enter

-1

to automatically locate the board.

If two or more boards of this type are in the target computer, enter the bus number and the PCI slot number of 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 computer, enter

-1

to automatically locate the board.

If two or more boards of this type are in the target computer, enter the bus number and the PCI slot number of the board associated

# UEI PD2-MFS 16-Bit Series Digital Input

---

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 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 computer, enter

-1

to automatically locate the board.

If two or more boards of this type are in the target computer, enter the bus number and the PCI slot number of 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

Support for the UEI PDXI-MF 12-Bit Series I/O boards

## 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. The xPC Target software does not support the counter/timers on these boards.

The xPC Target block library 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)

## UEI PDXI-MF 12-Bit Series

---

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 computer, enter

-1

to automatically locate the board.

If two or more boards of this type are in the target computer, enter the bus number and the PCI slot number of 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

# UEI PDXI-MF 12-Bit Series Frame Analog Input

---

corresponding channel has a longer settling time. To get a more 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 resulting from setting the **Mux settling time vector (slow bit)** value might cause an acquisition overrun that the xPC Target software does not detect. For example, on a PD2-MF-xx-333/16H board, each acquisition takes 3 microseconds if the slow bit is 0. If the slow bit is 1, that same acquisition takes more than 10 microseconds. With the xPC Target software, 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.

# UEI PDXI-MF 12-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

$$\text{Frame time} = \text{Framesize} \times \text{Scan time}.$$

After you specify two of the parameters, specify -1 for the third parameter and the equation determines the third parameter.

---

**Note** Because the xPC Target software 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 **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 computer, 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 computer, enter the bus number and the PCI slot number of 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

## Scaling Input to Output

Hardware Output	Block Input Data Type	Scaling
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 computer, enter

-1

to automatically locate the board.

If two or more boards of this type are in the target computer, enter the bus number and the PCI slot number of 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 computer, enter

-1

to automatically locate the board.

If two or more boards of this type are in the target computer, enter the bus number and the PCI slot number of the board associated

# UEI PDXI-MF 12-Bit Series Digital Input

---

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 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 computer, enter

-1

to automatically locate the board.

If two or more boards of this type are in the target computer, enter the bus number and the PCI slot number of 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

Support for the UEI PDXI-MF 14-Bit Series I/O boards

## 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. the xPC Target software does not support the counter/timers on these boards.

The xPC Target block library 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

# UEI PDXI-MF 14-Bit Series

---

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 computer, enter

-1

to automatically locate the board.

If two or more boards of this type are in the target computer, enter the bus number and the PCI slot number of 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

# UEI PDXI-MF 14-Bit Series Frame Analog Input

---

corresponding channel has a longer settling time. To get a more 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 resulting from setting the **Mux settling time vector (slow bit)** value might cause an acquisition overrun that the xPC Target software does not detect. For example, on a PD2-MF-xx-333/16H board, each acquisition takes 3 microseconds if the slow bit is 0. If the slow bit is 1, that same acquisition takes more than 10 microseconds. With the xPC Target software, 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 the xPC Target software 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 **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 computer, 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 computer, enter the bus number and the PCI slot number of 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 computer, enter

-1

to automatically locate the board.

If two or more boards of this type are in the target computer, enter the bus number and the PCI slot number of 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 computer, enter

-1

to automatically locate the board.

If two or more boards of this type are in the target computer, enter the bus number and the PCI slot number of the board associated

# UEI PDXI-MF 14-Bit Series Digital Input

---

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 Output

**Purpose** PDXI-MF 14-Bit Series Digital Output block

**Library** xPC Target Library for UEI

## Scaling Input to Output

Hardware Output	Block Input Data Type	Scaling
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 computer, enter

-1

to automatically locate the board.

If two or more boards of this type are in the target computer, enter the bus number and the PCI slot number of 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

Support for the UEI PDXI-MF 16-Bit Series I/O boards

## 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. The xPC Target software does not support the counter/timers on these boards.

The xPC Target block library 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

## UEI PDXI-MF 16-Bit Series

---

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 computer, enter

-1

to automatically locate the board.

If two or more boards of this type are in the target computer, enter the bus number and the PCI slot number of 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

# UEI PDXI-MF 16-Bit Series Frame Analog Input

---

corresponding channel has a longer settling time. To get a more 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 resulting from setting the **Mux settling time vector (slow bit)** value might cause an acquisition overrun that the xPC Target software does not detect. For example, on a PD2-MF-xx-333/16H board, each acquisition takes 3 microseconds if the slow bit is 0. If the slow bit is 1, that same acquisition takes more than 10 microseconds. With the xPC Target software, 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.

# UEI PDXI-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

$$\text{Frametime} = \text{Framesize} \times \text{Scantime}.$$

After you specify two of the parameters, specify -1 for the third parameter and the equation determines the third parameter.

---

**Note** Because the xPC Target software 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 **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 computer, 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 computer, enter the bus number and the PCI slot number of 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 computer, enter

-1

to automatically locate the board.

If two or more boards of this type are in the target computer, enter the bus number and the PCI slot number of 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 computer, enter

```
-1
```

to automatically locate the board.

If two or more boards of this type are in the target computer, enter the bus number and the PCI slot number of the board associated

# UEI PDXI-MF 16-Bit Series Digital Input

---

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 Output

**Purpose** PDXI-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 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 computer, enter

-1

to automatically locate the board.

If two or more boards of this type are in the target computer, enter the bus number and the PCI slot number of 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

Support for the UEI PDXI-MFS 12-Bit Series I/O boards

## 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. The xPC Target software does not support the counter/timers on these boards.

The xPC Target block library 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 computer, enter

-1

to automatically locate the board.

If two or more boards of this type are in the target computer, enter the bus number and the PCI slot number of 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

# UEI PDXI-MFS 12-Bit Series Frame Analog Input

---

corresponding channel has a longer settling time. To get a more 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 resulting from setting the **Mux settling time vector (slow bit)** value might cause an acquisition overrun that the xPC Target software does not detect. For example, on a PD2-MF-xx-333/16H board, each acquisition takes 3 microseconds if the slow bit is 0. If the slow bit is 1, that same acquisition takes more than 10 microseconds. With the xPC Target software, 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

total number of samples = `FrameSize` x `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

$$\text{Frame time} = \text{Framesize} \times \text{Scantime}.$$

After you specify two of the parameters, specify -1 for the third parameter and the equation determines the third parameter.

---

**Note** Because the xPC Target software 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 **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 computer, 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 computer, enter the bus number and the PCI slot number of 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 computer, enter

-1

to automatically locate the board.

If two or more boards of this type are in the target computer, enter the bus number and the PCI slot number of 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 computer, enter

-1

to automatically locate the board.

If two or more boards of this type are in the target computer, enter the bus number and the PCI slot number of the board associated

# UEI PDXI-MFS 12-Bit Series Digital Input

---

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 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 computer, enter

-1

to automatically locate the board.

If two or more boards of this type are in the target computer, enter the bus number and the PCI slot number of 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

---

**Purpose** Support for the UEI PDXI-MFS 14-Bit Series I/O boards

**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. The xPC Target software does not support the counter/timers on these boards.

The xPC Target block library 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

# UEI PDXI-MFS 14-Bit Series

---

	PDXI-MFS-8-2M/14 PDXI-MFS-8-2M/14DG
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 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 computer, enter

-1

to automatically locate the board.

If two or more boards of this type are in the target computer, enter the bus number and the PCI slot number of 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

# UEI PDXI-MFS 14-Bit Series Frame Analog Input

---

corresponding channel has a longer settling time. To get a more 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 resulting from setting the **Mux settling time vector (slow bit)** value might cause an acquisition overrun that the xPC Target software does not detect. For example, on a PD2-MF-xx-333/16H board, each acquisition takes 3 microseconds if the slow bit is 0. If the slow bit is 1, that same acquisition takes more than 10 microseconds. With the xPC Target software, 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

$$\text{Frame time} = \text{Framesize} \times \text{Scantime}.$$

After you specify two of the parameters, specify -1 for the third parameter and the equation determines the third parameter.

---

**Note** Because the xPC Target software 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 **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 computer, 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 computer, enter the bus number and the PCI slot number of 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 computer, enter

-1

to automatically locate the board.

If two or more boards of this type are in the target computer, enter the bus number and the PCI slot number of 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 computer, enter

-1

to automatically locate the board.

If two or more boards of this type are in the target computer, enter the bus number and the PCI slot number of the board associated

# UEI PDXI-MFS 14-Bit Series Digital Input

---

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 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 computer, enter

-1

to automatically locate the board.

