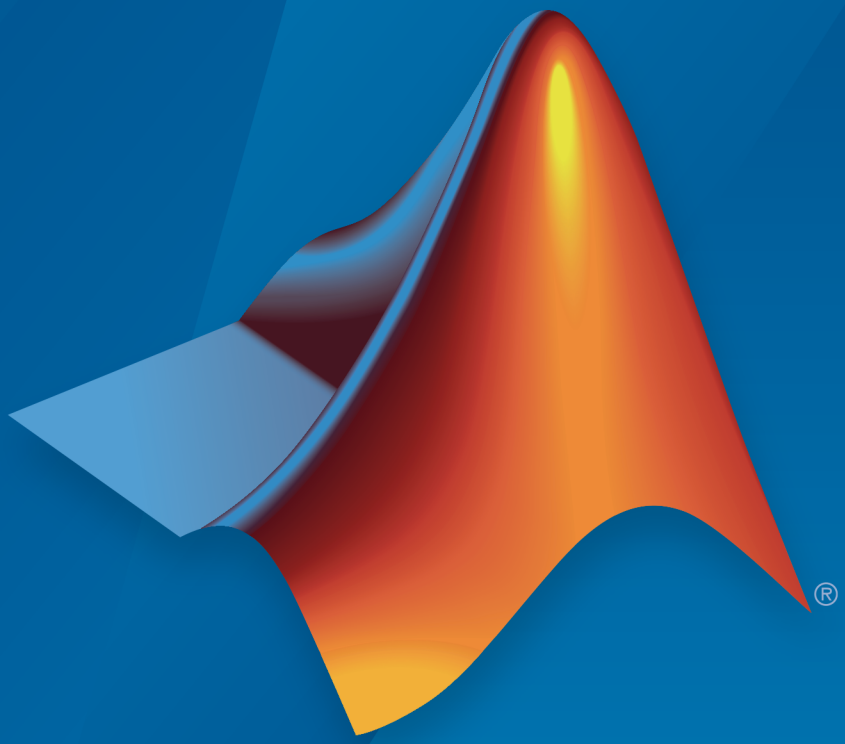
Simulink® Real-Time™

I/O Reference



MATLAB® & SIMULINK®

R2017a

MathWorks®

How to Contact MathWorks

Latest news: www.mathworks.com

Sales and services: www.mathworks.com/sales_and_services

User community: www.mathworks.com/matlabcentral

Technical support: www.mathworks.com/support/contact_us

Phone: 508-647-7000

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

Simulink® Real-Time™ I/O Reference

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)
March 2013	Online only	Revised for Version 5.4 (Release 2013a)
September 2013	Online only	Revised for Version 5.5 (Release 2013b)
March 2014	Online only	Revised for Version 6.0 (Release 2014a)
October 2014	Online only	Revised for Version 6.1 (Release 2014b)
March 2015	Online only	Revised for Version 6.2 (Release 2015a)
September 2015	Online only	Revised for Version 6.3 (Release 2015b)
March 2016	Online only	Revised for Version 6.4 (Release 2016a)
September 2016	Online only	Revised for Version 6.5 (Release 2016b)
March 2017	Online only	Revised for Version 6.6 (Release 2017a)

Contents

3 Serial Communications Support: Blocks

A Serial Communications Support: Internal Blocks

CAN, Encoders, Ethernet, EtherCAT

4 CAN I/O Support

9 Model-Based Ethernet Communications Support

10 Ethernet Blocks

11 Network Buffer Library for Model-Based Ethernet Communications Support

12 Network Buffer Library Blocks

13 Model-Based EtherCAT Communications Support

14 EtherCAT Blocks

TCP, UDP

15 Real-Time TCP Communication Support

22 Precision Time Protocol Blocks

23 SAE J1939

24 SAE J1939 Blocks

25 Shared Memory Support

Video, XCP

26 Video Image Processing

52 Asynchronous Event: Blocks

Utility Drivers, Target Management, Displays and Logging

53 Utility Blocks

54 Target Management, Display, and Logging Blocks

Drivers No Longer Recommended for Use

55 Simulink Real-Time Drivers No Longer Recommended for Use

Introduction, RS-232

Simulink Real-Time I/O Library

I/O Driver Blocks

In this section...

The Simulink Real-Time environment is a solution for prototyping and testing real-time systems using a desktop computer. To support this solution, the software allows you to add I/O blocks to your model. The blocks of the Simulink Real-Time library provides a particular function of an I/O module. By using I/O blocks in your model, you can generate executable code tuned specifically to your I/O requirements.

You add I/O driver blocks to your Simulink model to connect your model to I/O modules (I/O boards). These I/O modules then connect to the sensors and actuators in the physical system.

Speedgoat I/O Modules

Speedgoat real-time target machines are available with various I/O modules. See “Speedgoat I/O Modules” on page 47-3.

Third-Party Driver Blocks

In addition to the blocks contained in the Simulink Real-Time library, you can also use third-party driver blocks in your Simulink Real-Time model. The description of these blocks is beyond the scope of the Simulink Real-Time 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 Simulink Real-Time 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 Simulink Real-Time software.

Note, if your model contains I/O blocks, take I/O latency values into account for the model sample time. To find latency values for a board supported by the Simulink Real-Time block library, consult the vendor data sheet. To find a link to the vendor website, see:

`www.mathworks.com/products/simulink-real-time/supported/hardware-drivers.html`.

To find latency values for Speedgoat boards, contact Speedgoat technical support.

The Simulink Real-Time system supports PCI and ISA (PC/104) buses. If the bus type is not indicated in the driver block number, determine the bus type of the block by examining the block parameter dialog box. The last parameter is either a PCI slot, for PCI boards, or a base address, for ISA (PC/104) boards.

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

This library also contains the following blocks:

- Simulink Real-Time Driver Examples — When you double-click this block, the **Demos** tab in the MATLAB Help Navigator opens, displaying the Simulink Real-Time examples and example groups.
- Help for Simulink Real-Time — When you double-click this block, the Simulink Real-Time roadmap page is displayed. You can access the Simulink Real-Time documentation with this block.

Note: The Simulink Real-Time documentation describes only the Simulink Real-Time 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 displays the

I/O device driver blocks for the specified I/O functionality (for example, A/D, D/A, Digital Inputs, and Digital Outputs).

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

- Sample time
- Number of channels
- Voltage range
- PCI slot (PCI boards)
- Base address (ISA/104 boards)

Memory-Mapped Devices

Simulink Real-Time reserves a 112 kB memory space for memory-mapped devices in the address range:

```
C0000 - DBFFF
```

Drivers for some memory-mapped devices, such as the Softing CAN-AC2-104 board, support an address range higher than the range that Simulink Real-Time supports. Select an address range supported by both the device driver and the Simulink Real-Time software.

ISA Bus I/O Devices

There are two types of ISA boards:

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

The Simulink Real-Time 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 Simulink Real-Time 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 the PCI devices in the target system. You do not need to define base address information in the dialog boxes of the drivers.

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 cannot locate this slot in the target computer, your real-time application will generate an error message after downloading.

To set `PCI Slot (-1 Autodetect)` to a value equal to or greater than `0`, you must identify which board you want on the target computer. To identify the board, find the manufacturer identification number (Vendor ID) and board identification number (Device ID) of the boards supported by the I/O library. When the target is booted, the BIOS is executed and the target computer monitor shows parameters for the PCI boards installed on the target computer. For example:

Bus Number	Device Number	Function Number	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 location is known since the Measurement Computing vendor ID is `0x1307` and the device ID is `0xb`. In this case, you can see that the Measurement Computing board is plugged into PCI slot 11 (Device Number), 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.

Simulink Real-Time I/O Driver Structures

Properties for Simulink Real-Time I/O drivers are 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 run time. *Textual* in this context refers to a programming-language-like syntax and style.

The Simulink Real-Time software currently uses a character vector 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 character vector or value>
```

The driver defines the field names. Enter them 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 a large array, direct entry becomes cumbersome 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 Simulink Real-Time 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 Command Window, type

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

- When you move or copy the model file to a new folder, you must also move or copy the script defining the message structure.

During each sample interval, the driver block locates the message structure, interprets the 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 2-10
- "Initialization and Termination CAN Messages" on page 4-10

Simulink Real-Time Support and SimState

You can save complete model simulation states while simulating, on a development computer, a Simulink model that contains some Simulink Real-Time 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 development 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)
- 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

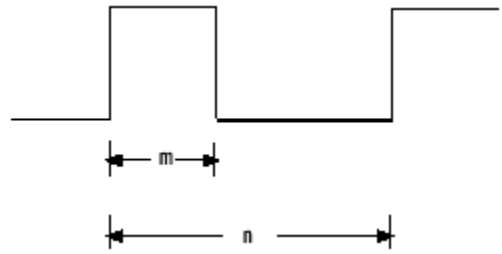
To prevent these messages, clear the **Save complete SimState in final state** check box in the **Data Import/Export** node of the Configuration Parameters dialog box.

PWM and FM Driver Block Notes

In PWM and FM driver blocks, your control over the output frequency and duty cycle is not precise. Although the base frequency value is exact, the way the base frequency is selected affects the output frequency and duty cycle.

At the beginning of each sample time, the block reads the current input signal values. It then computes two unsigned 16-bit integers, `n` and `m`, from the signal values and the block parameters. During the sample time, the block holds the output signal:

1. High for `m` cycles of the base frequency
2. Low for the next `n-m` cycles
3. High for the next `m` cycles
4. . . .



For a base frequency `b`, this algorithm 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 that specifies an output frequency of `b × f`. However, `m` and `n` must be integers. Therefore, it is not always possible to find values of `m` and `n` (duty cycle `m/n = 1/2`) such that:

```
f = b/n
```

exactly and

```
n = 2 * m
```

exactly. 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`.

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 Simulink Real-Time 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 (PC/104).
Access method	Whether the board is memory mapped or I/O mapped.
Multiple block instance support	Whether 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	Whether you can use multiple boards of the same type in one real-time application.

Add I/O Blocks to Simulink Model

Starting with a Simulink model, you can transform it to a Simulink Real-Time model that accesses I/O drivers using the Simulink Real-Time block library. The highest hierarchical level in the library lists I/O function groups. The second level lists board manufacturer groups. The manufacturer groups contain the driver blocks for specific boards.

This example uses the Simulink model `ex_slrt_osc` as an example of how to replace Simulink blocks with Simulink Real-Time I/O blocks (see `matlab: open_system(docpath(fullfile(docroot, 'toolbox', 'xpc', 'examples', 'ex_slrt_osc')))`).

1. In the Command Window, type

   ```
   slrtlib
   ```

   The **Library: slrtlib** 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.

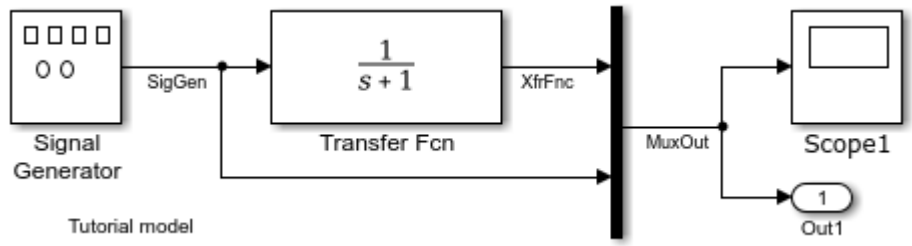
3. Open a manufacturer group. For example, to open the A/D driver blocks from Measurement Computing, double-click the group marked Measurement Computing.

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

4. In the Simulink Editor, type

   ```
   ex_slrt_osc
   ```

   The Simulink block diagram opens for the model `ex_slrt_osc`.

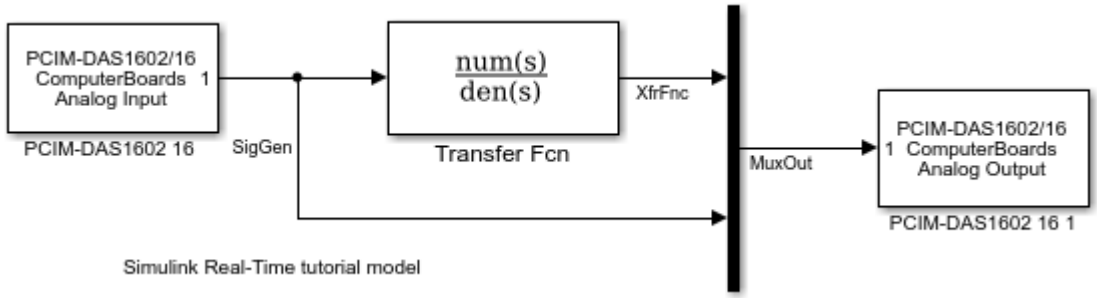


5 From the block library, click and drag the name of an Analog Input block to the Simulink block diagram. Likewise, click and drag the name of an Analog Output block to your model.

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.

7 Save the model with a new name, such as `ex_slrt_iob_osc` (`matlab: open_system(docpath(fullfile(docroot, 'toolbox', 'xpc', 'examples', 'ex_slrt_iob_osc')))`):



You cannot run this model unless the required I/O board is installed in your target computer. However, you can substitute 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 1-13.

Defining I/O Block Parameters

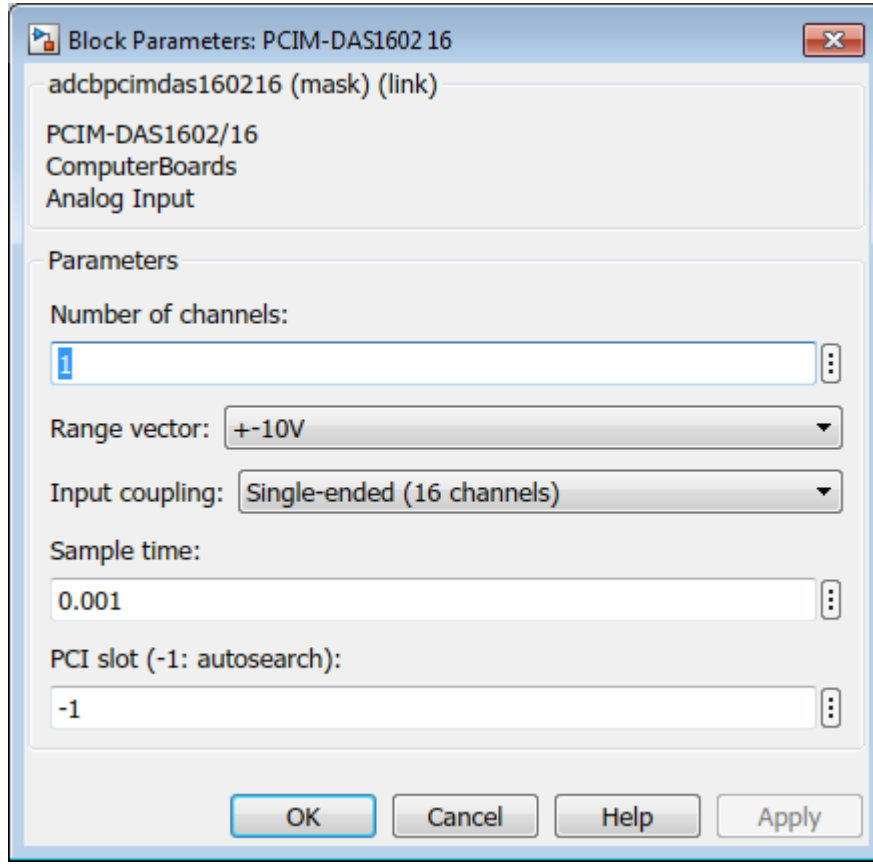
The I/O block parameters define values for your physical I/O boards. For example, I/O block parameters include channel numbers for multichannel boards, input and output voltage ranges, and sample time.

This procedure uses the Simulink model `ex_slrt_osc` (`matlab: open_system(docpath(fullfile(docroot, 'toolbox', 'xpc', 'examples', 'ex_slrt_osc')))`). It assumes that you have added an analog input and an analog output block to your model. To add an I/O block, see "Add I/O Blocks to Simulink Model" on page 1-11.

1. In the Simulink Editor, double-click the input block labeled `Analog Input`.

   The dialog box for the A/D converter opens.

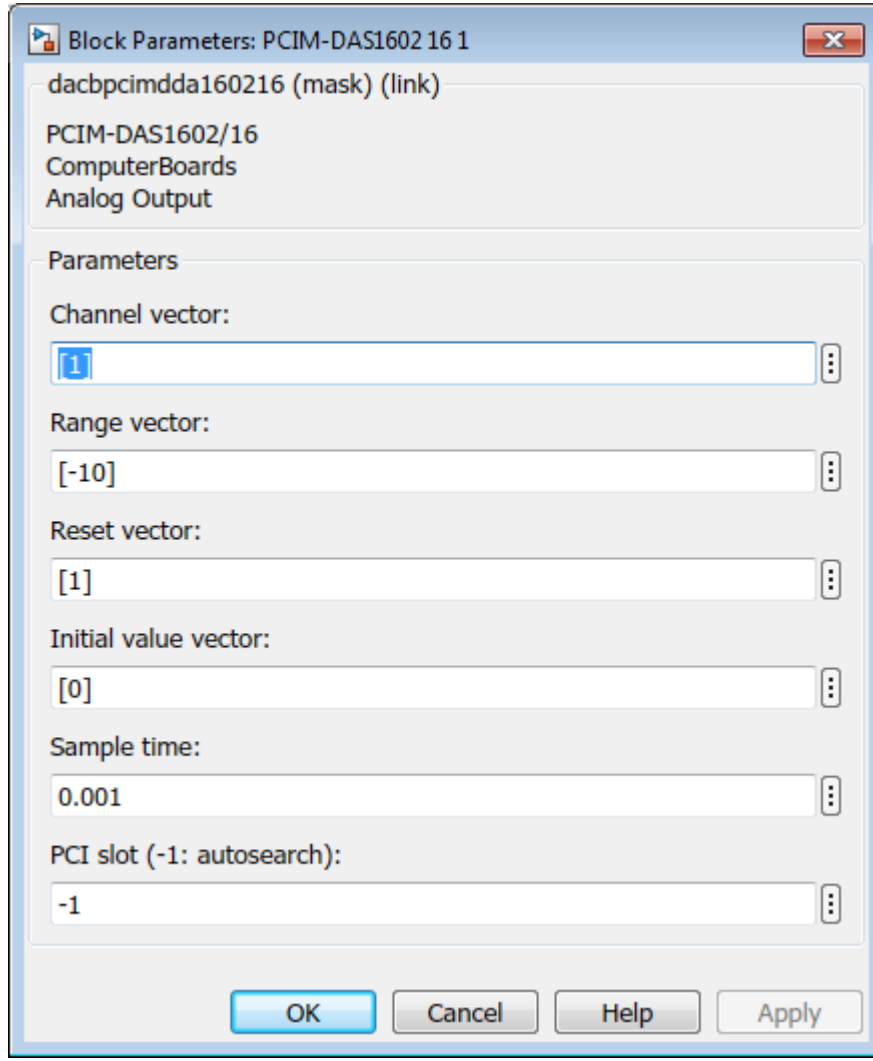
2. Fill in the dialog box. For example, for a single channel enter `1` in the **Number of Channels** box, select `±10 V` for the input range. Select `Single-ended (16 channels)` for the MUX switch position. Enter the same sample time you entered for the fixed step size in the **Solver** pane of the **Simulation**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



3 In the Simulink Editor, 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 PCI slot for this PCI-bus board.



If you change 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.

Serial Communications Support

RS-232 I/O Drivers

The Simulink Real-Time software interfaces the target computer to serial devices using either the COM1 or COM2 port of the main board. It uses Quatech drivers or Commtech drivers.

The Simulink Real-Time software supports RS-232 I/O communication with the following:

- Serial ports on the target computer mainboard
- Third-party Quatech PCI boards manufactured by B&B Electronics (`www.bb-elec.com`)
- Third-party Commtech PCI boards (`fastcomproducts.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 Simulink Real-Time 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 (`fastcomproducts.com`)

The Simulink Real-Time block library provides a set of functionally similar drivers for these boards. See “RS-232/422/485 Simulink Block Reference” on page 2-10 for a description of the driver blocks that support the different protocols.

The Simulink Real-Time block library supplies composite drivers to support RS-232 I/O communication (see “RS-232/422/485 Drivers (Composite)” on page 2-4). The composite drivers support RS-232 I/O for the target computer serial ports and the Quatech RS-232, RS-422, and RS-485 I/O devices. These drivers support communications in asynchronous binary mode. The Simulink Real-Time 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.

Internal Drivers

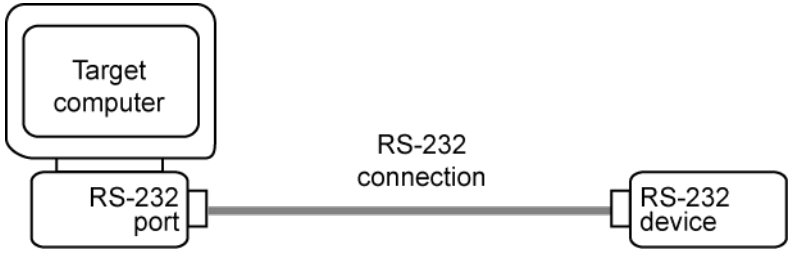
The composite drivers also include internal drivers. Use them only to modify the composite subsystems for your use. The internal blocks are composite blocks for:

- Quatech ESC-100, QSC-100, QSC-200/300, DSCP-200/300 series
- Mainboard Baseboard series
- Fastcom 422/2-PCI, Fastcom 422/2-PCI-335, Fastcom 422/4-PCI-335 series

The internal blocks and subsystems of the RS-232/422/485 boards are controlled from the mask parameters dialog box for the send/receive subsystem in which they are used.

Serial Connections for RS-232

The Simulink Real-Time software supports serial communication with the COM1 and COM2 ports on the target computer.



Your real-time applications can use these RS-232 ports as I/O devices. With the typical DTE/DCE configuration of the RS-232 device, the target computer is connected to the device with a null modem cable.

RS-232/422/485 Drivers (Composite)

This topic 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 Simulink Real-Time software supports the target computer serial ports (main board), Quatech RS-232/422/485 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 must customize the Simulink Real-Time RS-232/422/485 drivers, the composite nature of the drivers enables you to do so.

Note the following characteristics of the Commtech Fastcom: 422/2-PCI adapter boards (`fastcomproducts.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 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 (`fastcomproducts.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 and Quatech boards. The descriptions for these blocks are applicable across drivers, with additional information for specific boards.

In this section...

Adding RS-232 Blocks

You add RS-232 subsystem blocks to your Simulink model when you want to use one of the following serial interfaces for I/O:

- The serial ports on the target computer
- A Quatech QSC-100 or ESC-100 connected to the target computer

After you create a Simulink model, you can add Simulink Real-Time 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, first 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 box 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

In a `Custom` port pair, either set one or both ports of the pair to addresses other than these conventions, or assign a different IRQ value. Some boards allow you to set the IRQ numbers independently.

If you choose the port pairs `Com1/Com3` or `Com2/Com4`, you must 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 must 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 Command Window, type

   ```
   slrtlib
   ```

   The Simulink Real-Time driver block library opens.
2. Double-click the RS-232 group block.

   A window with blocks for RS-232 composite drivers opens.

   Alternatively, you can access the Simulink Real-Time block library from the Simulink Library Browser. In the Simulink Editor, and from the **View** menu, click **Show Library Browser**. In the left pane, double-click **Simulink Real-Time**, and then click **RS232**.
3. Drag 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 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 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 **Parameter group** values:

- When you select `Board Setup`, make sure that the **Configuration** value is consistent with your RS-232 serial port configuration.
- When you select `Receive Setup`, 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`. Set this parameter for all channels, including channels that you are not using; otherwise, you receive an error when generating code for the real-time application.

11 Add a Pulse Generator block and a target Scope block.

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

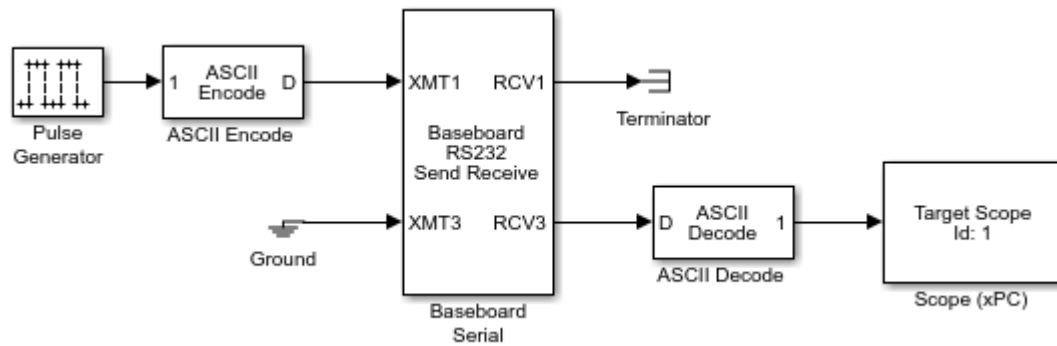
The dialog box changes to display a **Sample time** parameter. Enter a **Sample time** that is slower than the one you set for `Receive Setup`.

13 From the Simulink Library Browser, select **Sinks**. Depending on your configuration, drag one or more Terminator blocks to your model. To suppress unused port messages, connect this block to the unused RCV1 port.

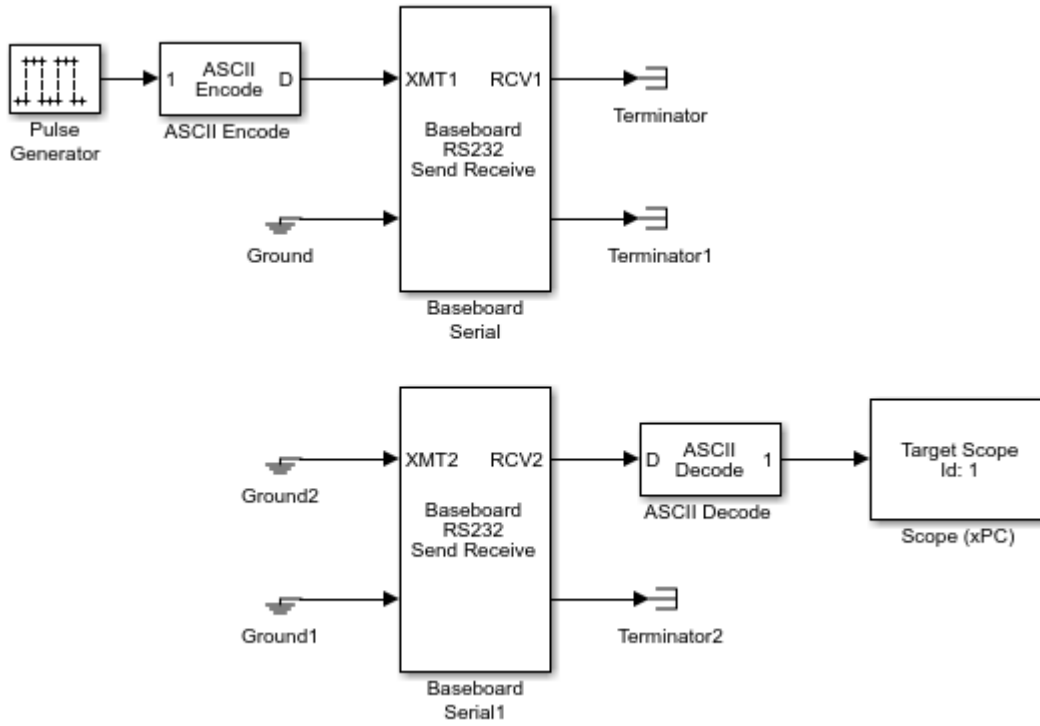
From the Simulink Library Browser, select **Sources**. Depending on your configuration, drag the Ground block to your model. To suppress unused port messages, connect this block to the unused XMT3 port.

Your model can use one block or two.

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



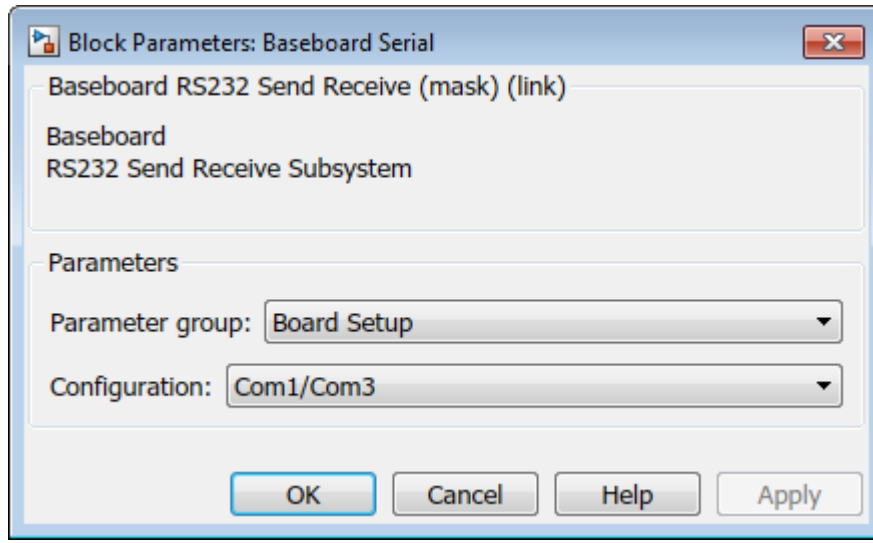
The two-block model uses two sets of `Custom` port pairs:



14 Double-click a Baseboard RS232 Send Receive block. To configure the ports on the target computer for this board, enter values.

Note: This dialog box 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 figure.



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

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

Your next task is to build and run the real-time application.

Building and Running the Real-Time Application (Composite)

The Simulink Real-Time 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 a real-time application. See “Build and Download Real-Time 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 real-time application.

Note: In this example, you cannot use a serial port to link the development and target computers. You can only use COM1 if it is not already in use for a serial link. Also, if COM1 and COM3 share an interrupt, you cannot use COM3 if COM1 is already in use for a serial link.

---

1. In the Simulink Editor, and from the **Code** menu, click **C/C++ Code** > **Build Model**.
2. In the Command Window, type