If two or more boards of this type are in the target computer, enter the bus number and the PCI slot number of 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

---

**Purpose** Support for the UEI PDXI-MFS 16-Bit Series I/O boards

**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. The xPC Target software does not support the counter/timers on these boards.

The xPC Target block library 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

<b>Board Characteristics</b>	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 computer, enter

-1

to automatically locate the board.

If two or more boards of this type are in the target computer, enter the bus number and the PCI slot number of 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

# UEI PDXI-MFS 16-Bit Series Frame Analog Input

---

corresponding channel has a longer settling time. To get a more 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 resulting from setting the **Mux settling time vector (slow bit)** value might cause an acquisition overrun that the xPC Target software does not detect. For example, on a PD2-MF-xx-333/16H board, each acquisition takes 3 microseconds if the slow bit is 0. If the slow bit is 1, that same acquisition takes more than 10 microseconds. With the xPC Target software, 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 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

$$\text{Frame time} = \text{Framesize} \times \text{Scantime}.$$

After you specify two of the parameters, specify -1 for the third parameter and the equation determines the third parameter.

---

**Note** Because the xPC Target software 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 **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 computer, 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 computer, enter the bus number and the PCI slot number of 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 computer, enter

-1

to automatically locate the board.

If two or more boards of this type are in the target computer, enter the bus number and the PCI slot number of 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 computer, enter

-1

to automatically locate the board.

If two or more boards of this type are in the target computer, enter the bus number and the PCI slot number of the board associated

# UEI PDXI-MFS 16-Bit Series Digital Input

---

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 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 computer, enter

-1

to automatically locate the board.

If two or more boards of this type are in the target computer, enter the bus number and the PCI slot number of 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

---

**Purpose** Support for the UEI PD2-AO Series I/O boards

**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.

The xPC Target block library 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

<b>Board Characteristics</b>	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

## Scaling Input to Output

Hardware Output	Block Input Data Type	Scaling
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 signal to the channels you do not want to use. For example,

# UEI PD2-AO Analog Output (D/A)

---

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.

If your board has 96 analog output channels, the board reads only the first value in the vector and resets all the channels according to that first value. For example, if the first value of the vector is 1, the board resets all 96 channels to the values specified in the **Initial value vector** parameter.

## **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 computer, enter

-1

to automatically locate the board.

If two or more boards of this type are in the target computer, enter the bus number and the PCI slot number of 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 computer, enter

-1

to automatically locate the board.

If two or more boards of this type are in the target computer, enter the bus number and the PCI slot number of the board associated

## UEI PD2-AO Digital Input

---

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 Output block

**Library** xPC Target Library for UEI

## Scaling Input to Output

Hardware Output	Block Input Data Type	Scaling
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 computer, enter

-1

to automatically locate the board.

If two or more boards of this type are in the target computer, enter the bus number and the PCI slot number of 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** Support for the UEI PD2-DIO-64 series I/O boards

**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. The xPC Target block library supports this series of boards with these driver blocks:

- UEI PD2-DIO-64 Digital Input
- UEI PD2-DIO-64 Digital Output

The xPC Target software 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 computer, enter

-1

to automatically locate the board.

If two or more boards of this type are in the target computer, enter the bus number and the PCI slot number of 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 computer, enter

-1

to automatically locate the board.

If two or more boards of this type are in the target computer, enter the bus number and the PCI slot number of 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

---

**Purpose** Support for the UEI PD2-DIO-128 Series I/O boards

**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.

The xPC Target block library supports this series of boards with these driver blocks:

- UEI PD2-DIO-128 Digital Input
- UEI PD2-DIO-128 Digital Output

The xPC Target software only supports the digital I/O channels for this board series. It does not support the additional functionality.

<b>Board Characteristics</b>	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 computer, 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 computer, enter the bus number and the PCI slot number of 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-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 computer, enter

-1

to automatically locate the board.

If two or more boards of this type are in the target computer, enter the bus number and the PCI slot number of 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** Support for the UEI PD2-DIO-128i Series I/O boards

**Board Series** UEI PD2-DIO-128i

**General Board Series** The PD2-DIO-128i series PCI boards have 64 opto-isolated digital input and 64 opto-isolated digital output ports in 16-bit ports.

The xPC Target block library supports this series of boards with these driver blocks:

- UEI PD2-DIO-128i Digital Input
- UEI PD2-DIO-128i Digital Output

<b>Board Characteristics</b>	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 computer, enter

-1

to automatically locate the board.

If two or more boards of this type are in the target computer, enter the bus number and the PCI slot number of 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 computer, enter

-1

to automatically locate the board.

If two or more boards of this type are in the target computer, enter the bus number and the PCI slot number of 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

---

**Purpose** Support for the UEI PDL-DIO-64 Series I/O boards

**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.

The xPC Target block library supports this series of boards with these driver blocks:

- UEI PDL-DIO-64 Digital Input
- UEI PDL-DIO-64 Digital Output

The xPC Target software only supports the digital I/O channels for this board series. It does not support the additional functionality.

<b>Board Characteristics</b>	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 computer, enter

-1

to automatically locate the board.

If two or more boards of this type are in the target computer, enter the bus number and the PCI slot number of 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

## Scaling Input to Output

Hardware Output	Block Input Data Type	Scaling
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 computer, enter

-1

to automatically locate the board.

If two or more boards of this type are in the target computer, enter the bus number and the PCI slot number of 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** Support for the UEI PDXI-DIO-64 Series I/O boards

**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. The xPC Target block library supports this series of boards with these driver blocks:

- UEI PDXI-DIO-64 Digital Input
- UEI PDXI-DIO-64 Digital Output

The xPC Target software only supports the digital I/O channels for this board series. It does not support the additional functionality.

<b>Board Characteristics</b>	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 computer, enter

-1

to automatically locate the board.

If two or more boards of this type are in the target computer, enter the bus number and the PCI slot number of 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 computer, enter

-1

to automatically locate the board.

If two or more boards of this type are in the target computer, enter the bus number and the PCI slot number of 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

---

**Purpose** Support for the UEI PDXI-AO Series I/O boards

**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. The xPC Target block library 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

<b>Board Characteristics</b>	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 computer, enter

-1

to automatically locate the board.

If two or more boards of this type are in the target computer, enter the bus number and the PCI slot number of 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 computer, enter

-1

to automatically locate the board.

If two or more boards of this type are in the target computer, enter the bus number and the PCI slot number of the board associated

## UEI PDXI-AO Digital Input

---

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 Output block

**Library** xPC Target Library for UEI

## Scaling Input to Output

Hardware Output	Block Input Data Type	Scaling
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 computer, enter

-1

to automatically locate the board.

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

```
getxpcpci
```

# Versallogic

---

This topic describes the Versallogic I/O board supported by the xPC Target product (<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 – Alphabetical List**

Versallogic VSBC-6

Versallogic VSBC-6 Analog Input (A/D)

Versallogic VSBC-6 Digital Input

Versallogic VSBC-6 Digital Output

Versallogic VSBC-6 Watch Dog

**Purpose** Support for Versallogic VSBC-6 single-board computer

**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.

The xPC Target block library 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

**Board Characteristics**

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.

**Purpose** VSBC-6 Digital Input block

**Library** xPC Target Library for Versallogic

<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 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. Each group of eight lines (1 to 8 or 9 to 16) can be only input or output. For example, if any bit in the range 1 to 8 is an input line, lines 1 to 8 must be input lines.

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. Each group of eight lines can only be input or output. Each group of eight lines (1 to 8 or 9 to 16) can be only input or output. For example, if any bit in the range 1 to 8 is an output line, lines 1 to 8 must be output lines.

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.

# Versalogic VSBC-6 Watch Dog

---

# Miscellaneous Blocks

---

This topic describes xPC Target miscellaneous blocks:

- “Asynchronous Event Support” on page 47-2
- “Blocks — Alphabetical List” on page 47-6

## Asynchronous Event Support

In this section...
“Adding an Asynchronous Event” on page 47-2
“Asynchronous Interrupt Examples” on page 47-4

### Adding an Asynchronous Event

The xPC Target software 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).

When developing a model in the Simulink interface 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 computer, you can add xPC Target asynchronous blocks to your Simulink model.