```
start(tg)
```

RS-232/422/485 Simulink Block Reference

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

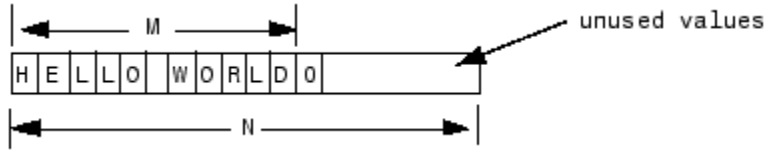
This section includes the following topics:

- "Signal Data Types" on page 2-10 — 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 character vectors. ASCII decoders parse the character vector from the Send/Receive subsystem.
- `FIFO Read/Write (Composite)` — (Generic) Describes FIFO read and write blocks.
- `RS232 State (Composite)` — (Generic) Monitors the error state information that is present in the output vector from the 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

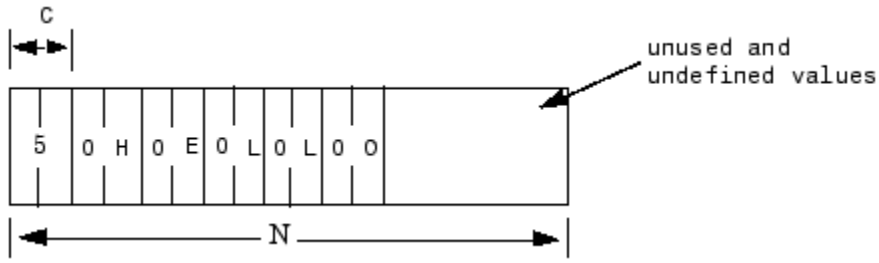
Signals between blocks in composite drivers can be one of several basic data types, 8-bit, 16-bit, and 32-bit. These types are structures.

The 8-bit data types are NULL-terminated character vectors 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 character vector has 11 characters terminated with a NULL byte (0). This data type cannot contain a NULL byte as part of the real data.

The 16-bit 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 Send block writes the low-order byte of each entry to the UART. The 16-bit and 32-bit data types allow the embedding of 8-bit data values, including 0. The 8-bit data type is most useful with the ASCII Encode and Decode blocks. The 16-bit and 32-bit data types are most useful for binary data streams.

Handling Zero Length Messages

Usually, you 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. However, you must also configure your model to deal with the possibility that a FIFO read block does not have a message on its output.

Receive FIFOs can have too few characters for a FIFO read operation. A model that receives serial I/O can have a FIFO read block that executes in this situation. This condition causes 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 Simulink Real-Time 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 returns either a new message or a zero length message.

To execute model code only if a new message arrives, check the first element of the returned vector, depending on the character vector data type:

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

If the message has nonzero length, enable a subsystem to process the new character vector; otherwise, do not process it.

Controlling When You Send a Message

You can use the structure of both serial data types ("Signal Data Types" on page 2-10) to control when a message is sent. In both cases, a `0` in the first position indicates an empty character vector.

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

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

- To send a specific character vector occasionally, use the Product block to multiply the entire character vector by either `0` or `1`. In this case, the `0` or `1` value becomes a transmit enable. To optimize this operation, use a Demux block to extract the first element. Multiply just that element by `0` or `1`, then use the Mux block to combine it again.

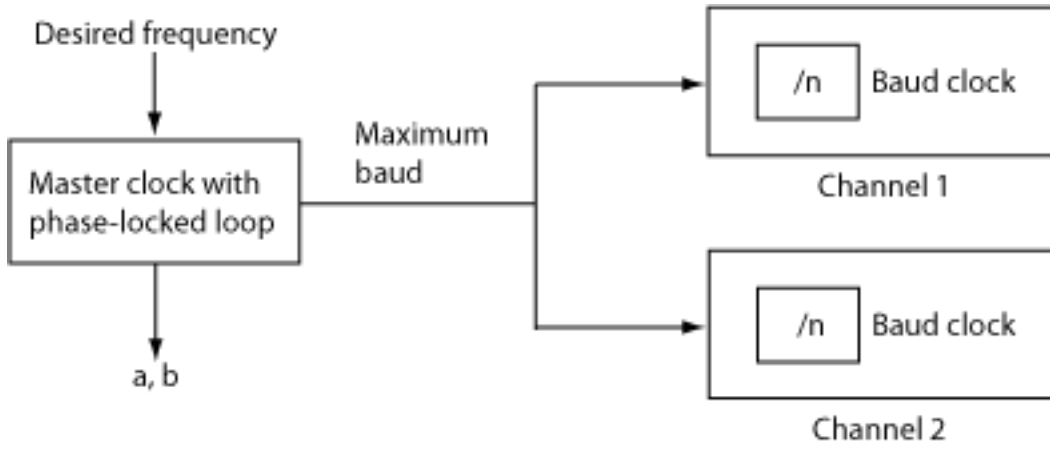
- Use a Manual Switch, Multiport Switch, or Switch block (configured for two ports to choose between different messages, with one of the choices being a vector of `0` values). The Switch block only selects between vectors of the same width. However, because the character vector length does not use the whole vector, you can pad your data to the same width with `0` values.

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

The Commtech Fastcom 422/2-PCI board can handle rates up to 1.5 megabaud. To configure a baud for the board, 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.

- In the Fastcom 422/2-PCI Send Receive block, set the **Parameter group** parameter to `Board Setup`, and then configure **Clock Bits**.
- In the Fastcom 422/2-PCI Send Receive block set the **Parameter group** parameter set to `Basic Setup`, and then configure **Baud Divisor**.

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 for both channels. The block determines the actual baud of a channel by dividing the maximum baud from the master clock by the baud divisor (`n`).



To set the block parameters for this board, choose a maximum baud, as follows. This procedure assumes that both channels require different baud. Determine a common base clock that can be divided to produce the required baud for both channels, as follows:

1. In the 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 to configure the baud for your board.

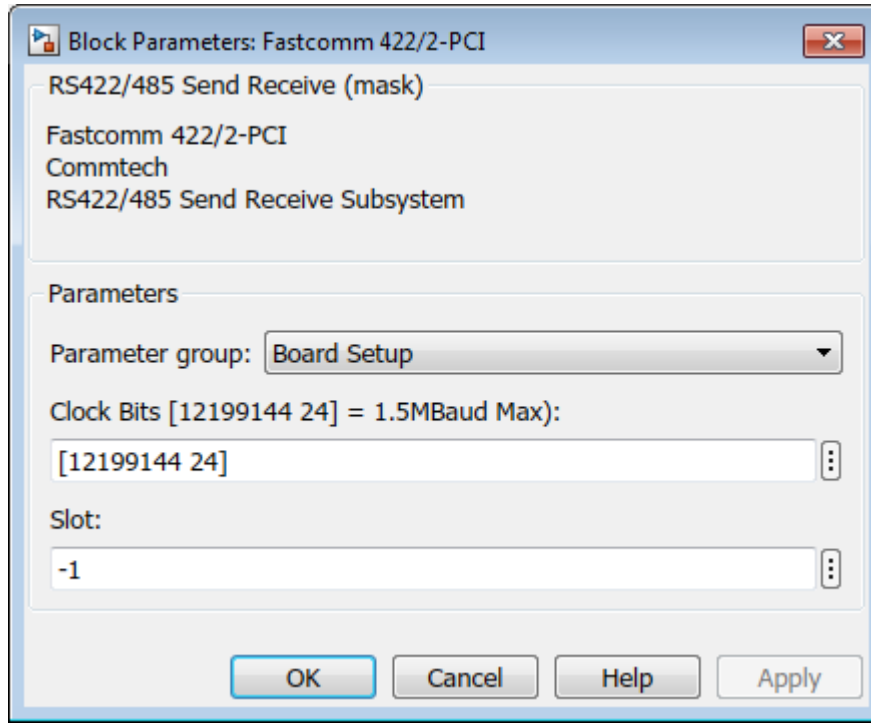
   ```
   [a b df ] = fc422mexcalcbits(1.5e6)
   ```

   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 can attain compared to your desired frequency. If the actual attainable frequency does not meet your requirements, you can try another frequency. In this example, the board can match the desired frequency of 1500000 (1.5e6).

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



After you define a maximum baud, to set a unique baud for each channel, choose a different baud divisor for each channel. For example, you can have Channel 1 have a baud of 750000 (1500000/2) and Channel 2 have a baud of 1500000 (1500000/1). To set Channel 1 to have a baud 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 slow baud rates (less than 30000), you must use the **Baud Divisor** parameter to achieve the desired baud.

See Also

`fc422mexcalcbits`

3

Serial Communications Support: Blocks

ASCII Encode/Decode (Composite)

ASCII Encode/Decode

Library

Simulink Real-Time Library for RS-232

Description

The ASCII Encode block generates a UINT8 output vector that contains a NULL-terminated character vector based on a `printf` like format string. The data comes from the input ports.

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 8 bits of each entry.

Encode Block Parameters

Format string

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`.

Number of variables

Enter the number of input ports to this block. The value on each port is inserted into the output character vector with the format specified in **Format string**.

Max output string length

Enter the maximum allowed length of the converted character vector, 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 character vector.

If the converted character vector exceeds this length, the block returns an error and does not write that character vector to the output port.

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 input ports. A cell vector with the same number of elements as specified in **Number of variables** can specify a different data type for each input port. A single element is replicated. For example,

```
nvars=3
```

`{ }` — The three inputs are doubles.

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

`{'uint16', 'double', 'uint8'}` — The first input is a `uint16`, the second input is a `double`, and the third input 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 `cdmabcdefgh90`, you must specify the value of the **Number of variables** parameter as 2.

The first variable is assigned the value `0xcd`. Next, the character vector `mabcdefgh` is considered a match to `more text` because

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

The second variable is then assigned the value `0x90`. Note that the character vector `mabcdefgh` 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
```

`{ }` — The three outputs are doubles.

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

`{'uint16', 'double', 'uint8'}` — The first output is a `uint16`, the second output is a `double`, and the third output is a `uint8`.

See Also

Topics

"RS-232 I/O Drivers" on page 2-2
"RS-232/422/485 Drivers (Composite)" on page 2-4

External Websites

www.bb-elec.com
fastcomproducts.com

Introduced in R2008a

ASCII Decode V2

ASCII Decode V2

Library

Simulink Real-Time Library for RS-232

Description

The ASCII Decode block parses an input vector produced by one of the following:

- Serial port Receive block
- Serial port FIFO Read block
- ASCII Encode block

It makes the converted values available to a Simulink program. It assumes that the input vector was prepared using an output format specifier similar to `printf` and uses an input format specifier similar to `scanf`.

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 8 bits of each entry.

This block generates inline code for the target computer. You cannot use it for Simulink simulation.

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 character vectors 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 character vector 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 character vector. If the model expects two values, and the returned count output value is less than two, the model detects an error in the data.

See Also

Topics

"RS-232 I/O Drivers" on page 2-2
"RS-232/422/485 Drivers (Composite)" on page 2-4

External Websites

www.bb-elec.com
fastcomproducts.com

Introduced in R2008a

FIFO Read/Write (Composite)

FIFO Read/Write

Library

Simulink Real-Time Library for RS-232

Description

This section describes the FIFO Read and Write blocks. There are three FIFO Read blocks.

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 ends with a new-line character. 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 without 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, the messages of a particular device use the same predefined terminator. Binary messages are normally of fixed length without a 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.

The 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 character vector like the form

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

use the simple FIFO Read block to read the message. Configure the FIFO Read block **Delimiter** parameter for a line feed (value of 10). 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 several different messages, each beginning with a different fixed character vector, use the FIFO Read HDRS block. For example, a digital multimeter connected through an RS-232 port sends 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]`.

  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. The block functions in two modes, set using the **Read to delimiter** check box.

- If you select the **Read to delimiter** check box, the block only reads elements if the specified delimiter has been written to the FIFO Write block. If the delimiter is found, the block returns elements up to and including the delimiter in the output vector. If the delimiter is not found, the block returns a zero length vector, as determined by the data type. (If you have a zero length vector, you can have your model perform a particular operation, or ignore the case.)
- If you clear the **Read to delimiter** check box, the block returns elements between **Minimum read size** and the smaller of the number of elements currently in the FIFO and **Maximum read size**.

You usually select the **Read to delimiter** check box when performing ASCII reads and clear it when performing binary reads.

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 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 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 character vector 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 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 one time. If a write operation to the FIFO causes the number of elements to exceed **Size**, an error occurs.

Input vector type

This parameter specifies the input vector type. From the list, select one of:

- `count+32 bit int`, `count+32 bit uint`, `count+16 bit int`, `count+16 bit uint` — For the 16- and 32-bit data types, include as first element the number of elements to expect in the rest of the input vector. The count controls how many bytes that the block copies into the FIFO. The block does not copy the count itself into the FIFO.
- `8 bit int null terminated`, `8 bit uint null terminated` — For the 8-bit data types, provide a null terminated character vector in the output vector. The block copies data into the FIFO up to, but not including, the null terminator.

For more information, see "RS-232/422/485 Drivers (Composite)" on page 2-4.

Data present output

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

SampleTime

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

ID

Enter a user-defined identifier for overflow messages.

Example

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

- **Transmit side of the interrupt service routine** — If the interrupt reason is not an empty FIFO, the maximum input port receives a value of `0`. If the FIFO is empty, it receives the FIFO size. The minimum input port receives the constant value of `1`.
- **Receive side of the interrupt service routine** — 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.

See Also

Topics

"RS-232 I/O Drivers" on page 2-2
"RS-232/422/485 Drivers (Composite)" on page 2-4

External Websites

www.bb-elec.com
fastcomproducts.com

Introduced in R2008a

FIFO Read HDRS (Composite)

FIFO Read HDRS block

Library

Simulink Real-Time Library for RS-232

Description

The FIFO Read HDRS block identifies and separates ASCII data streams that have embedded identifiers.

The data following a particular header can have varying lengths, but has a common termination marker such as `<CR><LF>`. Although you can attain this same functionality with the FIFO Read/Write (Composite) block, doing so requires a complicated state machine with the following behavior:

- If the same header arrives in the FIFO more than once after the block was last executed, the block returns the latest instance of the header. In this way, the block catches up with data that arrives faster than the block executes.
- If a header arrives in the FIFO that does not match an item in the headers list, the block discards the message.
- If bytes arrive in the FIFO that do not match a header, the block interprets the message as having an unspecified header. The block skips these bytes.

The `xpcdemos` folder 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 character vector for the data. Enter the characters defining the end of character vector, typically one or two characters.

Output behavior

From the list, select the behavior of the block if the FIFO has not received 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.

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 character vector in the output vector. For 16- and 32-bit data types, the first element contains the number of elements that are valid in the rest of the output vector.

SampleTime

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

See Also

See Also

`FIFO Read/Write (Composite)`

Topics

"RS-232 I/O Drivers" on page 2-2
"RS-232/422/485 Drivers (Composite)" on page 2-4

External Websites

www.bb-elec.com
fastcomproducts.com

Introduced in R2008a

FIFO Read Binary (Composite)

FIFO Read Binary

Library

Simulink Real-Time Library for RS-232

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. If the same header arrived in the FIFO more than once since the block was last executed, the block discards the older data. It then returns the latest instance of the header. In this way, the block catches up with data that arrives faster than the block executes.

The `xpcdemos` folder 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 character vector or a concatenation with `char(val)` for non-printable byte patterns.

Message Lengths

Enter the message length, in bytes. Include the header in the length.

Output behavior

From the list, select the behavior of the block if the FIFO does not have 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.

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 character vector 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.

See Also

Topics

"RS-232 I/O Drivers" on page 2-2
"RS-232/422/485 Drivers (Composite)" on page 2-4

External Websites

www.bb-elec.com
fastcomproducts.com

Introduced in R2008a

Modem Control (Composite)

Modem Control block

Library

Simulink Real-Time Library for RS-232

Description

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

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

Block Parameters

Port (Quatech, Commtech)

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

RTS

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

DTR

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

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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

Topics

"RS-232 I/O Drivers" on page 2-2
"RS-232/422/485 Drivers (Composite)" on page 2-4

External Websites

www.bb-elec.com
fastcomproducts.com

Introduced in R2008a

Modem Status (Composite)

Modem Status block

Library

Simulink Real-Time Library for RS-232

Description

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

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

Block Parameters

Port (Quatech, Commtech)

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

CTS

Select this check box to monitor the CTS line.

DSR

Select this check box to monitor the DSR line.

RI

Select this check box to monitor the RI line.

DCD

Select this check box to monitor the DCD line.

Sample time

Enter the base sample time or a multiple of the base sample time (`-1` means sample time is inherited).

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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

Topics

"RS-232 I/O Drivers" on page 2-2
"RS-232/422/485 Drivers (Composite)" on page 2-4

External Websites

www.bb-elec.com
fastcomproducts.com

Introduced in R2008a

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

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

Library

Simulink Real-Time Library for RS-232

Description

The dialog box 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). Both serial boards and 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 bits, 16 bits, or 32 bits). Characters appear 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 the FIFO read blocks.
    - **FIFO Read** — A model that contains an F block 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 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).

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 a Send/Receive block. COM1 is also of note because you can use it for linking the development and target computers. If COM1 is used for the serial link, you cannot use COM1 or COM3 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

Both F and non-F Send Receive blocks support the **Parameter Group** `Board Setup` option.

Configuration (Mainboard)

From the list, choose combinations of port pairs (`Com1/none`, `Com2/none`, `Com1/Com3`, `Com2/Com4`, `none/Com3`, `none/Com4`, or `Custom`). This parameter specifies the ports for which you are defining transmit and receive. A Custom port is one that

does not match the existing combinations of port pairs. For example, you can set the IRQ and two addresses or, if one of the ports is not used, set that to 0.

IRQ number (Quatech, Commtech)

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

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

This command displays 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.

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. Calculate this parameter using the function `fc422mexcalcbits`.

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 6 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.

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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

Basic Setup

Both F and non-F Send Receive blocks support the **Parameter Group** `Basic Setup` option.

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 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.
- **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:

```
reference = baud rate x sampling rate x divisor x prescale
```

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

```
reference = 1.10e6 x 16 x 1 x 1 = 16 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, (Commtech Fastcom 422/2-PCI-335, 422/4-PCI-335)

From the list, choose a baud rate.

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.

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 RS-232 boards work fine with a character stream that uses single (1) stop bits.

Hardware FIFO size (Quatech)

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 value of **Receive FIFO interrupt level**. If the FIFO on the board 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 an arbitrary number of bytes up to the 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 RTS/CTS handshake for flow control. This RTS/CTS handshake feature of the UART is used to prevent data loss due to FIFO overflow on the board.

Because of the large 64 byte FIFO in the I/O module, 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 FIFO. However, if you get 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 RTS/CTS handshake for flow control. This feature of the UART is used to prevent loss due to 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 FIFO gets too full. It then sends the XON character when the receive FIFO empties. Because the UART handles this, 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 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 de-asserts 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

Only non-F Send Receive blocks support the **Parameter Group** `Transmit Setup` option.

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.

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 character vector 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. Because the 8-bit data types are NULL terminated character vectors, the NULL byte would terminate the character vector.

Receive Setup

Only non-F Send Receive blocks support the **Parameter Group** `Receive Setup` option.

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 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:

```
reference = baud rate x sampling rate x divisor x prescale
```

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

```
reference = 1.10e6 x 16 x 1 x 1 = 16 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.