- 1 In the MATLAB Command Window, type

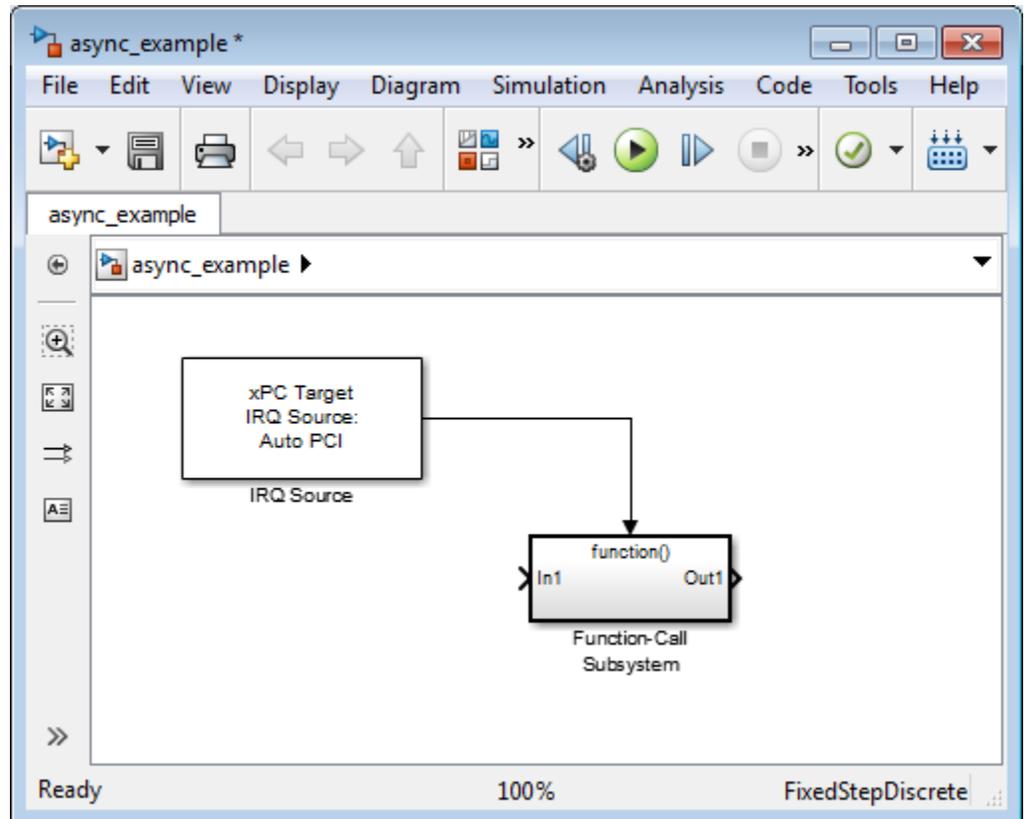
```
xpplib
```

The xPC Target Library opens.

- 2 Double-click the Asynchronous Event group block.

The Library: xpplib/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 Simulink Coder 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** drop-down list, choose **Auto (PCI only)** to enable the xPC Target software to automatically determine the IRQ that the BIOS assigned to the board and use it.

Alternatively, you can choose **3, 4, . . . , 15** for this number. To determine the available IRQ line numbers on the target computer, 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** drop-down list, select an interrupt board.
- 8 In the **PCI slot (-1: autosearch) or ISA base address** field, enter the PCI slot number or enter - 1 to let the xPC Target software 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 47-4.

If you are using a CAN field bus with interrupts, see “Asynchronous Interrupt Examples” on page 47-4.

## Asynchronous Interrupt Examples

The xPC Target software provides several example models. If you installed the MATLAB software 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 parallel port. 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 parallel port. Data exchange between an 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>Unlike the rate transition block, Async Buffer Write and Read blocks do not copy data from one buffer to another. Instead, the software disables interrupts and swaps buffer pointers. This method disables interrupts for a shorter time than the rate transition block 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 47-2

# 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 the Simulink and xPC Target software 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 specified interrupt occurs.
<b>Block Parameters</b>	<p><b>IRQ line number</b></p> <p>Select Auto (PCI only) to enable the xPC Target software to automatically determine the IRQ that the BIOS assigned to the board and use it.</p> <p>Alternatively, 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 computer 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>I/O board generating the interrupt</b></p> <p>For many I/O boards, you need to set up the board 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 board you intend to use from the drop-down list.</p> <p><b>PCI slot (-1: autosearch) or ISA base address</b></p> <ul style="list-style-type: none"><li>• If PCI:<ul style="list-style-type: none"><li>– If only one board of this type is in the target computer, enter -1</li></ul></li></ul> <p>to automatically locate the board.</p>

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

```
getxpcpci
```

- If ISA, enter the base address.

### **See Also**

“Asynchronous Event Support” on page 47-2

# 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 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 47-2

<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 computer hard disk and distributes that data in chunks every sample time. As the xPC Target kernel on the target computer 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 Byte Unpacking block. This 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 signal values by the Byte Unpacking block.</p> <p>See the following topics:</p> <ul style="list-style-type: none"><li>• “File Format” on page 47-11 — Describes the target computer source file format</li><li>• “Block Parameters” on page 47-13 — Describes the block parameters for the From File block</li></ul>
<b>File Format</b>	<p>Before you use a target computer file as the source for the From File block, you must format the data in the file. The file format is a concatenation of the different data elements for one time step, followed by the next time step, and so on.</p>

# From File

---

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 computer before transferring it to the target computer. 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 computer 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 to fill the buffer.

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

# From File

---

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.

## **Show IsValid port**

Select the **Show IsValid port** check box to make the port IsValid visible in the model. Port IsValid outputs 1 if the file read succeeds and 0 if it fails.

**Purpose** Data from target computer

**Library** xPC Target Library for Misc.

**Description** This block behaves like a source. Its output is usually connected to the input of a display device.

The From xPC Target block runs in nonreal time on the host computer. It is completely asynchronous to the model running in real time on the target computer. Depending on the sample time of the target computer and the actual speed at which the model containing the From xPC Target block executes (in nonreal time), it is possible for the From xPC Target block to query the target computer more than once in one sample time of the target computer. The target computer returns the same value in this case. Conversely, it is possible for the From xPC Target block to miss some sample time(s) between two successively returned values.

---

**Note** The use of From xPC Target blocks requires a connection between the host and target computer. Operations such as opening a model that contains these blocks or copying these blocks within or between models take significantly longer than normal with no connection between host and target computer.

---

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**.

# From xPC Target

---

## 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 the xPC Target software uses to build the target application.

### Signal name (block name)

The function `xpcsliface` automatically enters a name entry for this parameter. For multiple blocks, `xpcsliface` creates a From xPC Target block for each block. Using this method of specifying signals returns signal values one per time step.

You can also manually enter a cell array of signals for this parameter. Using this method of specifying signals returns the values of a vector of signals (up to 1000) as fast as it can acquire them. The signal values may not be at the same time step and the signal values are more likely to be spaced closely together.

### 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 Simulink Coder to build and download the target application to the default target computer. This assumes that you configured a default target computer through the xPC Target Explorer (see sections “Serial Communication Setup” and “Network Communication Setup” 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 computer.

## See Also

“Creating a Custom Graphical Interface”

**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, the xPC Target block library 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' }
```

- 4** Click **OK**.

The number of outputs from the block changes to reflect the length of the I/O-Port address cell array.

# I/O Port Read

---

## **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.

<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.
<b>Example</b>	<p>To access this special address space and communicate directly to a device, the xPC Target block library 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).</p> <ol style="list-style-type: none"><li>1 Double click the I/O Port Write block.  The Block Parameters: I/O Port Write dialog box opens.</li><li>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 <pre>{ '0x300', '0x304' }</pre></li><li>3 Click <b>OK</b>.  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 <code>uint32</code> writes a double word, an <code>uint16</code> input signal writes a word, and an <code>uint8</code> input signal writes a byte.</li></ol>
<b>Block Parameters</b>	<p><b>I/O-Port address (cell array)</b> 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.</p> <p><b>Sample time</b> Enter a base sample time or a multiple of the base sample time.</p>

# 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`). The output increments up at the CPU clock rate. To convert this rate to time in seconds, divide by clock frequency.

---

**Note** Use this block in models that are on Pentium machines only. Using this block in models that are not on 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

<b>Purpose</b>	Time stamp delta
<b>Library</b>	xPC Target Library for Misc.
<b>Description</b>	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.
<b>Block Parameters</b>	None

# Scope (xPC)

---

**Purpose** xPC Target Scope block

**Library** xPC Target Library for Misc.

**Description** xPC Target Scope blocks come in three types: Target, Host, and File. The block dialog box changes depending on the setting for parameter **Scope type**. By default, the block dialog box displays the parameters for Target scopes. The following sections describe the parameters for each scope type:

- “Target Scope Block Parameters” on page 47-22
- “Host Scope Block Parameters” on page 47-27
- “File Scope Block Parameters” on page 47-28

---

## Note

- If you want to monitor an output signal from a Constant block by connecting it to an xPC Target Scope block, you must add a test point for the Constant block output signal.
  - If your model has the output of a Mux block connected to the input of an xPC Target Scope block, the signal might not be observable. To observe the signal, add a unity gain block (a Gain block with a gain of 1) between the Mux block and the xPC Target Scope block.
- 

## Target Scope 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 one of the following:

- **Numerical** — Displays the data numerically.
- **Graphical redraw** — Displays a cycle of data continuously without any scrolling (refreshing the entire plot).
- **Graphical sliding** — Like graphical rolling mode. This mode displays the older data on the right side of the scope and the newer data on the left side of the scope. A vertical line separates the two types of data.
- **Graphical rolling** — Displays running data continuously scrolling from left to right across the scope (similar behavior to oscilloscopes).

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, the xPC Target software displays the signal using the default format of `%15.6f`. This format is a floating point one, 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]'
```