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 nothing.

If the buffer has 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 character vector 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.

FIFO Setup

Only F Send Receive blocks support the **Parameter Group** `FIFO Setup` option.

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 character vector 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.

See Also

See Also

`fc422mexcalcbits` | `FIFO Read Binary (Composite)` | `FIFO Read HDRS (Composite)`

Topics

"RS-232 I/O Drivers" on page 2-2
"RS-232/422/485 Drivers (Composite)" on page 2-4

External Websites

www.bb-elec.com
fastcomproducts.com

Introduced in R2008a

RS232 State (Composite)

RS232 State block

Library

Simulink Real-Time Library for RS-232

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 false. If the input vector is `Int16` or `UInt16`, the upper byte contains 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-bit or 32-bit data streams. The RS232 State block looks at this status. 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 FIFO in the UART was filled 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.

For some serial I/O port modules, disconnecting the serial cable does not cause a break.

See Also

Topics

"RS-232 I/O Drivers" on page 2-2
"RS-232/422/485 Drivers (Composite)" on page 2-4

External Websites

www.bb-elec.com
fastcomproducts.com

Introduced in R2008a

A

Serial Communications Support: Internal Blocks

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

RS-232/422/485 Enable TX Interrupt block

Library

Simulink Real-Time Library for RS-232

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 the interrupt 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.

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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

Topics

"RS-232 I/O Drivers" on page 2-2
"RS-232/422/485 Drivers (Composite)" on page 2-4

External Websites

www.bb-elec.com
fastcomproducts.com

Introduced before R2006a

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

RS-232/422/485 Filter Interrupt Reason block

Library

Simulink Real-Time Library for RS-232

Description

The Filter Interrupt Reason block filters the output of the Read Int(errupt) Status block.

If the condition that the interrupt query block reads from the IIR register matches the one chosen here, the output is true.

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

Block Parameters

Port

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

Filter value

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

Note that `Modem status change` currently does nothing because the interrupt is not enabled.

See Also

Topics

"RS-232 I/O Drivers" on page 2-2

"RS-232/422/485 Drivers (Composite)" on page 2-4

External Websites

www.bb-elec.com
fastcomproducts.com

Introduced before R2006a

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

RS-232/422/485 Read Hardware FIFO block

Library

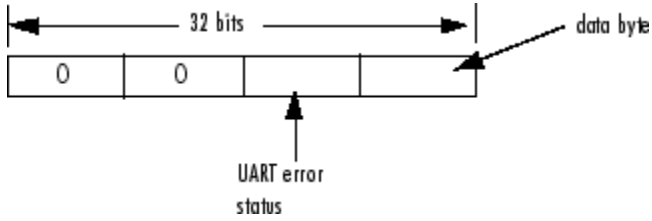
Simulink Real-Time Library for RS-232

Description

The Read Hardware FIFO block reads characters from the I/O module 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 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 zero-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 FIFO from this port.

Flush HW FIFO on startup

Select this check box to flush the 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 FIFO.

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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

Topics

"RS-232 I/O Drivers" on page 2-2
"RS-232/422/485 Drivers (Composite)" on page 2-4

External Websites

www.bb-elec.com
fastcomproducts.com

Introduced before R2006a

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

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

Library

Simulink Real-Time Library for RS-232

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 4 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).

The 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. Some bytes in this register give the active FIFO depth. Other bytes give the maximum number of characters that the transmitter empty interrupt handlers can write to the transmit 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.

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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

Topics

"RS-232 I/O Drivers" on page 2-2
"RS-232/422/485 Drivers (Composite)" on page 2-4

External Websites

www.bb-elec.com
fastcomproducts.com

Introduced before R2006a

RS-232/422/485 Setup (Composite)

RS-232/422/485 Setup block

Library

Simulink Real-Time Library for RS-232

Description

A setup block is a subsystem block that sets up the interface characteristics for the board.

For Quatech boards, this setup block is for one channel or port.

Block Parameters

Port (Quatech, Commtech)

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

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)

From the list, choose a baud rate.

Number of data bits

From the list, choose either 5, 6, 7 or 8 to define the number of data bits for the port.

Number of stop bits

From the list, choose either 1 or 2 to define the number of stop bits for the port.

Parity

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

Fifo mode (Mainboard, Quatech)

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 FIFO depth of 1 character (1 deep), 16 characters (16 deep), or 64 characters (64 deep).

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 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 handshaking using the RTS and CTS modem control lines. If this is not checked, handshaking is not done.

Base Address (Mainboard, ISA)

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

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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

Topics

"RS-232 I/O Drivers" on page 2-2
"RS-232/422/485 Drivers (Composite)" on page 2-4

External Websites

www.bb-elec.com
fastcomproducts.com

Introduced before R2006a

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

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

Library

Simulink Real-Time Library for RS-232

Description

(Quatech and Commtech only) The Board Setup and Interrupt Check block checks for instances where the 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 computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

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.

See Also

Topics

"RS-232 I/O Drivers" on page 2-2
"RS-232/422/485 Drivers (Composite)" on page 2-4

External Websites

www.bb-elec.com
fastcomproducts.com

Introduced before R2006a

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

RS-232/422/485 Write Hardware FIFO block

Library

Simulink Real-Time Library for RS-232

Description

The Write Hardware FIFO block writes the data from the input port (labeled E) to the FIFO in the I/O module 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
}
```

In words: if the enable port (input port E) becomes `true` and the input data has length `0`, then the block turns off the transmitter buffer empty interrupt. Otherwise, the block adds input data to the FIFO.

Block Parameters

Base address (Mainboard, ISA boards)

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

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 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.

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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

Topics

"RS-232 I/O Drivers" on page 2-2
"RS-232/422/485 Drivers (Composite)" on page 2-4

External Websites

www.bb-elec.com
fastcomproducts.com

Introduced before R2006a

CAN, Encoders, Ethernet, EtherCAT

CAN I/O Support

CAN Basics

In this section...

Simulink Real-Time CAN Library

The Simulink Real-Time block library offers support to connect a target computer to a CAN network using the CAN driver blocks provided by the Simulink Real-Time 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 Simulink Real-Time applications to connect to an arbitrary 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 buffer mode of the CAN-AC2 firmware to achieve real-time performance.

The Simulink Real-Time 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. The following examples use CAN_MESSAGE data types.

CAN-AC2-PCI Example	Description
`“CAN I/O - Simple Use Case”` — CAN_MESSAGE data type	Shows simple CAN I/O communication.
`“CAN I/O - Message-Based Interrupts”` — CAN_MESSAGE data type	Shows asynchronous message-based event support using interrupt driven CAN I/O communication.

The size of the driver code of the CAN boards supported by the Simulink Real-Time block library is significant. Because not all Simulink Real-Time applications use CAN, the CAN library code is not linked by default when building a real-time application. This non-linking makes real-time applications smaller if CAN communication functionality is not required. If the model to be built contains CAN driver blocks, Simulink Real-Time links in the CAN library code required to run the model.

For each CAN board three driver blocks are provided:

- Setup block — Defines the type of physical connection (baud, and so forth). Exactly one instance of the Setup block must be defined in a model for each physically installed CAN board.
- Send block — Transmits (sends) the data entering the block input ports to the connected CAN network. One or more instances of the Send block can be used in a model.
- Receive block — 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.

Drivers for the supported CAN boards program the boards for the so-called dynamic object buffer mode. Dynamic object buffer mode is one of three modes in which the CAN board firmware from Softing can operate. The other two are FIFO mode and static object mode. For a more detailed discussion of the three modes, see the board user manual.

Dynamic object buffer mode is best suited for real-time environments where each component of the application must have deterministic time behavior. Because Simulink Real-Time exhibits this behavior, it provides dynamic object buffer mode as basic support. 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 Simulink Real-Time CAN blocks support the following message data types.

- 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 and consider updating existing CAN application models to use them.

  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 block. 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 Simulink Real-Time 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) and 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 for using CAN_MESSAGE data types.
- Remove object mode CAN Message and CANDBC Translator blocks. Updated models do not require the conversions that these blocks perform.

Sending and Receiving Remote Frames

Ordinarily, a subsystem sends its value over the CAN bus according to its own schedule. Another subsystem stores that value for use whenever downstream processing requires it. Sometimes, 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-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 `company.softing.com/en`.

- 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 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 PCI BIOS of the target computer assigns the address space automatically. The address space lies usually in the range of 2–4 GB.
- 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 extension is not available. The board is both I/O mapped and memory mapped. The I/O-mapped area uses a 4 byte I/O address range. The memory-mapped area uses a 4-KB address range between 640 KB and 1 MB.

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 Simulink Real-Time I/O block library in the group CAN/Softing.

The CAN-AC2-PCI SJA 1000 block group contains the three available CAN blocks:

- `Softing CAN-AC2-PCI with SJA1000 Setup`

- `Softing CAN-AC2-PCI with SJA1000 Send`
- `Softing CAN-AC2-PCI with SJA1000 Receive`

It also includes a FIFO Mode group, which is discussed in “FIFO Mode” on page 6-2.

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 Simulink Real-Time I/O block library in the group CAN/Softing.

The CAN-AC2-104 SJA 1000 block group contains the CAN blocks:

- `Softing CAN-AC2-104 with SJA1000 Setup`
- `Softing CAN-AC2-104 with SJA1000 Send`
- `Softing CAN-AC2-104 with SJA1000 Receive`

It also includes a FIFO Mode group, which is discussed in “FIFO Mode” on page 6-2.

Model Execution Driven by CAN Messages

In certain applications, you can pace execution of the real-time application by incoming CAN messages. The standard behavior of the Simulink Real-Time kernel is to drive the real-time application in time monotonic fashion using a time interrupt. However, you can replace the driving interrupt by other interrupts. The supported CAN boards can fire an interrupt upon reception of a specific CAN message. Therefore, you can replace the timer interrupt line in the kernel by the interrupt line assigned to a CAN board. You can then drive the real-time application by sending CAN messages, interpreting them in the CAN Receive blocks, and firing an interrupt in response.

To set up this capability, do the following:

1. Replace the timer interrupt line in the kernel setup with the interrupt line of the board.
2. Set up the CAN Setup and CAN Receive blocks.

Both steps are slightly different for each of the supported CAN boards.

In this section...
"CAN-AC2-PCI" on page 4-7
"CAN-AC2-104 (PC/104)" on page 4-8

CAN-AC2-PCI

The CAN-AC2 is a PCI board, and the PCI BIOS automatically assigns the interrupt line during the initialization of the target system. At the model level, open the **Simulation > Model Configuration Parameters** dialog box. At the **Simulink Real-Time Options** node, set **Real-time interrupt source** `Auto (PCI only)`. This option enables the Simulink Real-Time software to determine the IRQ that the BIOS assigned to the board and use it. Alternatively, use the Simulink Real-Time function `SimulinkRealTime.target.getPCIInfo` at the 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 Editor, 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** box to `slrt.tlc`.

4 Select the **Simulink Real-Time Options** node.

5 In the **Real-time interrupt source** box, select the interrupt line number that you retrieved with the `SimulinkRealTime.target.getPCIInfo` command.

   Alternatively, select `Auto (PCI only)` to enable the Simulink Real-Time software to 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 that the CAN messages defined through their identifiers in this Receive block instance are messages that fire an interrupt. In other words, you cannot define a single CAN message within the set of defined identifiers as the only one to fire an interrupt. Usually, 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 the remaining 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 interrupt line is set with a software setting within the CAN Setup block. Make sure that the interrupt line is not used by any other device in the Simulink Real-Time system (for example by the Ethernet card).

1 In the Simulink Editor, 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** box to `slrt.tlc`.

4 Select the **Simulink Real-Time Options** node.

5 In the **Real-time interrupt source** box, select the free interrupt line number that you chose.

6 In the **I/O board generating the interrupt** box, 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 that the CAN messages defined through their identifiers in this Receive block instance are messages that fire an interrupt. In other words, you cannot define a single CAN message within the set of defined identifiers as the only one to fire an interrupt. Usually, 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 the remaining normal CAN messages to be received (**Generate interrupts** cleared).

After you complete these steps, you are ready to build the model. After the download has succeeded and the real-time application execution has been started, the selected CAN messages drive the execution. The execution-time information displayed on the target display is now directly dependent on the reception of the corresponding message. If no message is received, the time does not advance.

Generate the corresponding CAN message on the other CAN node only if the Simulink Real-Time application is running. Otherwise, the target screen can display unexpected interrupt messages.

Initialization and Termination CAN Messages

The CAN Setup driver blocks for the supported CAN boards allow you to define CAN messages to be sent during initialization and termination of the real-time application. The driver blocks send these messages 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, such as CANopen or DeviceNet nodes. Even if those CAN layers are not directly supported, the model can usually communicate with those nodes with standard CAN messages, provided the nodes are initialized. The initialization and termination fields of the Setup blocks are intended for this purpose.

You define the initialization and termination CAN messages by using MATLAB struct arrays with CAN specific field names. This approach was also used for the RS-232 and general counter driver blocks found in the Simulink Real-Time 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'` (character vectors).
- **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 from 0 through 255 (decimal).
- **Pause** — Defines the amount of time, in seconds, that the Setup block waits after this message has been sent before sending the next message. Valid values are 0–0.05 seconds. Some CAN nodes need time to settle before they can accept the next

message. For example, the node must settle when the previous message puts it in a new operational mode. Use this field to specify those idle times.

CAN Data Frames

CAN data frames have a maximum size of 8 bytes (64 bits). The CAN driver blocks in the Simulink Real-Time 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, the Simulink Real-Time I/O library contains two sets of utility blocks, found in subgroup CAN/Utilities. Each set supports bit-packing (construction) and bit-unpacking (extraction) of data frames in a flexible way.

- To construct and extract data frames that use the double data type, use the `CAN Bit-Packing` and `CAN Bit-Unpacking` blocks. These blocks support legacy models.
- To construct and extract data frames that use the CAN_MESSAGE data type, use the `CAN Pack` and `CAN Unpack` blocks.

The main purpose of these blocks is to be used with CAN Send and Receive driver blocks. You can use the CAN Bit Packing and CAN Bit Unpacking blocks for other types of double type data manipulation. Their functionality is independent of the CAN driver blocks or CAN library.

You can use the CAN Pack and CAN Unpack blocks to prepare a remote frame. For more information, see “Sending and Receiving Remote Frames” on page 4-4.

Timeouts When Receiving CAN Messages

The model in this topic supports the CAN_MESSAGE data type.

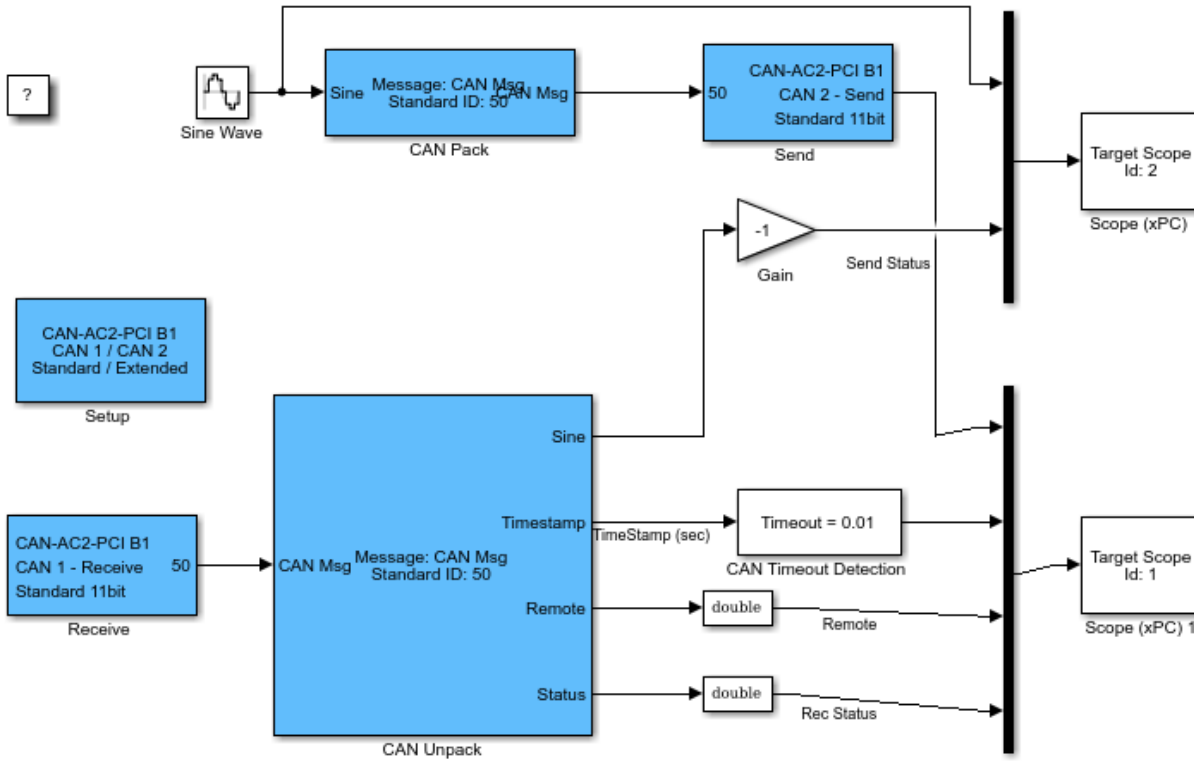
The Receive blocks for the supported CAN boards allow you to output the timestamp at which the latest corresponding CAN message was received. This information can be used to detect that another CAN node is not sending CAN messages and special action must be taken. Assume that a CAN message is expected from another CAN node every 2 milliseconds. If a new message is not received within 10 milliseconds, the other CAN node is considered faulty, and the Simulink real-time application must proceed accordingly.

The CAN blockset in the Simulink Real-Time I/O block library provides a utility block called CAN Timeout Detection. This subsystem uses the timestamp information to calculate the timeout condition.

To use this block:

1. Select the **Output timestamp** check box of a CAN Unpack block.
2. Connect the `Timestamp` output port to the input port of a CAN Timeout Detection block.
3. Connect the output port of the CAN Timeout Detection block to a sink, such as a real-time Scope block.
4. Set the **Timeout** value to the required value, such as `0.01` s.

Applying these steps to the "`CAN I/O - Simple Use Case`" results in the model shown in the figure.



See Also

matlab: open_system(docpath(fullfile(docroot, 'toolbox', 'xpc', 'examples', 'ex_slrt_canpcis_timeout')))

CAN I/O Blocks

Softing CAN-AC2-PCI with SJA1000 Setup

Softing CAN-AC2-PCI with SJA1000 Setup block

Library

Simulink Real-Time 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 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 the module is not 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 ]
```