## Scope (xPC)

---

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:

Type	Description
<code>%e</code> or <code>%E</code>	Exponential format using <code>e</code> or <code>E</code>
<code>%f</code>	Floating point
<code>%g</code>	Signed value printed in <code>f</code> or <code>e</code> format depending on which is smaller
<code>%G</code>	Signed value printed in <code>f</code> or <code>E</code> 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 the first signal. The default numerical format (%15.6f) applies to the remaining signals. 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 default format applies to the remaining signals. If you have two entries and one signal, the xPC Target software 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 target scopes 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 target scopes 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.

## Scope (xPC)

---

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.

If you select a **Scope mode** of Numerical, the block updates the output every **Number of samples**.

### **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, 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”.

For additional information on triggers, see “Advanced Data Acquisition Topics”.

## Host Scope 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 target scope, the scope window opens automatically. With a host scope, 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.

# Scope (xPC)

---

## 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 `xpctarget.xpcsc.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.

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”.

## File Scope 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 file scope collects data up to **Number of samples**, then starts over again, overwriting the buffer. If the **AutoRestart** box is not selected, the file scope 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**

## Scope (xPC)

---

**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”.

### Filename

Enter a name for the file to contain the signal data. By default, the target computer writes the signal data to a file named `C:\data.dat`.

If you select the **Dynamic file name enabled** and **AutoRestart** check boxes, configure **Filename** to dynamically increment. Use a base file name, an underscore (`_`), and a `< >` specifier. Within the specifier, enter one to eight `%` symbols. Each symbol `%` represents a decimal location in the file name. The specifier can appear anywhere in the file name. For example, the following value for **Filename**, `C:\work\file_<%%>.dat` creates file names with the following pattern:

```
file_001.dat
file_002.dat
file_003.dat
```

The last file name of this series will be `file_999.dat`. If the function is still logging data when the last file name reaches its maximum size, the function starts from the beginning and overwrites the first file name in the series.

## 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.

## AutoRestart

Select this check box to enable the file scope 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 file scope collect data up to **Number of samples**, then stop.

Setting this check box enables the **Dynamic file name enabled** and **Max file size in bytes (multiple of WriteSize)** parameters.

## Dynamic file name enabled

Select this check box to enable the ability to dynamically create multiple log files for file scopes.

To enable this parameter, select the **AutoRestart** check box. When you enable **Dynamic file name enabled**, configure **Filename** to create incrementally numbered file names for the

## Scope (xPC)

---

multiple log files. Failure to do so causes an error when you try to start the scope.

You can enable the creation of up to 99999999 files (<%%%%%%%%>.dat). The length of a file name, including the specifier, cannot exceed eight characters.

### **Max file size in bytes (multiple of WriteSize)**

Provide the maximum size of **Filename**, in bytes. This value must be a multiple of **WriteSize**. Default is 536870912.

When the size of a log file reaches **Max file size in bytes (multiple of WriteSize)**, the software creates a subsequently numbered file name, and continues logging data to that file, up until the highest log file number you have specified. If the software cannot create any additional log files, it overwrites the first log file.

**Purpose** Software Reboot

**Library** xPC Target Library for Misc.

**Description** You can use the Software Reboot block to reboot the target computer when your simulation reaches a certain state. For example, if your control system becomes unstable you may want to reboot the target computer.

This block has one input. The input is the reboot signal and accepts the following values:

- 1 — Reboots the target computer.
- 0 — If you use a Software Reboot block and you don't want the target computer 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 “Run Confidence Test on Configuration”.

---

**Block  
Parameters**

**Sample time**

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

# To xPC Target

---

**Purpose** Data to target computer

**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 a MATLAB 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.

The To xPC Target block runs in nonreal time on the host computer. It is completely asynchronous to the model running in real time on the target computer. Therefore, in the case of continuously changing input to the To xPC Target block, it is possible for two parameter updates to be sent to the target computer before the next sample run of the target application. In this case, the target application only uses the last parameter value received. Conversely, it is also possible for one or more sample time(s) to elapse on the target computer before the To xPC Target block sends the next parameter value.

In either case, the To xPC Target block only sends parameter values to the target computer when there is a change (for example, the input of the To xPC Target block changes).

---

**Note** The use of To xPC Target blocks requires a connection between the host and target computers. If the host and target computer are not connected, operations such as opening a model that contains these blocks or copying these blocks within or between models 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 Simulink Coder to build and download the target application to the default target computer. This assumes that you configured a default target computer through the xPC Target Explorer (see sections “Serial Communication Setup” and “Network Communication Setup” 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 computer.

## See Also

“Creating a Custom Graphical Interface”

# XCP Server

---

<b>Purpose</b>	XCP server
<b>Library</b>	xPC Target Library for Misc.
<b>Description</b>	This block enables an XCP Server process to run in the target application as a background process.
<b>Block Parameters</b>	<b>Target Address</b> Target IP address for target computer. The default value is <code>getxpcenv('TcpIpTargetAddress')</code> . Typically, you will want to leave the default entry. Otherwise, enter the TCP IP address for the target computer.
	<b>Server Port</b> Port for communication between target computer and XCP server. The default value is 5555. This value must be the same as the port number you specify for the Vector CANape device.

# xPC Target Get Free Stack Size

---

<b>Purpose</b>	xPC Target Get Free Stack Size
<b>Library</b>	xPC Target Library for Misc.
<b>Description</b>	This block outputs the number of bytes of stack memory currently available to the target application thread.
<b>See Also</b>	xPC Target Get Minimal Free Stack Size

# xPC Target Get Minimal Free Stack Size

---

**Purpose** xPC Target Get Minimal Free Stack Size

**Library** xPC Target Library for Misc.

**Description** This block outputs the number of bytes that have never been used in the stack since the thread was created.

---

**Note** The underlying function traverses the entire stack to find unused bytes. For performance reasons, this block should be used only for diagnostic purposes.

---

**See Also** xPC Target Get Free Stack Size

# xPC Target Get Overload Counter

---

<b>Purpose</b>	Return current CPU overload count
<b>Library</b>	xPC Target Library for Misc.
<b>Description</b>	This block returns the CPU overload count. When you connect this block to an xPC Target scope (Scope (xPC)), you might need to adjust the xPC Target scope block parameter <b>Number of samples</b> to a smaller number, such as 3.
<b>Block Parameters</b>	<b>Sample time</b> Enter a base sample time or a multiple of the base sample time.
<b>See Also</b>	xPC Target Set Overload Counter

# xPC Target Set Overload Counter

---

<b>Purpose</b>	Set current CPU overload count
<b>Library</b>	xPC Target Library for Misc.
<b>Description</b>	The block enables you to adjust the CPU overload count.
<b>Block Parameters</b>	<b>Sample time</b> Enter a base sample time or a multiple of the base sample time.
<b>See Also</b>	xPC Target Get Overload Counter

**Purpose** xPC Target TET

**Library** xPC Target Library for Misc.

**Description** This block outputs the Task Execution Time (TET) in seconds.  
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.

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

# xPC Target Time

---

<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 computer.</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 computer 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 utility blocks support miscellaneous functions. Some of these blocks exist in the Utilities library, available at the top-level of the xPC Target Block Library. Others are available as sublibraries of the I/O function they support. For example, the Rollover Counter block is located in the Incremental Encoder/Utilities sublibrary.

## **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.

**Purpose** Deconstruct data frames

**Library** xPC Target Library for Utilities

**Description** 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.

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.

## **Block Parameters**

### **Bit Patterns**

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 **Input port (packed) data types** specify the data type of the input port.

### **Input port (packed) data types**

From the list, select an input port (packed) data type.

- double
- single
- int8
- uint8
- int16
- uint16
- int32
- uint32
- boolean

# 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. 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 sign extension. This functionality enables you to pack and unpack negative numbers without losing precision.

<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></p> <p>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></p> <p>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. Each element in the list of signals starts on a boundary specified by the alignment relative to the start of the vector.

**Purpose** Deconstruct data frames

**Library** xPC Target Library for Utilities

**Description** This block is the exact analog of the Pack block. It can receive a vector of one of the following types:

- double
- single
- int8
- uint8
- int16
- uint16
- int32
- uint32
- boolean

It outputs various Simulink data types in different sizes.

## **Block Parameters**

### **Output port (unpacked) data types (cell array)**

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

- double
- single
- int8
- uint8
- int16
- uint16

# 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. 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 <code>uint32</code> (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 <code>uint32</code> 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. 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 sign extension. This functionality enables you to pack and unpack negative numbers without losing precision. 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>	<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.
	<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.  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, $2^{(n-1)}$ ), where $n$ is the size of the register in bits.
<b>See Also</b>	Extended Counter

# Shared Memory Pack

---

**Purpose** Shared memory pack

**Library** xPC Target Library for Shared Memory

**Description** 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. The Simulink interface is not aware of structures; you must pass the output of each structure segment as input to the Shared Memory Pack block.

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:

- “Shared Memory Structure Reference” on page 17-18
- “Shared Memory Structure Reference” on page 17-18
- “Shared Memory Structure Reference” on page 17-5

Memory partitions consist of groups of Simulink signals, which are combined into blocks (packets) of 32-bit words.

This block ignores the **Address** field of the partition structure.

**Block Parameters** **Partition struct**  
Enter the name of the predefined shared memory partition structure.

**See Also** Shared Memory Unpack  
“Shared Memory Structure Reference” on page 17-18  
“Shared Memory Structure Reference” on page 17-18  
“Shared Memory Structure Reference” on page 17-5

<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>• “Shared Memory Structure Reference” on page 17-18</li><li>• “Systran Shared Memory Structure Reference” on page 17-26</li><li>• “Shared Memory Structure Reference” on page 17-5</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>“Shared Memory Structure Reference” on page 17-18</p> <p>“Shared Memory Structure Reference” on page 17-18</p> <p>“Shared Memory Structure Reference” on page 17-5</p>

# Shared Memory Unpack

---

# Obsolete Drivers

---

The following drivers are obsolete in the xPC Target product:

- “xPC Target Library of Obsolete Drivers” on page 49-2
- “Obsolete Blocks and Boards — Alphabetical List” on page 49-4

## xPC Target Library of Obsolete Drivers

You can access obsolete driver blocks by typing the following command at the MATLAB Command Window:

```
xpcobsoletelib
```

---

**Note** Existing models that use obsolete drivers will still work. However, they may not be supported in future releases. Do not use these driver blocks in new models.

---

This library contains obsolete drivers from the following organizations:

- Burr-Brown
- Diamond Systems
- Gespac
- J1939
- Measurement Computing
- National Instruments
- Real Time Devices
- Sensoray

The following drivers have been updated. Use the new version of these drivers.

- Measurement Computing drivers
  - Measurement Computing CIO-QUAD02 Incremental Encoder
  - Measurement Computing CIO-QUAD04 Incremental Encoder
  - Measurement Computing PCI-QUAD04 Incremental Encoder
- J1939 drivers
  - J1939 Database Setup
  - J1939 Message Triggering

- J1939 Pack
- J1939 Receive
- J1939 Transmit
- J1939 Unpack

You can use the `xpcobsoletelib` command to access the previous versions of these drivers. These versions of these drivers may not be supported in future releases. Do not use these driver blocks in new models.

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

**Purpose** Burr-Brown PCI-20003M board (obsolete)

**Board** Burr-Brown PCI-20003M

**General Description** The PCI-20003M is an I/O board with two analog output (D/A) channels (12-bit).

The xPC Target product 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 (obsolete)

**Library** Obsolete xPC Target Block Library

<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.

<b>Input Range (V)</b>	<b>Range Code</b>
-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 (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 (for example 0xd000)**

Enter the base address of the carrier 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-20019M

---

**Purpose**                      Obsolete Burr-Brown PCI-20019M board (obsolete)

**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).

The xPC Target product 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)

<b>Board Characteristics</b>	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 (obsolete)

**Library** Obsolete xPC Target Block Library

<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.

### Range vector

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
-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 (for example 0xd000)

Enter the base address of the carrier 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 the xPC Target software.

Jumper Number	Jumper
W6	out
W8	in
W10	out
W11	in
W12	out
W22	out
W27	out
W30	-
W31	-

**Purpose** Burr-Brown PCI-20023M board (obsolete)

**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).

The xPC Target product 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)

<b>Board Characteristics</b>	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 (obsolete)

**Library** Obsolete xPC Target Block Library

<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.

### Range vector

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 the xPC Target software.

# 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 (for example 0xd000)

Enter the base address of the I/O carrier 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 the xPC Target software.

Jumper Number	Jumper
W6	out
W8	in
W9	-
W10	out
W11	in
W12	out
W27	out
W30	-
W31	-

# Burr-Brown PCI-20041C

---

**Purpose** Burr-Brown PCI-20041C board (obsolete)

**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.

The xPC Target product 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 (obsolete)

**Library** Obsolete xPC Target Block Library

**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 (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 (for example 0xd000)

Enter the base address of the carrier 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 (obsolete)

**Library** Obsolete xPC Target Block Library

**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.

### **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 (for example 0xd000)**

Enter the base address of the carrier board. This entry must correspond to the DIP switch settings on the board. For example, if the base address is d000 (hexadecimal), enter

0xd000

**Purpose** Burr-Brown PCI-20098C board (obsolete)

**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.

The xPC Target product 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 (obsolete)

**Library** Obsolete xPC Target Block Library

<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 vector**

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 (for example 0xd000)**

Enter the base address of the carrier 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 (obsolete)

**Library** Obsolete xPC Target Block Library

**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**

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 (for example 0xd000)**

Enter the base address of the carrier 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 (obsolete)

**Library** Obsolete xPC Target Block Library

**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.

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

### Port

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 (for example 0xd000)

Enter the base address of the carrier 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

---

**Purpose** Diamond Systems Quartz-MM-10 board (obsolete)

**Board** Diamond Quartz-MM-10

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

The xPC Target product supports this board with these driver blocks:

- 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 Counter PWM

**Purpose** Quartz-MM-10 Counter PWM block (obsolete)

**Library** Obsolete xPC Target Block Library

**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.

## 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. One block is required for each counter used.

### 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 0.25. 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 (obsolete)

**Library** Obsolete xPC Target Block Library

**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.

## 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. One block is required for each counter used.

### 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 0.25. 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**.  $400 \text{ kHz} \times 0.04375 = 17.5 \text{ kHz}$

# Diamond Quartz-MM-10 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)

## 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 (obsolete)

**Library** Obsolete xPC Target Block Library

**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. One block is required for each counter used.

**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 1-10 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 (obsolete)

**Library** Obsolete xPC Target Block Library

**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.

## 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. One block is required for each counter used.

### 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 1-10 for details.

---

# Diamond Quartz-MM-10 Counter FM & 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 **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 (obsolete)

**Library** Obsolete xPC Target Block Library

**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.

## 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, 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 (obsolete)

**Library** Obsolete xPC Target Block Library

**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.

## Scaling Input to Output

### Hardware Output

TTL

### Block Input Data Type

Double

### Scaling

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. One block is required for each counter used.

### 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

# Diamond Quartz-MM-5

---

**Purpose** Diamond Systems Quartz-MM-5 board (obsolete)

**Board** Diamond Quartz-MM-5

**General Description** The Quartz-MM 5 has 8 digital input lines, 8 digital output lines.  
The xPC Target product supports this board with these driver blocks:

- 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 Counter PWM

**Purpose** Quartz-MM-5 Counter PWM block (obsolete)

**Library** Obsolete xPC Target Block Library

**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.

## 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, or 4 to select which counter is used with this driver block. One block is required for each counter used.