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 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 the module is not 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.

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)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

The board allows you to terminate each of the two CAN ports separately with 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.

See Also

External Websites

www.can-cia.org

company.softing.com/en

Softing CAN-AC2-PCI with SJA1000 Send

Softing CAN-AC2-PCI with SJA1000 Send block

Library

Simulink Real-Time 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.

- 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 buffer 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, in bytes, for each CAN message represented by an identifier in the **Identifiers** list. 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 the identifiers must be the same, you can provide the size as a scalar, and scalar expansion applies. If the sizes are different for at least two identifiers, you must provide one element for each identifier. Therefore the length 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 CAN message represented by an identifier in the **Identifiers** list. 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 **Identifiers** or **CAN identifier range** value. This is not a Softing standard.
-3	CAN_MESSAGE data type only. Input port message from CAN Pack block has incorrect **Data frame sizes** value. This is not a Softing standard.
-4	Timed out firmware communication.
-99	Board not initialized.

Sample time

Defines the sample time at which the Send block is executed during a real-time 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.

See Also

External Websites

www.can-cia.org
company.softing.com/en

Softing CAN-AC2-PCI with SJA1000 Receive

Softing CAN-AC2-PCI with SJA1000 Receive block

Library

Simulink Real-Time 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.

- 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 buffer 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.

Message lengths

Defines the message length, in bytes, for each CAN message represented by an identifier in the **Identifiers** list. It must be a row vector in which the elements define a set of message lengths. Each element must be in the range between 0 and 8. If the message length for the identifiers must be the same, you can provide the length as a scalar, and scalar expansion applies. If the lengths are different for at least two identifiers, you must provide one length element for each identifier. Therefore the dimensions of the two vectors must be the same.

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.

`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	Timed out 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 real-time application execution.

Sample time

Defines the sample time at which the Send block is executed during a real-time 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.

See Also

External Websites

www.can-cia.org
company.softing.com/en

Softing CAN-AC2-104 with SJA1000 Setup

Softing CAN-AC2-104 with SJA1000 Setup block

Library

Simulink Real-Time Library for CAN

Description

The Setup block defines general settings of the stacked CAN boards. The CAN driver blocks for this board support up to three boards for each target system, making up to six CAN ports available. For each board in the target system, you must use exactly one Setup driver block.

Block Parameters

Board

Defines the board being accessed by this driver block instance. If multiple boards are present in the target 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. If just one board is present in the target system, select board number 1. The physical board referenced by the board number depends on the **I/O base address** parameter.

CAN 1 - baud rate

Defines the most common baud rates for CAN port 1. If your model requires a special baud rate, select the value `User defined`. In this case, use the **CAN 1 - user defined baud rate** parameter to provide the four values for the timing information. The vector elements have the following meanings:

```
[ Prescaler, Synchronization-Jump-Width, Time-Segment-1,
Time-Segment-2 ]
```

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

CAN 1 - user defined baud rate

See **CAN 1- baud rate**.

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, 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

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`

  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 4 bytes of I/O address space. It 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 (the board does not have a corresponding DIP switch). 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.

The Softing CAN-AC2-104 driver can address a memory map in the memory range `C8000 - F7FFF`. However, the Simulink Real-Time software only reserves memory-mapped space in the address range `C0000 - DBFFF`. Therefore, you must address the Softing CAN-AC2-104 in the range

```
C8000 - DBFFF
```

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.

Interrupt line

Selects an interrupt line from the list.

See Also

External Websites

www.can-cia.org

company.softing.com/en

Softing CAN-AC2-104 with SJA1000 Send

Softing CAN-AC2-104 with SJA1000 Send block

Library

Simulink Real-Time 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.

- 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 buffer 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, in bytes, for each CAN message represented by an identifier in the **Identifiers** list. 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 the identifiers must be the same, you can provide the size as a scalar, and scalar expansion applies. If the sizes are different for at least two identifiers, you must provide one element for each identifier. Therefore the length 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 CAN message represented by an identifier in the **Identifiers** list. 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 **Data frame sizes** value.
-1	Request overrun.
-2	CAN_MESSAGE data type only. Input port message from CAN Pack block has incorrect **Identifiers** or **CAN identifier range** value. This is not a Softing standard.
-3	CAN_MESSAGE data type only. Input port message from CAN Pack block has incorrect **Data frame sizes** value. This is not a Softing standard.
-4	Timed out firmware communication.
-99	Board not initialized.

Sample time

Defines the sample time at which the Send block is executed during a real-time 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.

See Also

External Websites

www.can-cia.org
company.softing.com/en

Softing CAN-AC2-104 with SJA1000 Receive

Softing CAN-AC2-104 with SJA1000 Receive block

Library

Simulink Real-Time 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.

- 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.

Message lengths

Defines the message length, in bytes, for each CAN message represented by an identifier in the **Identifiers** list. It must be a row vector in which the elements define a set of message lengths. Each element must be in the range between 0 and 8. If the message length for identifiers must be the same, you can provide the length as a scalar, and scalar expansion applies. If the lengths are different for at least two identifiers, you must provide one length element for each identifier. Therefore the dimensions of the two vectors must be the same.

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.

`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	Timed out 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 real-time application execution.

Sample time

Defines the sample time at which the Receive block is executed during a real-time 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.

See Also

External Websites

www.can-cia.org
company.softing.com/en

CAN Bit-Packing

CAN Bit-Packing block

Library

Simulink Real-Time 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). This port 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 dialog box.

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 you do not need an explicit sample time in the block dialog box.

Example

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.

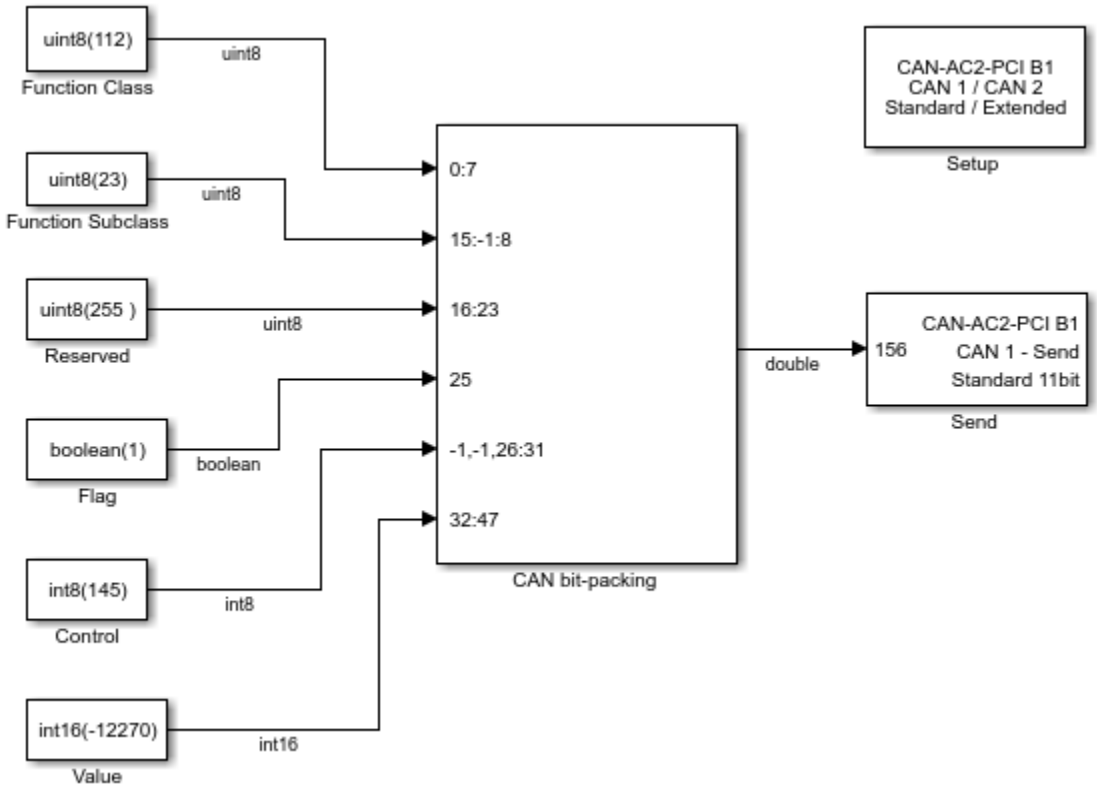
Assume that a node on the CAN network must receive a CAN message with identifier 156 having the following data frame content. The data frame must be 6 bytes long.

`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 2–7 must be bits 2–7 of an incoming int8 value (control).
`Byte 4 and 5`	Value of type `int16`.

The bit pattern cell array, which bit-packs the data frame according to the above specification, can look as follows:

```
{ [0:7] , [15:-1:8] , [16:23] , [25] , [-1,-1,26:31] , [32:47] }
```

And the Simulink model simulating the 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 0–7) 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 pattern array element does not reference a bit position in the outgoing data frame, the bit is 0 by default. However. you cannot set the bits to 1 as the default. Therefore, you must bring in a uint8

constant with value `255` externally. The constant 255 must get to bit positions 16–23 (byte 2) of the outgoing data frame.

Bit 0 of data frame byte 3 (bit 24) must be 0. Because 0 is the default value for a nonreferenced bit, you do not need to take an explicit action 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`. However, only bits 2–7 must be mapped into the outgoing data frame. In other words, bits 0 and 1 must be thrown away. Indexing of incoming values starts with the first bit (bit 0). Therefore, you must use a special indexing value (-1) to skip bits 0 and 1 of the incoming int8 value. Bits 2–7 are directly mapped to bits 2–7 of byte 3 (bits 26–31) of the outgoing data frame. The resulting bit pattern is `[-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 32–47) of the outgoing data frame. Therefore the first bit (element 1 of double array [32:47]) must get bit 32 of the data frame, the second bit 33, and so on. It is easiest to define this mapping by the MATLAB colon operator (:).

The output of the CAN Bit-Packing block consists of a double value representing the packed data types within the first 6 bytes. The last 2 bytes are zero. Therefore, even when fewer than 8 bytes are significant, a double value (8 bytes) represents the CAN data frame. The value of the constructed floating-point double does not have a 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 dialog box, the data frame size is 6 bytes. Therefore, only the first 6 bytes of the incoming double value are transmitted as part of the CAN message.

See Also

External Websites

www.can-cia.org
company.softing.com/en

Introduced before R2006a

CAN Bit-Unpacking

CAN Bit-Unpacking block

Library

Simulink Real-Time 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). This port 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 dialog box.

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
```

The sample time of the block is inherited from the driving block. Therefore you do not need to provide an explicit sample time in the block dialog box.

If you unpack the data frame into a signed type (int8, int 16, or int 32), 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

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.

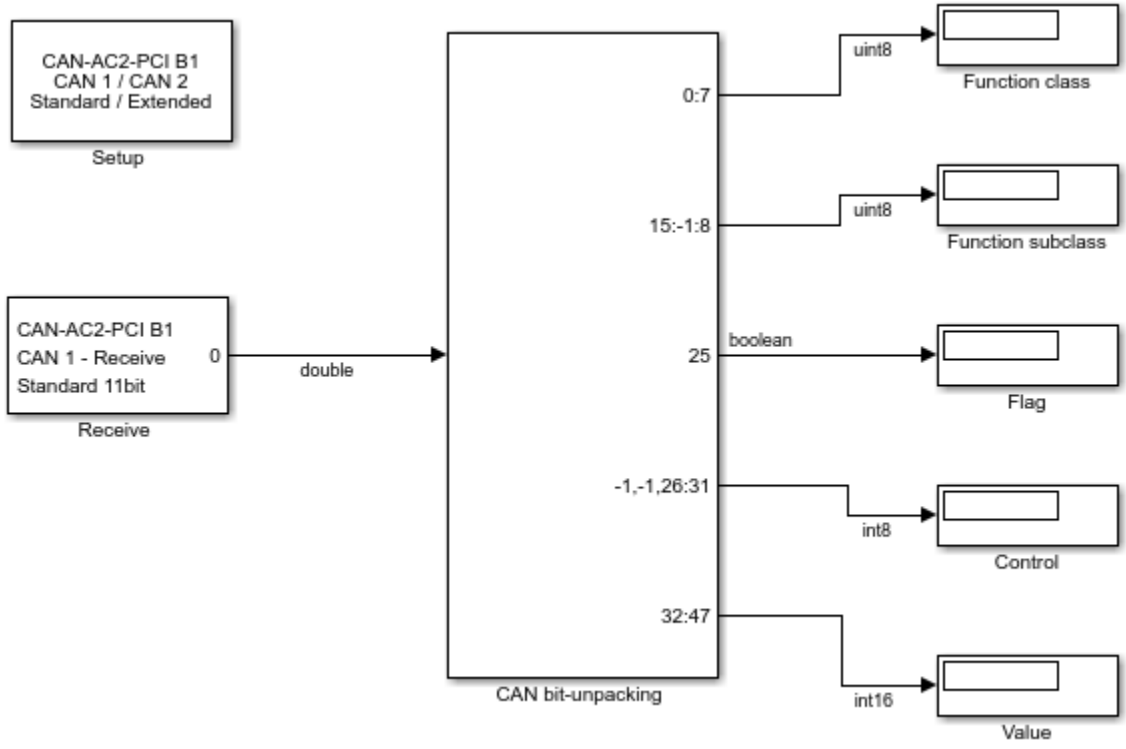
The example in the CAN Bit-Packing shows the transmission of a data frame to an external CAN node. In this case, an external CAN node sends the data frame and the real-time application running on a target computer receives it. Therefore, you use the CAN Bit-Unpacking block to extract the various data 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 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] }
```

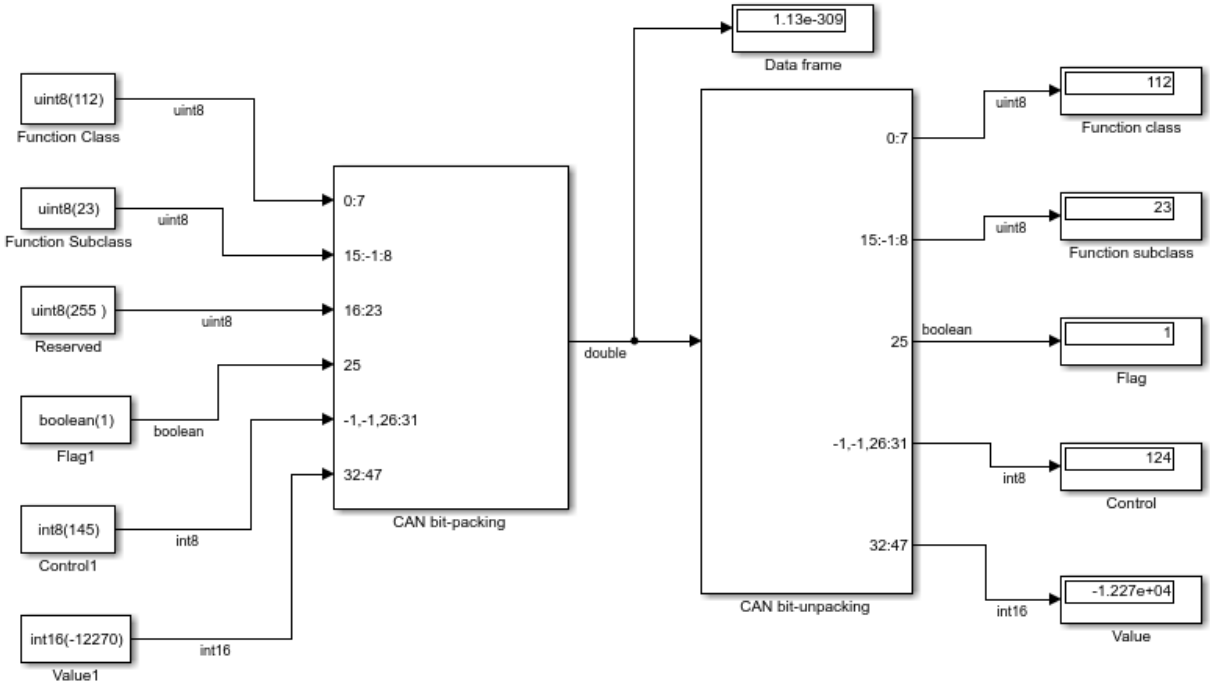
and the data type edit field as

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

These requirements result in the following Simulink model.