### 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 0.25. 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 1-10 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 (obsolete)

**Library** Obsolete xPC Target Block Library

**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.

## 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. One block is required for each counter used.

### 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 0.25. 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 & 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)

## 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 (obsolete)

**Library** Obsolete xPC Target Block Library

**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. One block is required for each counter used.

**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 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 & ARM

**Purpose** Quartz-MM-5 Counter FM & ARM block (obsolete)

**Library** Obsolete xPC Target Block Library

**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. One block is required for each counter used.

### 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 1-10 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 (obsolete)

**Library** Obsolete xPC Target Block Library

**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.

## 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, 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 (obsolete)

**Library** Obsolete xPC Target Block Library

**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. One block is required for each counter used.

### **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

# Diamond Quartz-MMxx

---

**Purpose** Diamond Quartz-MMxx block (obsolete)

**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.

**Purpose** Obsolete Gespac GESADA-1 board

**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).

The xPC Target product supports this board with these driver blocks:

- Gespac GESADA-1 Analog Input (A/D)
- Gespac GESADA-1 Analog Output (D/A)

---

**Note** The xPC Target product does not support the external trigger and interrupt propagation on this board.

---

<b>Board Characteristics</b>	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 (obsolete)

**Library** Obsolete xPC Target Block Library

<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 (obsolete)

**Library** Obsolete xPC Target Block Library

<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.

<b>Input Range (V)</b>	<b>Range Code</b>
-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.

# Gespac GESPIA-2A

---

**Purpose** Gespac GESPIA-2A board (obsolete)

**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 Freescale. Each PIA has two ports (A and B) with 8 digital lints which can be defined as input or output.

The xPC Target product 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

**Purpose** GESPIA-2A Digital Input block (obsolete)

**Library** Obsolete xPC Target Block Library

**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 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 (obsolete)

**Library** Obsolete xPC Target Block Library

**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.

### **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

**Purpose** J1939 Database Setup (obsolete driver)

**Library** xPC Target Library for CAN/J1939

**Description** The J1939 Database Setup block identifies the user-supplied database for the J1939 block set. Use one block per model.

**Block Inputs**

None

**Block Outputs**

None

**Block Parameters**

**J1939 database file**

Specify the J1939 database location and file name, for example 'J1939.dbc'.

This file defines the J1939 message set and is in a format defined by Vector Informatik GmbH.

---

**Note** This is a customer-supplied database file. The xPC Target software does not supply this file.

---

# J1939 Message Triggering

---

<b>Purpose</b>	J1939 Message Triggering (obsolete driver)
<b>Library</b>	xPC Target Library for CAN/J1939
<b>Description</b>	The J1939 Message Triggering block manages the multiple trigger conditions possible for triggering the transmission of a J1939 message.

## Block Inputs

This block has the following input ports:

Enable

Receives the signal that indicates when the block is enabled.

Request

Receives the signal that indicates a request to send has been received.

Active

Indicates when a message is in the active state.

Change

Receives the signal whose change triggers the sending of a message.

## Block Outputs

This block has the following output ports:

Message Enable

Signal of type double that is set to 1 when the message should be packed. It is typically connected to the input port of a J1939 Pack block.

## Block Parameters

**Repetition**

From the list, select:

- On Request Only
- Engine Speed Based

- Time Based

## **Repeat on Interval**

Select this check box to repeat at the interval specified in the **Repetition Interval** parameter.

## **Repetition Interval**

Specify the enabled repetition interval in milliseconds. The repetition interval should be an integer multiple of the model update rate.

## **Change Interval on External Active Signal**

Select this check box to enable the message to repeat at the interval specified in **Interval when Active** when the external signal is active.

## **Interval when Active**

Enter the active repetition interval in milliseconds. The interval should be an integer multiple of the model update rate.

## **Send on Change**

Select this check box to send the message when the input signal that is wired to the change port changes by the value in the **Change threshold** parameter.

## **Change Threshold**

Enter the threshold for message transmission to any change (Any) or a change greater than 10% (10%).

## **Minimum Change Interval**

Enter a minimum change interval.

# J1939 Pack

---

**Purpose** J1939 Pack (obsolete driver)

**Library** xPC Target Library for CAN/J1939

**Description** The J1939 Pack block assembles a message suitable for sending over the CAN network.

## Block Inputs

The block input signal indicates when the message should be sent. You can connect this signal to a J1939 Message Triggering block or to a user selected block. When the signal is 1, the block packs a new message from the current signal input values.

The other input ports are dynamic and depend on the value of the **PGN** value of the message. All ports are of type double.

## Block Outputs

The single output is a vector of unsigned bytes of variable length. This output is typically connected to the input of the J1939 Transmit block.

## Block Parameters

**PGN**  
Select the message that this block unpacks. From the list, select a parameter group number (PGN). The block automatically creates input ports according to the number and type of suspect parameter numbers (SPNs) that correspond to the PGN.

**Priority**  
Enter the three priority bits of this message. Enter a message priority from 0 to 7, with 0 being the highest and 7 being the lowest.

**Destination Address**  
Enter the destination address for destination-specific messages. See the SAE J1939-21 for information.

- Purpose** J1939 Receive (obsolete driver)
- Library** xPC Target Library for CAN/J1939
- Description** The J1939 Receive block receives the PGNs.

## Block Inputs

Connect the J1939 Receive block directly to the output of a CAN FIFO Read block. The input signal is a matrix of size  $m \times 6$ , where  $m$  is the FIFO read depth defined in the FIFO Read block dialog box. Each row with its six elements contains all the information defining a CAN message:

- Port
- Identifier
- Event type
- Data frame size
- Timestamp
- Data

See one of the FIFO Read blocks (for example, Softing CAN-AC2-PCI with Philips SJA1000 FIFO Read) for a detailed description of these elements.

## Block Outputs

The number of parameter group numbers (PGNs) in the **PGN list** parameter determines the number of block outputs. The block creates and labels an output port for each PGN in the list. Each output signal is a vector of unsigned bytes with a length of 16. The bytes are defined as follows:

Bit	Description
y[1]	Protocol data unit format (PDU format or PF)
y[2]	Group extension if PF is greater than 240, else destination address (DA)

# J1939 Receive

---

Bit	Description
y[3]	Source address
y[4]	Timestamp[0]
y[5]	Timestamp[1]
y[6]	Timestamp[2]
y[7]	Timestamp[3]
y[8]	Data byte 1
y[9]	Data byte 2
y[10]	Data byte 3
y[11]	Data byte 4
y[12]	Data byte 5
y[13]	Data byte 6
y[14]	Data byte 7
y[15]	Data byte 8

## Block Parameters

### CAN Port

Select the filter criterion for the CAN port. From the list, select Any, 1, or 2.

### PGN List

Enter the acceptance filter criterion for the parameter group number (PGN). Enter a list of PGNs that you want to receive. You can provide a set of PGNs as a row vector.

When the CAN FIFO blocks receives the data that corresponds to these PGNs, they filter these messages and send them to the J1939 Unpack block.

### CA Address

Enter the filter acceptance criterion for the specified controller application ID. Only messages whose destination addresses match

the current controller application address or the global address will be routed to a PGN output.

# J1939 Transmit

---

<b>Purpose</b>	J1939 Transmit (obsolete driver)
<b>Library</b>	xPC Target Library for CAN/J1939
<b>Description</b>	The J1939 Receive block sends the PGNs.

## Block Inputs

The J1939 Transmit block inputs are typically connected directly to one or more J1939 Pack blocks. The output of the J1939 Transmit block is typically connected to the input of a CAN FIFO Send block. The J1939 Transmit block input is of type unsigned 8-bit integer. The bytes are:

- Priority
- Data Page and Reserved Bits
- PDU Format
- PDU Specific
- Destination Address
- Data Frame Size [0]
- Data Frame Size [1]
- New message Flag

## Block Outputs

The J1939 Transmit block has a single output port of type double. The output signal is a matrix of size  $n \times 6$ , where  $n$  is the value in the **Number of input messages** parameter. At this port, you must provide information required to construct valid CAN messages to be written into the transmit FIFO. For each CAN message, you must pass six elements. See the documentation for the FIFO Write block (such as Softing CAN-AC2-PCI with Philips SJA1000 FIFO Write) for a detailed description of these elements.

### Port

The value can be either 1 (port 1) or 2 (port 2) and defines the 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 the CAN message is extended, the range is 0 to  $2^{29}-1$ . J1939 identifiers are always extended. J1939 messages can share the physical network with another protocol using standard identifiers.

## 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. J1939 identifiers are always extended.

## Data frame size

The value can be in the range of 0 to 8 and defines the data frame size of the outgoing CAN message in bytes. The CAN messages generated by J1939 are nearly always 8 bytes in length. The exception is the request message, which has a length of 3 bytes.

## 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.

## Block Parameters

### Port (1 or 2)

Enter the port from which to send the message. Enter a value of 1 (port 1) or 2 (port 2) to define the port the CAN message is sent from.

### Number of input messages

Specify the number of inputs to this block. One input for each unique PGN and destination address is required.

### Source Address

Specify the source address for the messages of the current node.

# J1939 Unpack

---

**Purpose** J1939 Unpack (obsolete driver)

**Library** xPC Target Library for CAN/J1939

**Description** The J1939 Unpack block disassembles a message suitable for sending over the CAN network.

## Block Inputs

Connect the J1939 Receive block directly to the output of a J1939 Unpack block. The input signal is a vector of unsigned bytes with a length of 16. The bytes are defined as follows:

Bit	Description
y[0]	Reserved bit and data page bit
y[1]	Protocol data unit format (PDU format or PF)
y[2]	Group extension if PF is greater than 240, else destination address (DA)
y[3]	Source address
y[4]	Timestamp[0]
y[5]	Timestamp[1]
y[6]	Timestamp[2]
y[7]	Timestamp[3]
y[8]	Data byte 1
y[9]	Data byte 2
y[10]	Data byte 1
y[11]	Data byte 2
y[12]	Data byte 3
y[13]	Data byte 4

Bit	Description
y[14]	Data byte 5
y[15]	Data byte 6

## Block Outputs

The PGN parameter determines the number of block outputs. The number and definition for each parameter is dynamic. For each suspect parameter number (SPN), the block creates an output of type double.

## Block Parameters

### PGN

Set the message that this block unpacks. From the list, select a PGN. The driver automatically creates input ports according to the number and type of SPNs that correspond to the PGN.

### Source

Enter the acceptance filter criterion for the source address.

### Destination

Enter the filter acceptance criterion for the specified destination.  
From the list, select:

- global and specific
- global only
- specific only

# Measurement Computing CIO-CTR05

---

**Purpose** Measurement Computing CIO-CTR05 board (obsolete)

**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.

The xPC Target product 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 (obsolete)

**Library** Obsolete xPC Target Block Library

**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. One block is required for each counter used.

### 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 0.25. 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 CIO-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)

## 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 (obsolete)

**Library** Obsolete xPC Target Block Library

**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.

## 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. One block is required for each counter used.

### 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 0.25. 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 CIO-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)

## 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 (obsolete)

**Library** Obsolete xPC Target Block Library

**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. One block is required for each counter used.

### **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 1-10 for details.

---

### **Level sequence of square wave**

From the list, choose either high-low or low-high:

# Measurement Computing CIO-CTR05 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

# Measurement Computing CIO-CTR05 Counter FM & ARM

**Purpose** CIO-CTR05 Counter FM & ARM block (obsolete)

**Library** Obsolete xPC Target Block Library

**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.

## 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. One block is required for each counter used.

### 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 1-10 for details.

---

# Measurement Computing CIO-CTR05 Counter FM & 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 **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 (obsolete)

**Library** Obsolete xPC Target Block Library

**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. One block is required for each counter used.

### 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 (obsolete)

**Library** Obsolete xPC Target Block Library

**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. One block is required for each counter used.

### **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

**Purpose** CIO-CTR10 counter/timer board (obsolete)

**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.

The xPC Target block library 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-CTR10
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 (obsolete)

**Library** Obsolete xPC Target Block Library

**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.

## 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. One block is required for each counter used.

### 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.

### Relative output frequency

Enter a value between 0 and 0.25. 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 CIO-CTR10 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)

## 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 (obsolete)

**Library** Obsolete xPC Target Block Library

**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.

<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. One block is required for each counter used.

### **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.

### **Relative output frequency**

Enter a value between 0 and 0.25. 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 (obsolete)

**Library** Obsolete xPC Target Block Library

**Note** The CIO-CTR10 has two AM9513A chip with ten counters.  
The CIO-CTR10 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. One block is required for each counter used.

### **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-CTR10 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 1-10 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 (obsolete)

**Library** Obsolete xPC Target Block Library

**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.

## 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. One block is required for each counter used.

### 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.

### 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 1-10 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 (obsolete)

**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. One block is required for each counter used.

**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 (obsolete)

**Library** Obsolete xPC Target Block Library

**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. One block is required for each counter used.

### **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).

### **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

# Measurement Computing CIO-CTR10 Frequency Capture

---

0x300

# Measurement Computing CIO-CTRxx

---

<b>Purpose</b>	CIO-CTRxx block (obsolete)
<b>Library</b>	Obsolete xPC Target Block Library
<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.

# Measurement Computing CIO-QUAD02 Incremental Encoder

<b>Purpose</b>	CIO-QUAD02 Incremental Encoder (obsolete driver)
<b>Library</b>	Obsolete xPC Target Block Library
<b>Description</b>	<p>This driver block has three block outputs: <b>Angle</b>, <b>Turns</b>, and <b>Init</b>.</p> <p>You can use <b>Init</b> to determine when the block output values are valid. <b>Init</b> is first set to 0. When the encoder reached the first index, <b>Init</b> is set to 1. From then on, you can determine the exact position, direction, and velocity. <b>Init</b> remains 1 unless <b>Counter Reset by Index</b> is set to <b>First Only</b>, and the counter detects a rollover. For more information, see “Block Parameters” on page 49-93.</p> <p><b>Turns</b> is the number of complete revolutions made by the encoder. <b>Angle</b> is the amount the encoder turns since the last full revolution.</p> <p>The distance is given by:</p> $\text{distance} = 2 * \pi * \text{Turns} + \text{Angle}$ <p>The velocity is given by:</p> $\text{velocity} = (\text{distance}(t_s) - \text{distance}(t_s-1)) / t_s$ <p>The direction is given by:</p> $\text{direction} = \text{distance}(t_s) - \text{distance}(t_s-1)$ <p>A negative value is reverse, while a positive value is forward.</p>
<b>Block Parameters</b>	<p><b>Function module</b></p> <p>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.</p> <p><b>Counter Reset by Index</b></p> <p>From the list choose either <b>Only First</b>, or <b>Continuous</b>.</p> <p>If you choose <b>Only First</b>, the first time the encoder reaches the index, the counter is reset to zero, the <b>Init</b> output signal is set</p>

# 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 valid 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 (obsolete driver)

**Library** Obsolete xPC Target Block Library

**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 49-93.

**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:

$$\text{direction} = \text{distance}(t_s) - \text{distance}(ts-1)$$

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 valid 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

<b>Purpose</b>	PCI-QUAD04 Incremental Encoder block (obsolete driver)
<b>Library</b>	Obsolete xPC Target Block Library
<b>Description</b>	<p>This driver block has three block outputs: <b>Angle</b>, <b>Turns</b>, and <b>Init</b>.</p> <p>You can use <b>Init</b> to determine when the block output values are valid. <b>Init</b> is first set to 0. When the encoder reached the first index, <b>Init</b> is set to 1. From then on, you can determine the exact position, direction, and velocity. <b>Init</b> remains 1 unless <b>Counting reset by index</b> is set to <b>Only First</b>, and the counter detects a rollover. For more information, see “Block Parameters” on page 49-93.</p> <p><b>Turns</b> is the number of complete revolutions made by the encoder. <b>Angle</b> is the amount the encoder turns since the last full revolution.</p> <p>The distance is given by:</p> $\text{distance} = 2 * \pi * \text{Turns} + \text{Angle}$ <p>The velocity is given by:</p> $\text{velocity} = (\text{distance}(t_s) - \text{distance}(t_s-1)) / t_s$ <p>The direction is given by:</p> $\text{direction} = \text{distance}(t_s) - \text{distance}(t_s-1)$ <p>A negative value is reverse, while a positive value is forward.</p>
<b>Block Parameters</b>	<p><b>Function module</b> 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.</p> <p><b>Counting reset by index</b> From the list choose <b>Only First</b>, <b>Continuous</b>, or <b>Index input disabled</b>.</p>

# 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 valid 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.

# Measurement Computing PCI-CTR05

---

**Purpose** Measurement Computing PCI-CTR05 board (obsolete)

**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.

The xPC Target product 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

## Board Characteristics

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 (obsolete)

**Library** Obsolete xPC Target Block Library

**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. One block is required for each counter used.

### **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 0.25. 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 (obsolete)

**Library** Obsolete xPC Target Block Library

**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. One block is required for each counter used.

### 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 0.25. 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 (obsolete)

**Library** Obsolete xPC Target Block Library

**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. One block is required for each counter used.

**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 1-10 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 (obsolete)

**Library** Obsolete xPC Target Block Library

**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. One block is required for each counter used.

### 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 1-10 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 (obsolete)

**Library** Obsolete xPC Target Block Library

**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. One block is required for each counter used.

### 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 (obsolete)

**Library** Obsolete xPC Target Block Library

**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.

## 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. This selects which counter the driver block uses to determine the FM. One block is required for each counter used.

### 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 (obsolete)

**Library** Obsolete xPC Target Block Library

**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.

# National Instruments PC-TIO-10

---

**Purpose** National Instruments PC-TIO-10 board (obsolete)

**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).

The xPC Target product supports this board with these driver blocks:

- 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-TIO10 Counter FM & ARM
- National Instruments PC-TIO10 PWM Capture
- National Instruments PC-TIO10 FM Capture
- National Instruments PC-TIO-10xx

## Board Characteristics

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 Counter PWM

**Purpose** PC-TIO-10 Counter PWM block (obsolete)

**Library** Obsolete xPC Target Block Library

**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.

## Scaling Input to Output

Hardware Input	Block Output Data Type	Scaling
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. One block is required for each counter used.

### 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 between 0 and 0.25. 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**.  $100 \text{ kHz} \times 0.175 = 17.5 \text{ kHz}$

# National Instruments PC-TIO-10 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)

## 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 (obsolete)

**Library** Obsolete xPC Target Block Library

**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.

## Scaling Input to Output

Hardware Input	Block Output 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, 5, 6, 7, 8, 9, or 10 to select which counter is used with this driver block. One block is required for each counter used.

### 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 between 0 and 0.25. 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 (obsolete)

**Library** Obsolete xPC Target Block Library

**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.

## Scaling Input to Output

Hardware Input	Block Output Data Type	Scaling
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. One block is required for each counter used.

### 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 1-10 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 (obsolete)

**Library** Obsolete xPC Target Block Library

**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. One block is required for each counter used.

### 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 1-10 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 (obsolete)

**Library** Obsolete xPC Target Block Library

**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, 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 PC-TIO10 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 (obsolete)

**Library** Obsolete xPC Target Block Library

**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.

## Scaling Input to Output

### Hardware Output

TTL

### Block Input Data Type

Double

### Scaling

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. One block is required for each counter used.

### 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 PC-TIO10 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

**Purpose**

National Instruments PC-TIO-10xx (obsolete)

**Library**

Obsolete xPC Target Block Library

**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.

# Real Time Devices DM6804

---

**Purpose** Real Time Devices DM6804 board (obsolete)

**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.

The xPC Target product supports this board with these driver blocks:

- 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 Counter PWM

**Purpose** DM6804 Counter PWM block (obsolete)

**Library** Obsolete xPC Target Block Library

**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. One block is required for each counter used.

### 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 0.25. 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:

# Real Time Devices DM6804 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 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 (obsolete)

**Library** Obsolete xPC Target Block Library

**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. One block is required for each counter used.

### 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 0.25. 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 (obsolete)

**Library** Obsolete xPC Target Block Library

**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. One block is required for each counter used.

### 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 1-10 for details.

---

### Level sequence of square wave

From the list, choose either high-low or low-high:

# Real Time Devices DM6804 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

# Real Time Devices DM6804 Counter FM & ARM

**Purpose** DM6804 Counter FM & ARM block (obsolete)

**Library** Obsolete xPC Target Block Library

**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:	<0.5 disarmed
	Double	≥0.5 armed
	Arm: Double	

## Block Parameters

### Counter

From the list, choose 1, 2, 3, 4, or 5 to select which counter is used with this driver block. One block is required for each counter used.

### 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.

# Real Time Devices DM6804 Counter FM & ARM

---

---

**Note** Your control over the output duty cycle is not always precise. See “PWM and FM Driver Block Notes” on page 1-10 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 (obsolete)

**Library** Obsolete xPC Target Block Library

**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. One block is required for each counter used.

### 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 (obsolete)

**Library** Obsolete xPC Target Block Library

**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.

## 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. This selects which counter the driver block uses to determine the FM. One block is required for each counter used.

### 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 correspond to the DIP-switch settings on the board. For example, if the base address is 300 (hexadecimal), enter

0x300

# Real Time Devices DM6804xx

---

**Purpose** DM6804xx block (obsolete)

**Library** Obsolete xPC Target Block Library

**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.

**Purpose** Sensoray 626 board (obsolete)

**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.

The xPC Target product 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

# Sensoray 626 Initialize

---

**Purpose** Sensoray 626 Initialize block (obsolete)

**Library** Obsolete xPC Target Block Library

**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.

**Purpose** Sensoray 626 Analog Input block (obsolete)

**Library** Obsolete xPC Target Block Library

**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.

<b>Purpose</b>	Sensoray 626 Analog Output block (obsolete)
<b>Library</b>	Obsolete xPC Target Block Library
<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 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</p> <p>[ 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 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.</p>

# Sensoray 626 Analog Output

---

## **Sample time**

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

<b>Purpose</b>	Sensoray 626 Digital Input block (obsolete)
<b>Library</b>	Obsolete xPC Target Block Library
<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 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</p> <p>[ 1, 2, 3 ]</p> <p>Number the channels beginning with 1 even though the board manufacturer numbers them beginning with 0.</p> <p><b>Sample time</b></p> <p>Base sample time or a multiple of the base sample time.</p>

# Sensoray 626 Digital Output

---

**Purpose** Sensoray 626 Digital Output block (obsolete)

**Library** Obsolete xPC Target Block Library

**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 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

[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 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.

## **Sample time**

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

# Sensaray 626 Encoder

---

**Purpose** Sensaray 626 Encoder block (obsolete)

**Library** Obsolete xPC Target Block Library

**Block Parameters** **Board**  
From the list, select 1, 2, 3, or 4 to represent which of the physical Sensaray 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.

- 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.

# Sensaray 626 PWM Capture

---

**Purpose** Sensaray 626 PWM Capture block (obsolete)

**Library** Obsolete xPC Target Block Library

**Block Parameters**

**Board**  
From the list, select 1, 2, 3, or 4 to represent which of the physical Sensaray 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

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.

# Sensoray 626 PWM Capture

---

# Serial Communications Support with Internal Drivers

---

- “RS-232/422/485 Internal Blocks” on page A-2
- “Blocks — Alphabetical List” on page A-3

## RS-232/422/485 Internal Blocks

This topic describes the internal blocks and subsystems 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 subsystems for your use.

---

**Note** If you do not need to modify the subsystems, you should not need to use these blocks directly. For serial communications, use the composite serial communications drivers described in “RS-232/422/485 Drivers (Composite)” on page 4-6.

---

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.

## **Blocks – Alphabetical List**

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

---

**Purpose** RS-232/422/485 Enable TX Interrupt block

**Library** xPC Target Library for RS232

**Description** The Enable TX Interrupt block enables the transmitter buffer empty interrupt when data is present in the software FIFO.

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.

## **Block Parameters**

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

Enter the base address of the UART for which you want to enable the transmitter buffer empty interrupt.

**Port** (Quatech, Commtech)

From the list, choose a port. This parameter specifies the input port for which this block enables the interrupt.

**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 Filter Interrupt Reason (Composite)

---

<b>Purpose</b>	RS-232/422/485 Filter Interrupt Reason block
<b>Library</b>	xPC Target Library for RS232
<b>Description</b>	<p>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.</p> <p>This block is used exclusively inside the interrupt service subsystem for this board.</p>
<b>Block Parameters</b>	<p><b>Port</b> From the list, choose a port. This parameter specifies the port from which this block gets control data.</p> <p><b>Filter value</b> 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.</p> <p>Note that Modem status change currently has no effect because the interrupt is never enabled.</p>

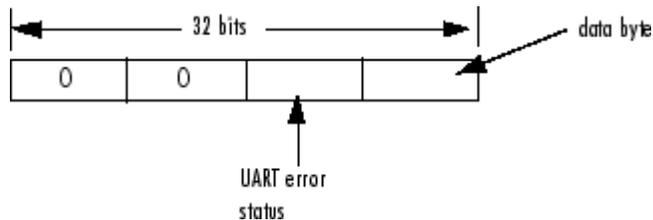
# RS-232/422/485 Read Hardware FIFO (Composite)

**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-232/422/485 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>

## 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 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 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 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 board documentation for further information.

# RS-232/422/485 Write Hardware FIFO (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
```