Often it makes sense to test the bit-packing and unpacking operations in a Simulink model (simulation) before building the real-time 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.



See Also

External Websites

www.can-cia.org
company.softing.com/en

Introduced before R2006a

CAN Timeout Detection

CAN Timeout Detection block

Library

Simulink Real-Time 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 an example of this usage, add this block to `xpccanpcis`, a loopback example for the CAN-AC2-PCI board using CAN_MESSAGE data types for message storage.

Block Parameters

Timeout

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

- `0`, if a timeout has not been detected
- `1`, if a timeout has been detected

See Also

External Websites

www.can-cia.org
company.softing.com/en

Introduced before R2006a

CAN Pack

Pack individual signals into CAN message

CAN Pack
(With raw data input)

CAN Pack
(With manually specified data input)

CAN Pack
(With CANdb specified data input)

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™ Rapid 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 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.
- **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 in 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.

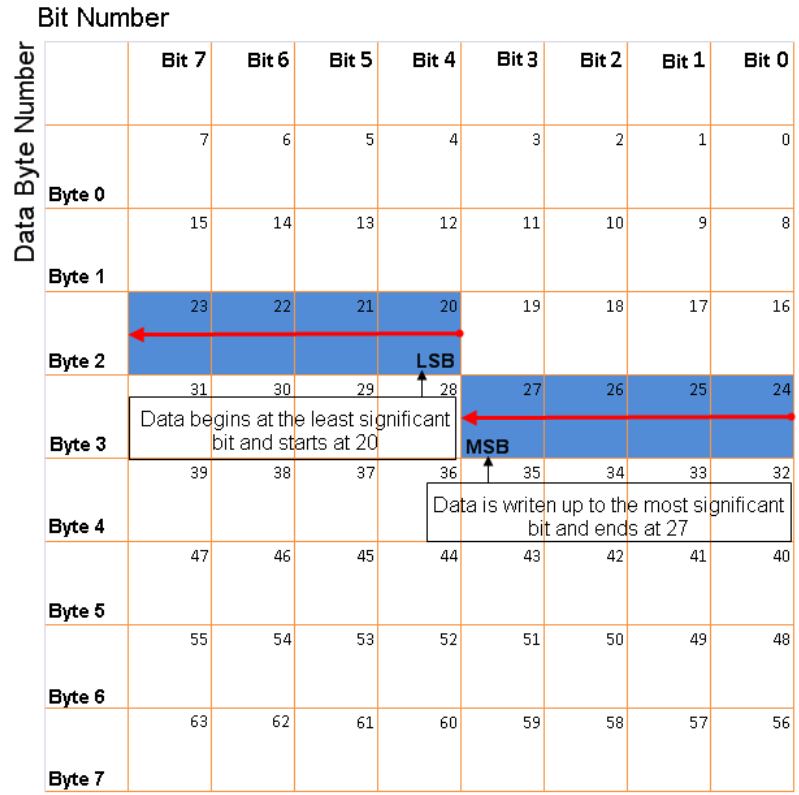
Length (bits)

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

Byte order

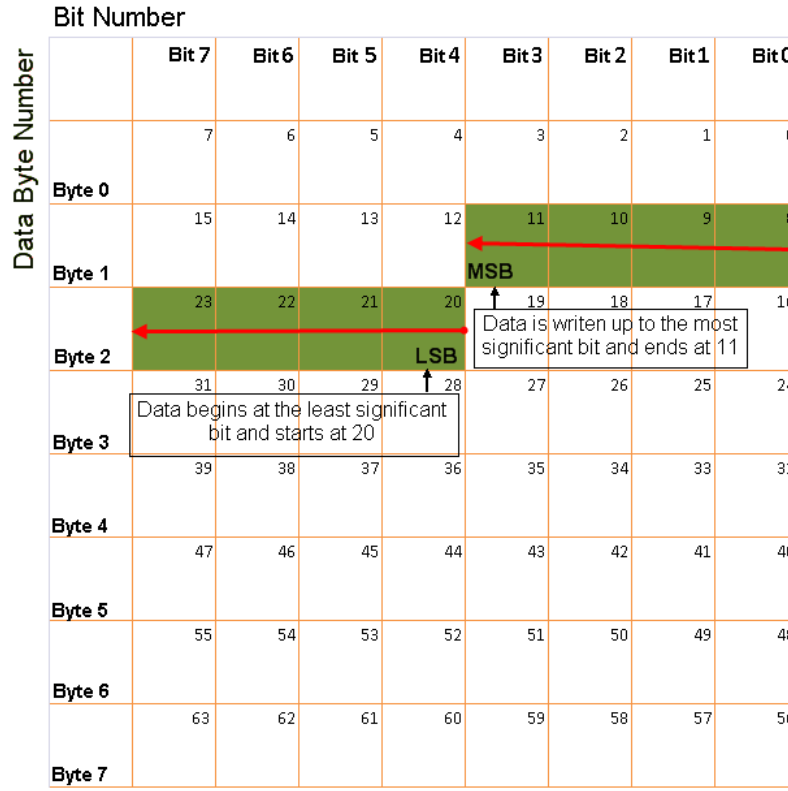
Select either of the following options:

- `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.



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.



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 5-41 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 5-41 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 5-41 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 5-41 to understand how physical values are converted to raw values packed into a message.

Conversion Formula

The conversion formula is

```
raw_value = (physical_value - Offset) / Factor
```

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.

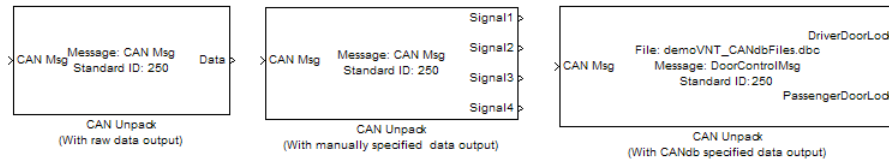
See Also

`CAN Unpack`

Introduced in R2009a

CAN Unpack

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 Rapid 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 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 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.

  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.

  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 a 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.

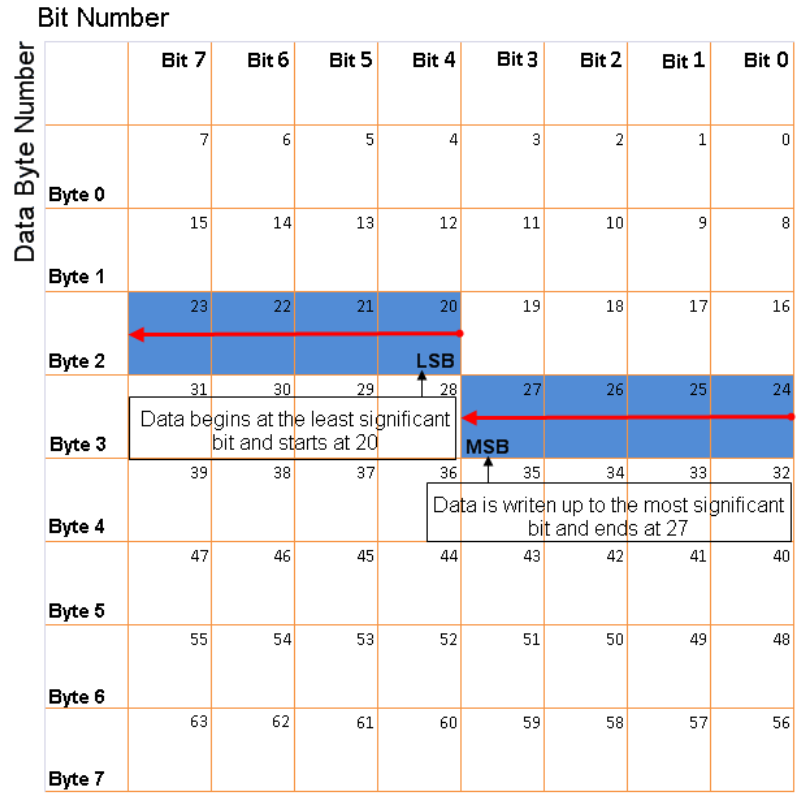
Length (bits)

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

Byte order

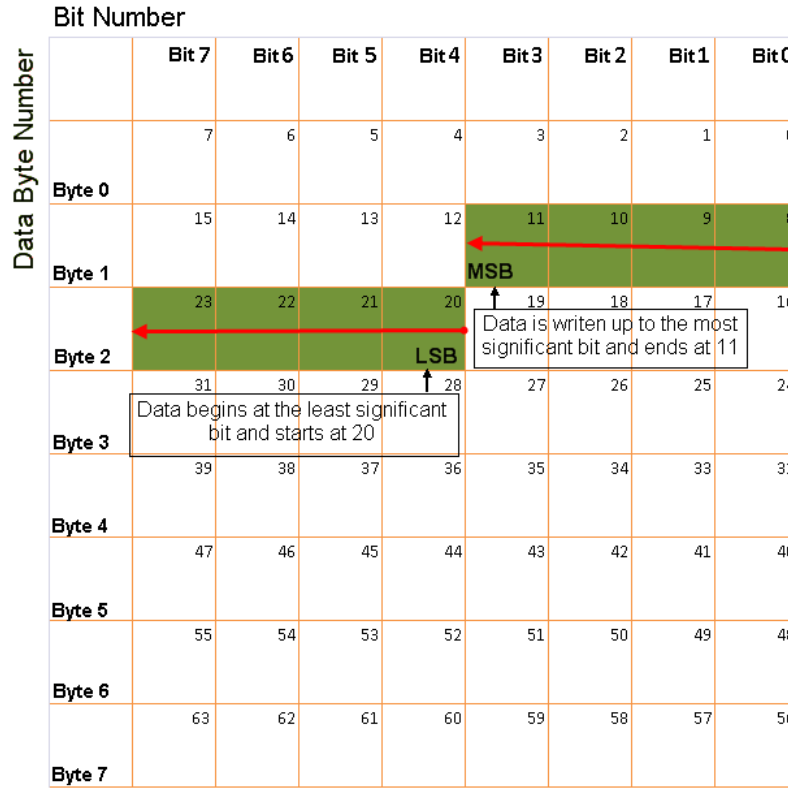
Select either of the following options:

- 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.



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 5-50 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 5-50 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 5-50 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 5-50 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

```
physical_value = raw_value *   Factor + 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.

See Also

`CAN Pack`

Introduced in R2009a

6

CAN I/O FIFO Support

CAN FIFO Basics

In this section...

FIFO Mode

This topic describes the alternative First In First Out (FIFO) mode CAN drivers provided with the Simulink Real-Time blocks. The standard CAN drivers for the CAN boards from Softing GmbH (`company.softing.com/en`) 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 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 at which the driver accesses the board to provide the information for the CAN message being sent.
  - The time at which the board returns the acknowledgment that the firmware received the information.

  FIFO mode blocks send messages in the order that they are input to the block. They do not order them by the priority of the input.

  Note: Each CAN channel has two FIFOs: DPRAM and SRAM. The Simulink Real-Time 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. Although the SRAM can buffer up to 127 messages, the DPRAM can only buffer up to 31 messages. Therefore, 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 at which the driver accesses the board to ask for current data (object data) of a certain CAN identifier.

- The time at which the board returns the actual data and other information about the CAN message.

Disadvantages of Dynamic Object Buffer mode — The reaction time of the board firmware defines these latency times. For the Softing boards, the latency time is the same for sending and receiving messages, with a fixed value of about 40 µs. If your real-time application has to manage many messages, the latency time can increase until you cannot run the application at the required sample time.

For example, assume that a specific real-time application gets data from 12 CAN identifiers and transmits data by using 8 CAN messages. The total number of CAN board read and write accesses adds up to 20. This network traffic results in a total CAN I/O latency time of:

```
20*40 µs = 800 µs
```

With such an application, base sample times below 800 µs are impossible even if the corresponding Simulink model would only need 20 µs of computation time.

Advantages of Dynamic Object Buffer mode — Even if the CAN I/O latency time is high, the latency time stays constant almost independent of the traffic volume on the CAN network. Therefore, dynamic object buffer mode is best suited for real-time applications that deal with only a small subset of the CAN messages going over the network.

Data Types

The Simulink Real-Time 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 system models. Consider updating existing CAN models to use CAN_MESSAGE data types.

  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 system 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 Simulink Real-Time 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) and 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 do not require the conversions that these blocks perform.

FIFO Mode Drivers for CAN Boards from Softing

The CAN boards from Softing support another mode, called first-in-first-out (FIFO) mode. In FIFO mode, the dynamic object buffer mode abstraction layer in the firmware is missing. Instead, the firmware provides an interface between the receive and transmit FIFOs and the drivers in the real-time application code. Because of this slimmer interface, the I/O latency times are considerably smaller. Writing to the transmit FIFO takes 4 μs per CAN message and reading one event (CAN message) from the receive

FIFO takes 17 µs. Both of these latency times are smaller than the 40 µs for dynamic object buffer mode.

While writing to the transmit FIFO is efficient, reading from the receive FIFO is not. The receive FIFO contains all of the CAN messages (identifiers) going over the CAN network. Your program must read much data (CAN messages) out of the FIFO even if their data is not used in the real-time application. Because of the FIFO structure, events (messages) have to be read until the message is returned that is propagated to the real-time application. The driver code for reading the receive FIFO is principally a while loop, which can add the problem of nondeterministic latency times.

To resolve the latency time issue in the Simulink Real-Time CAN FIFO drivers, define a receive FIFO read depth that is a constant number during real-time application execution. For example, assume that the FIFO read depth is `5`. Each time the Read Receive FIFO driver block is executed, the driver code reads and returns five events (messages) from the receive FIFO. This number is independent of how many events the FIFO currently contains. If the FIFO contains only two messages, the third to fifth attempts return the "No new event" code. Nevertheless, because the FIFO read latency does not exceed 17 µs, the latency time becomes deterministic and is the Read FIFO Depth multiplied by 17 µs.

The driver block returns all new events and therefore all CAN messages going over the network. If your real-time application processes only a small subset of the CAN messages received, the total latency can exceed the latency of dynamic object buffer mode.

For the read-receive FIFO side, FIFO mode creates another specific restriction. You cannot use more than one Read Receive FIFO block in a Simulink model. After one block instance reads a new event (message), another block instance cannot read it (the event was removed from the FIFO buffer). Therefore, you must concentrate the entire CAN receive part of your model in one Read Receive FIFO block. 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 system models. For more information, see “Acceptance Filters” on page 6-8.

Look again at the example of 12 messages to be received and 8 messages to be transmitted. If those 20 messages are the only messages going over the CAN network (100% usage ratio), then the total latency time is:

```
12*17 µs + 8*4 µs = 236 µs
```

This latency is considerably smaller than the 800 µs that results when you use dynamic object buffer mode drivers.

For the next case, assume that the network regularly passes only 12 additional messages that the real-time application does not process. Also, assume that the CAN controller acceptance filter cannot filter those messages. Then the total latency time increases to

```
12*17 µs +20*4 µs = 284 µs
```

There is little impact on the final result. The FIFO mode drivers are best suited for either CAN network monitoring or for low-latency CAN systems. In both cases, 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 can return additional information, such as the bus state or the reception of error frames. For this reason, FIFO mode drivers are especially suited for network monitoring. Dynamic object buffer mode drivers do not allow you to query such information.

This documentation only covers the differences between the dynamic object buffer mode drivers (standard drivers) and the FIFO mode drivers. 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 Simulink Real-Time.

If you use FIFO mode drivers in your model, you must replace the dynamic object buffer mode blocks (Setup, Send, Receive) with FIFO mode driver blocks. Because the CAN-AC2 boards from Softing do not let you run the two CAN ports in different modes, the mode has to be same for both ports. However, you can use more than one CAN board and run the boards in different just by selecting different I/O driver blocks.

FIFO mode drivers are 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 Simulink Real-Time I/O block library in the group CAN/Softing. The CAN-AC2-PCI SJA 1000 block group contains the FIFO mode CAN subgroup.

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 Simulink Real-Time I/O block library in the group CAN/Softing. The CAN-AC2-104 SJA 1000 block group contains the FIFO Mode subgroup.

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 real-time 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). A mask parameter and a code parameter define each acceptance filter.

- For each bit of the identifier, the mask parameter defines whether the filtering process cares about this bit or not. A `0` means “do not care” and a `1` means “do care.”
- For each bit that the filtering process cares about, the code parameter defines the bit value (`0` or `1`).

For standard identifiers, the mask parameter and code parameter must both be in the range `0`–`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 a bit is `1`, the message is voided.

According to this description, acceptance filters work using binary evaluation, while most systems 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 evaluates to `0`, all incoming messages reach the receive FIFO (no filtering takes place). The 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 network model where messages with the following identifiers (standard) are crossing the CAN network:

```
2-30, 48-122 (decimal)
```

The real-time application processes only incoming messages 4–29.

Ideally, all messages not having identifiers 4–29 would be filtered, but the mask and code parameters do not allow this scheme. You can, however, filter the messages with identifiers above 31 by using 2047 – 31 = 2016 for the mask parameter and 0 for the code parameter. The FIFO read driver block cannot filter messages with identifiers 0, 1, 2, and 3. The block returns them, even if the real-time application does not process them.

CANdb DBC Format Databases

You can use CANdb DBC files to specify the packing and unpacking of CANdb messages in a real-time 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.

CAN FIFO Loopback Tests

The Simulink Real-Time product contains the following examples that illustrate the use of the Softing CAN FIFO blocks. The models use CAN_MESSAGE data types. 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.

Example	Shows CAN I/O Communication Using FIFO Mode of the...
“CAN I/O FIFO Mode - Simple Use Case (with CAN_MESSAGE)”	Softing CAN-AC2-PCI board. It illustrates the basic functionality of the board.
“CAN I/O FIFO Mode - Simple Use Case (with CAN_MESSAGE and Function Call Output)”	Softing CAN-AC2-PCI board. It illustrates CAN I/O communication using a function call output format for the FIFO Read block.

7

CAN I/O FIFO Blocks

Softing CAN-AC2-PCI with Philips SJA1000 FIFO Setup

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

Library

Simulink Real-Time 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 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 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 a module is not 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 ]
```

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 arbitrary 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 the messages. For information on how to define the acceptance information to filter certain messages, see “Acceptance Filters” on page 6-8.

CAN 2 - physical bus

Defines the physical CAN bus type of CAN port 2. In the board's standard 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 a module is not 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 ]
```

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 arbitrary 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 messages. For information on how to define the acceptance information to filter certain messages, see “Acceptance Filters” on page 6-8.

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 network monitoring, where you want to be informed about all events going over the bus. For low-latency-time system models, 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 4-10.

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 4-10.

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)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

The board allows you to terminate each of the two CAN ports separately with 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.

See Also

External Websites

www.can-cia.org
company.softing.com/en

Softing CAN-AC2-PCI with Philips SJA1000 FIFO Write

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

Library

Simulink Real-Time 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, the required information must be provided to construct

valid CAN messages to be written into the transmit FIFO. For each CAN message, five elements are passed in this order:

`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.

`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.

This information can be dynamically changed in FIFO mode during real-time application execution. Therefore, you send the information through the block input instead of setting block parameters. To transmit more than one CAN message per block instance, use a matrix signal as a container.

The dimension of the matrix signal entering the block is `n*5`, where `n` is the number of CAN messages that you expect this block instance to send. Each row of the matrix signal defines one CAN message and each row combines the five information elements:

```
[Port, Identifier, Identifier type, Data frame size, Data]
```

For examples of constructing the matrix signal for the FIFO Write block, see “CAN FIFO Loopback Tests” on page 6-11.

For some models, you can 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`. The sixth column defines whether the corresponding CAN message is written into the transmit FIFO (`1`) or not (`0`).

Block Parameters

Board

Define the board used to send the CAN messages defined by this block instance. For more information about the meaning of the board number, see the Setup driver block. 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.

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 real-time 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.

See Also

External Websites

www.can-cia.org
company.softing.com/en

Introduced in R2006a

Softing CAN-AC2-PCI with Philips SJA1000 FIFO Read

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

Library

Simulink Real-Time 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*6`, where `m` is the FIFO Read depth defined in the block parameters. For example, if the FIFO read depth is `5`, then the matrix signal of port 1 has size `5*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 6-11.

If you choose output as double values in a matrix, each row with its six elements contains 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.

`Identifier`

Identifier of the CAN message being received. If it is a standard CAN message the range is 0 to 2047. It is an extended CAN message, the range is 0 to 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.

`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. 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 7-10 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 "`CAN I/O FIFO Mode - Simple Use Case (with CAN_MESSAGE)`" 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 "`CAN I/O FIFO Mode - Simple Use Case (with CAN_MESSAGE and Function Call Output)`" 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 FIFO 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 an event is not 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.

Code	Description
1	Error passive.
2	Bus off.

Sample time

Defines the sample time at which the FIFO Read block is executed during a real-time application run.

Note: Use only one instance of this block per physical CAN board in your model. Otherwise, one instance can read events that are intended for blocks connected to another instance.

See Also

External Websites

www.can-cia.org
company.softing.com/en

Introduced in R2006a

Softing CAN FIFO Read Filter

Softing CAN FIFO Read Filter block (architecture independent)

Library

Simulink Real-Time Library for CAN

Description

The FIFO Read Filter block is a utility block for the CAN FIFO driver block set. It is used to filter events out of the signal leaving the FIFO Read driver block (also known as the event matrix). It is connected to the first output port of the FIFO Read driver block. Because the FIFO Read Filter block does not actually access the CAN board or other device, it is independent of the bus architecture of the board.

The block code walks through the rows of the incoming event matrix signal and looks for events matching the criteria defined in the block parameters. If an event matches, the entire event information (row) is written to the 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, all of type `double`. The input port accepts a matrix signal of size `m*6`. The first output port, a row vector of size `1*6`, outputs the filtered event. The second, a scalar, outputs 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 consists of a concatenation of space-delimited keywords:

Keyword	Description
SDF	Standard data frame.

Keyword	Description
SRF	Standard remote frame.
EDF	Extended data frame.
ERF	Extended remote frame.
EF	Error frame.
NE	No new event.
CBS	Change of bus state.

Message type selection mode

Defines how the event type (message type) listed in **Message type command** 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 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, see “CAN FIFO Loopback Tests” on page 6-11.

See Also

External Websites

www.can-cia.org
company.softing.com/en

Introduced in R2013a

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

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

Library

Simulink Real-Time Library for CAN

Description

The FIFO Read XMT Level driver block reads the current number of CAN messages stored in the transmit FIFO that the firmware is to process. 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 driver ignores the new message. You can use this driver block to check for this condition and perform an error action. For example, you can stop execution or wait for the firmware to remove a message from the FIFO.

The block has a single output port of type double, returning a scalar value containing the current transmit FIFO level (number of messages to be processed).

Block Parameters

Board

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

Sample time

Defines the sample time at which the FIFO Read XMT Level driver block is executed during a real-time application run.

See Also

External Websites

www.can-cia.org
company.softing.com/en

Introduced in R2006a

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

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

Library

Simulink Real-Time Library for CAN

Description

The FIFO Reset XMT driver block resets the transmit FIFOs. Resetting the FIFOs deletes the messages currently stored in the transmit FIFO and resets the level counter to 0. As an example, you can use this driver block to reset the transmit FIFO after having detected a fault condition.

The block has a single input port of type double. If the input receives a scalar value of `1`, the transmit FIFO is reset. If it receives a scalar value of `0`, nothing is done.

Block Parameters

Board

Defines the board accessed to reset the transmit FIFO. For more information about the meaning of the board number, see the Setup driver block. 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 real-time application run.

See Also

External Websites

www.can-cia.org

company.softing.com/en

Introduced in R2006a

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

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

Library

Simulink Real-Time 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 a FIFO Read driver block does not attempt to read the stored messages, the driver ignores new incoming messages. Increments to the lost message counter show the number of lost messages. You can use this driver block to check for this condition and perform an error 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. 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 real-time application run.

See Also

External Websites

www.can-cia.org
company.softing.com/en

Introduced in R2006a

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

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

Library

Simulink Real-Time Library for CAN

Description

The FIFO Reset RCV driver block resets the receive FIFO. Resetting the FIFO deletes the messages currently stored in the receive FIFO and resets the level counter to `0`. As an example, you can use this driver block to reset the receive FIFO after your model detects a fault condition.

The block has a single input port of type double. If the input receives a scalar value of `1`, the transmit FIFO is reset. If the input receives a scalar value of `0`, nothing is done.

Block Parameters

Board

Defines the board accessed to reset the receive FIFO. For more information about the meaning of the board number, see the Setup driver block. If just one board is present in the target system, select board number 1.

Sample time

Defines the sample time at which the FIFO Reset RCV driver block is executed during a real-time application run.

See Also

External Websites

www.can-cia.org

company.softing.com/en

Introduced in R2006a

Softing CAN-AC2-104 with Philips SJA1000 FIFO Setup

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

Library

Simulink Real-Time 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 arbitrary 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 messages. For information on how to define the acceptance information to filter certain messages, see "Acceptance Filters" on page 6-8.

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 arbitrary 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 messages. For information on how to define the acceptance information to filter certain messages, see "Acceptance Filters" on page 6-8.

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 network monitoring, where you want to be informed about all events going over the bus. For low-latency-time models, 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 4-10.

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 4-10.

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.

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 4 bytes of I/O address space. It 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 (the board does not have a corresponding DIP switch). 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 Simulink Real-Time 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 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.

See Also

External Websites

www.can-cia.org
company.softing.com/en

Introduced in R2006a

Softing CAN-AC2-104 with Philips SJA1000 FIFO Write

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

Library

Simulink Real-Time 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 the 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.

`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.

This information can be dynamically changed in FIFO mode during real-time application execution. Therefore, you send the information through the block input instead of setting block parameters. To transmit more than one CAN message per block instance, use a matrix signal as a container.

The dimension of the matrix signal entering the block is `n*5`, where `n` is the number of CAN messages that you expect this block instance to send. Each row of the matrix signal defines one CAN message and each row combines the five information elements:

```
[Port, Identifier, Identifier type, Data frame size, Data]
```

For examples of constructing the matrix signal for the FIFO Write block, see "CAN FIFO Loopback Tests" on page 6-11.

For some models, you can make the writing of a CAN message into the transmit FIFO dependent on the model dynamics. In this case, the matrix signal can also be of dimension `n*6` instead of `n*5`. The sixth column defines whether the corresponding CAN message is written into the transmit FIFO (`1`) or not (`0`).

Block Parameters

Board

Defines the board used to send the CAN messages defined by this block instance. For more information about the meaning of the board number, see the Setup driver block. 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.

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.

Code	Description
-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 real-time 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.

See Also

External Websites

www.can-cia.org
company.softing.com/en

Introduced in R2006a

Softing CAN-AC2-104 with Philips SJA1000 FIFO Read

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

Library

Simulink Real-Time 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 out the events. The receive FIFO can store up to 127 events (messages). If it is full and a FIFO Read driver block does not attempt to read the stored events, the driver ignores new incoming events. Increments to the lost message counter tally the lost messages. You can use the FIFO Read RCV Level driver block to check for this condition and perform an error action. For example, you can stop the execution or reset 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*6`, where `m` is the FIFO read depth defined in the block parameters. For example, if the FIFO read depth is `5`, the matrix signal of port 1 has size `5*6`. Therefore, the matrix contains 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 6-11.

If you choose output as double values in a matrix, each row with its six elements contains the information defining a CAN message:

`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	
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 1µs.

`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. 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 7-10 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 “`CAN I/O FIFO Mode - Simple Use Case (with CAN_MESSAGE)`” 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 `"CAN I/O FIFO Mode - Simple Use Case (with CAN_MESSAGE and Function Call Output)"` 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 FIFO 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 receive FIFO. The read depth (m) also defines the size of the matrix signal (m*6) leaving the first output port. If an event is not 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 real-time application run.

Note: Use only one instance of this block per physical CAN board in your model. Otherwise, one instance can read events that are intended for blocks connected to another instance.

See Also

External Websites

www.can-cia.org
company.softing.com/en

Introduced in R2006a

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

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

Library

Simulink Real-Time Library for CAN

Description

The FIFO Read XMT Level driver block reads the current number of CAN messages stored in the transmit FIFO that the firmware is to process. 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 perform an error action. For example, you can stop the execution or wait for the transmit FIFO to empty.

The block has a single output port of type double, returning a scalar value containing the current transmit FIFO level (number of messages to be processed).

Block Parameters

Board

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

Sample time

Defines the sample time at which the FIFO Read XMT Level driver block is executed during a real-time application run.

See Also

External Websites

www.can-cia.org
company.softing.com/en

Introduced in R2006a

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

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

Library

Simulink Real-Time Library for CAN

Description

The FIFO Reset XMT driver block resets the transmit FIFO. Resetting deletes the 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, nothing is done.

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. 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 real-time application run.

See Also

External Websites

www.can-cia.org

company.softing.com/en

Introduced in R2006a

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

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

Library

Simulink Real-Time 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 a FIFO Read driver block does not attempt to read the stored events, the driver ignores new incoming events. Increments to the lost message counter mark the lost messages. You can use this driver block to check for this condition and perform an error action. For example, you can stop the execution or reset the receive FIFO.

The block has a single output port of type double, returning a scalar value containing the current receive FIFO level (number of messages to be processed).

Block Parameters

Board

Defines the board to access to read the current receive FIFO level. For more information about the meaning of the board number, see the Setup driver block. 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 real-time application run.

See Also

External Websites

www.can-cia.org
company.softing.com/en

Introduced in R2006a

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

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

Library

Simulink Real-Time Library for CAN

Description

The FIFO Reset RCV driver block resets the receive FIFO. Resetting deletes the messages currently stored in the receive FIFO and resets the level counter to 0. For example, you can use this driver block to reset the receive FIFO after detecting a fault condition.

The block has a single input port of type double. If a scalar value of 1 is passed, the transmit FIFO is reset; if 0 is passed nothing is done.

Block Parameters

Board

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

Sample time

Defines the sample time at which the FIFO Reset RCV driver block is executed during a real-time application run.

See Also

External Websites

www.can-cia.org

company.softing.com/en

Introduced in R2006a

8

Encoders

Handle Encoder Register Rollover

Encoder boards have a fixed size counter register of 16 bits, 24 bits, or 32 bits. Regardless of the size, the register eventually overflows and rolls over. Registers can roll over in either the positive or negative direction.

Some boards provide a mechanism to account for overflows or rollovers. As a best practice, design your model to deal with overflows or rollovers. Defining an initial count can handle the issue for some system models.

To handle register rollovers, use standard Simulink blocks to design the following counter algorithm types:

- Rollover Counter — Counts the number of times the output of an encoder block has rolled over. The counter increases for positive direction rollovers and decreases for negative direction rollovers.
- Extended Counter — Provides a rollover count not limited by register size. For an `n`-bit register, this counter must be able to count values greater than `2^(n-1)`.

The Incremental Encoder/Utilities/Rollover sublibrary of the Simulink Real-Time library contains example blocks for these two types of counters. See Rollover Counter and Extended Counter for further details. You can use these blocks in your model as is, or modify them for your model. Connect the output of the encoder block to these blocks.

Note: To view the algorithms used in these implementations, right-click the subsystem and select **Mask > Look Under Mask**.

Keep the following requirements in mind when using these blocks:

- Some driver blocks allow an initial starting value to be loaded into the register. To adjust for the starting value, pass it to the rollover blocks.
- The rollover block must know how many counts each rollover represents. Typically, this number is `2^n`, where `n` is the size of the register in bits.

9

Model-Based Ethernet Communications Support

Model-Based Ethernet Communications

In this section...

What Is Model-Based Ethernet Communications?

The Simulink Real-Time 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 real-time 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 IEEE® 802.3 standard.

By itself, raw Ethernet does not implement the TCP/IP or UDP standards. For information about modeling protocols built upon raw Ethernet, see "Real-Time UDP".

Ethernet Hardware

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 development and target computers. Therefore, your target computer must have at least two Ethernet cards, one to connect the development and target computers, and one for model-based Ethernet communication. The Simulink Real-Time 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 `Real-Time Ethernet Configuration` 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 information represents the Ethernet card that is installed on the target computer for dedicated communication between the development and target computers.
3. In the MATLAB Command Window, type

   ```
   tg = slrt;
   getPCIInfo(tg, 'ethernet')
   ```

   This command determines which Ethernet cards are installed in the target computer.
4. In the list, find the Ethernet card with a bus and slot different from the bus and slot that are displayed on the target computer monitor.
5. Note the PCI bus and slot of the free Ethernet card. Use the card for model-based Ethernet communications.

MAC Addresses

Several 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 character vector-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 character vector`*`')`, 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 Simulink Real-Time 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. These blocks specify criteria that the drivers use while parsing each buffer on the chain. Based on these criteria, the drivers 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. `Real-Time Ethernet Configuration`
2. The remaining raw Ethernet and network buffer library blocks

Simulink Real-Time Ethernet Block Library

To access the Simulink Real-Time Ethernet library blocks, in the Simulink Real-Time block library, double-click Ethernet. The Simulink Real-Time Ethernet library is displayed.

The Simulink Real-Time 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 for use in creating other subsystems. However, the top-level Ethernet blocks provide enough functionality for model-based Ethernet communications.

More About

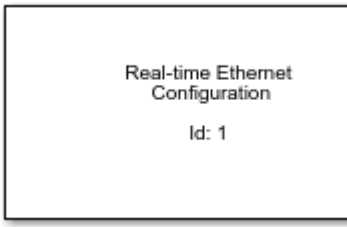
- "Real-Time Transmit and Receive over Ethernet"
- "Filtering on MAC Address"
- "Filtering on EtherType"

Ethernet Blocks

Real-Time Ethernet Configuration
Create Ethernet Packet
Ethernet Init
Ethernet Rx
Ethernet Tx
Extract Ethernet Packet
Filter Address
Filter Type
Header Extract

Real-Time Ethernet Configuration

Configure network interface for real-time raw Ethernet communication
Library: Ethernet



Description

To initialize the network and network buffers, use the Real-Time Ethernet Configuration block.

Parameters

Device

`Device ID` — Ethernet board identifier
`1-8`

From the list, select a unique number to identify the Ethernet board.

`Driver` — Drivers for chip families that this block supports
`Intel 8255X` (default) | `Intel Gigabit`

Identifies the driver for each chip family that the block supports.

`PCI bus` — PCI bus number of Ethernet card
`0` (default) | integer

Enter the PCI bus number for the Ethernet card.

`PCI slot` — PCI slot number of Ethernet card
`0` (default) | integer

Enter the PCI slot number for the Ethernet card.

`Sample time (-1 for inherited)` — Sample time of block
`-1` (default) | numeric

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

Addressing

`Address source` — Source of MAC address
`EEPROM` (default) | `Specify`

From the list, select:

- `EEPROM` — The block gets the Ethernet card MAC address that is built into the Ethernet card.
- `Specify` — Explicitly enter a MAC address for the Ethernet card.

Dependency

To see the **MAC** parameter, select `Specify`.

`MAC` — MAC address for Ethernet card
`macaddr('00:00:00:00:00:00')` (default) | `macaddr('xx:xx:xx:xx:xx:xx')`

Enter the MAC address for the Ethernet card.

Dependency

To make this parameter visible, set **Address source** to `Specify`.

`Rx promiscuous` — Receive all packets regardless of their destination address
`off` (default) | `on`

To direct the model to receive all packets regardless of their destination address, select this check box.

`Multicast address list` — List of multicast address vectors
`{}` (default) | cell array

Enter a list of multicast address vectors as a cell array. The Ethernet Rx block uses these addresses and the broadcast and unicast addresses.

Advanced

`Rx bad frames` — Receive all packets, including erroneous ones

`off` (default) | `on`

To direct the model to receive all packets, including erroneous ones (such as CRC error and alignment error), select this check box.

`Rx short frames` — Receive all packets, including short ones

`off` (default) | `on`

To direct the model to receive all packets, including frames that are less than 64 bytes in length, select this check box.

The Intel Gigabit Ethernet controller does not distinguish between bad packets and short packets. Therefore, selecting either **Rx Bad Frames** or **Rx Short Frames** produces the same results for **Driver** type `Intel Gigabit`.

`Max MTU` — Maximum transmission unit number

1518 (default) | numeric

Specify a maximum transmission unit number (MTU). With this parameter, you can specify a smaller maximum transmission unit number.

`Tx threshold` — Determine when device begins DMA on packets from memory

224 (default) | numeric

Enter a value that controls when the Ethernet device begins to perform direct memory access (DMA) on packets from memory.

This parameter applies only to **Driver** type `Intel 8255X`. Before you change this parameter, see *Intel 8255x 10/100 Mbps Ethernet Controller Family — Open Source Software Developer Manual*.

`Tx buffers` — Maximum number of queued transmit buffers

128 (default) | numeric

Enter the maximum number of buffers that the driver holds in the queue before it drops new transmit requests.

The number of buffers must be a multiple of 8.

`Rx buffers` — Maximum number of queued receive buffers
64 (default) | numeric

Enter the maximum number of buffers that the driver holds in the queue before it drops new receive packets.

The number of buffers must be a multiple of 8.

`Display tuning information` — Display statistical data
`off` (default) | `on`

To enable a display of statistical data collected during the run of the model, select this check box.

See Also

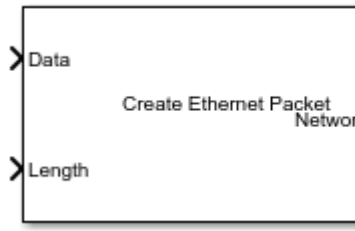
Topics
"Real-Time Transmit and Receive over Ethernet"

External Websites
www.iso.org

Introduced in R2014b

Create Ethernet Packet

Create Ethernet packet based on the MAC address and EtherType provided
Library: Ethernet



Description

To create the Ethernet packets that you want to transfer, use the Create Ethernet Packet block.

Ports

Input

`Data` — Payload data for Ethernet packet

vector

Data Types: `uint8`

`Length` — Number of bytes in data vector

scalar

Output

`Network Buffer` — Network buffer containing packet data

scalar

The parameter is a reference to the network buffer.

Parameters

`Destination MAC` — MAC address of the target computer to receive the data
`macaddr(00:1B:21:85:37:6C)` (default) | `macaddr(`*`xx:xx:xx:xx:xx:xx`*`)`

Enter the MAC address of the target computer that receives the data.

`EtherType (use 0 for length)` — EtherType or the use of Ethernet length
`hex2dec('0000')` (default) | numeric

Enter a value that represents either the EtherType or the use of Ethernet length:

- EtherType — If you are creating Ethernet packets that use EtherType values, to specify which prototype the Ethernet frame transfers, enter a valid EtherType value.
- Ethernet length — If you are creating Ethernet packets that use Ethernet lengths, enter `0`.

See Also

See Also
`Extract Ethernet Packet`

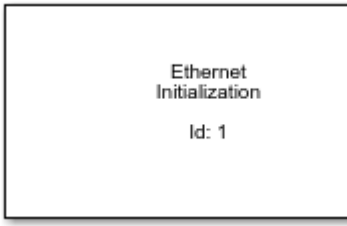
Topics
“Real-Time Transmit and Receive over Ethernet”
“Filtering on MAC Address”
“Filtering on EtherType”

External Websites
www.iso.org

Introduced in R2008b

Ethernet Init

Initialize network card for real-time raw Ethernet communication
Library: Ethernet



Description

To initialize the Ethernet communication channel, use the Ethernet Init block. Use a separate Ethernet Init block for each Ethernet board.

Note: The Ethernet Init and Buffer Mngmt blocks are combined in the `Real-Time Ethernet Configuration` block. For new development, use this block.

Parameters

Device

`Device ID` — Ethernet board identifier
`1-8`

From the list, select a unique number to identify the Ethernet board.

`Driver` — Drivers for chip families that this block supports
`Intel 8255X` (default) | `Intel Gigabit`

Identifies the driver for each chip family that the block supports.

`PCI bus` — PCI bus number of Ethernet card
`0` (default) | integer

Enter the PCI bus number for the Ethernet card.

`PCI slot` — PCI slot number of Ethernet card
`0` (default) | integer

Enter the PCI slot number for the Ethernet card.

`Sample time (-1 for inherited)` — Sample time of block
`-1` (default) | numeric

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

Addressing

`Address source` — Source of MAC address
`EEPROM` (default) | `Specify`

From the list, select:

- `EEPROM` — The block gets the Ethernet card MAC address that is built into the Ethernet card.
- `Specify` — Explicitly enter a MAC address for the Ethernet card.

Dependency

To see the **MAC** parameter, select `Specify`.

`MAC` — MAC address for Ethernet card
`macaddr('00:00:00:00:00:00')` (default) | `macaddr('xx:xx:xx:xx:xx:xx')`

Enter the MAC address for the Ethernet card.

Dependency

To make this parameter visible, set **Address source** to `Specify`.

`Rx promiscuous` — Receive all packets regardless of their destination address
`off` (default) | `on`

To direct the model to receive all packets regardless of their destination address, select this check box.

`Multicast address list` — List of multicast address vectors
`{}` (default) | cell array

Enter a list of multicast address vectors as a cell array. The Ethernet Rx block uses these addresses and the broadcast and unicast addresses.

Advanced

`Rx bad frames` — Receive all packets, including erroneous ones
`off` (default) | `on`

To direct the model to receive all packets, including erroneous ones (such as CRC error and alignment error), select this check box.

`Rx short frames` — Receive all packets, including short ones
`off` (default) | `on`

To direct the model to receive all packets, including frames that are less than 64 bytes in length, select this check box.

The Intel Gigabit Ethernet controller does not distinguish between bad packets and short packets. Therefore, selecting either **Rx Bad Frames** or **Rx Short Frames** produces the same results for **Driver** type `Intel Gigabit`.

`Max MTU` — Maximum transmission unit number
1518 (default) | numeric

Specify a maximum transmission unit number (MTU). With this parameter, you can specify a smaller maximum transmission unit number.

`Tx threshold` — Determine when device begins DMA on packets from memory
224 (default) | numeric

Enter a value that controls when the Ethernet device begins to perform direct memory access (DMA) on packets from memory.

This parameter applies only to **Driver** type `Intel 8255X`. Before you change this parameter, see *Intel 8255x 10/100 Mbps Ethernet Controller Family — Open Source Software Developer Manual*.

`Tx buffers` — Maximum number of queued transmit buffers
`128` (default) | numeric

Enter the maximum number of buffers that the driver holds in the queue before it drops new transmit requests.

The number of buffers must be a multiple of `8`.

`Rx buffers` — Maximum number of queued receive buffers
`64` (default) | numeric

Enter the maximum number of buffers that the driver holds in the queue before it drops new receive packets.

The number of buffers must be a multiple of `8`.

`Display tuning information` — Display statistical data
`off` (default) | `on`

To enable a display of statistical data collected during the run of the model, select this check box.

See Also

See Also
`Real-time Ethernet Configuration`

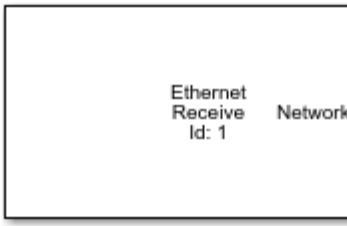
Topics
"Real-Time Transmit and Receive over Ethernet"

External Websites
www.iso.org

Introduced in R2008b

Ethernet Rx

Receive data over Ethernet network
Library: Ethernet



Description

To receive Ethernet packets and to filter on the received packets, use the Ethernet Rx block. 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.

Ports

Output

`Network Buffer` — Network buffer containing packet data
scalar

The parameter is a reference to the network buffer.

Parameters

Rx

`Device ID` — Ethernet board identifier
1-8

From the list, select a unique number to identify the Ethernet board. Select the same **Device ID** as the ID that you selected for the Real-Time Ethernet Configuration block.

`Sample time (-1 for inherited)` — Sample time of block
`-1` (default) | numeric

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

Filter

`Filter criteria` — Filter on EtherTypes or Ethernet lengths
`Receive all unmatched types [0 to 65535]` (default) | `Receive unmatched lengths [0 to 1500]` | `Receive unmatched EtherTypes [150 to 65535]` | `Specify types to match`

From the list, select how you want to filter on EtherTypes (Ethernet II framing standard) or Ethernet lengths (IEEE 802.3 framing standard).

- `Receive all unmatched types [0 to 65535]` — Output all unmatched packets, both Ethernet II framing and IEEE 802.3 framing standards.
- `Receive unmatched lengths [0 to 1500]` — Output all packets with IEEE 802.3 framing standard.
- `Receive unmatched EtherTypes [150 to 65535]` — Output all output packets with Ethernet II framing standard.
- `Specify types to match` — Explicitly enter the EtherTypes to output.

Dependency

To see the **Receive these types (vector of types 0–65535)** parameter, select `Specify types to match`.

`Receive these types (vector of types 0–65535)` — EtherTypes to output
`[hex2dec('0000')]` (default) | vector

Enter a vector of EtherTypes that you want to output.

Dependency

To make this parameter visible, set **Filter criteria** to `Specify types to match`.

See Also

See Also

`Ethernet Tx`

Topics

"Real-Time Transmit and Receive over Ethernet"
"Ethernet Rx Block Filtering"

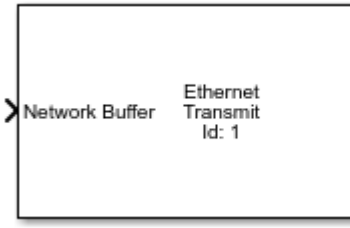
External Websites

www.iso.org

Introduced in R2008b

Ethernet Tx

Transmit data over Ethernet network
Library: Ethernet



Description

To send network packets, use the Ethernet Tx block.

Ports

Input

`Network Buffer` — Network buffer containing packet data
scalar

The parameter is a reference to the network buffer.

Parameters

`Device ID` — Ethernet board identifier
1 - 8

From the list, select a unique number to identify the Ethernet board. Select the same **Device ID** as the ID that you selected for the Real-Time Ethernet Configuration block.

`Sample time (-1 for inherited)` — Sample time of block
`-1` (default) | numeric

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

See Also

See Also

`Ethernet Rx`

Topics

"Real-Time Transmit and Receive over Ethernet"

External Websites

www.iso.org

Introduced in R2008b

Extract Ethernet Packet

Extract data from Ethernet packet
Library: Ethernet

Description

To extract data from an Ethernet packet, use the Extract Ethernet Packet block.

Ports

Input

`Network Buffer` — Network buffer containing packet data
scalar

The parameter is a reference to the network buffer.

Output

`Data` — Packet payload data
vector

Data Types: `uint8`

`Dst` — Ethernet address of packet destination
`'xxx.xxx.xxx.xxx'`

`Src` — Ethernet address of packet source
`'xxx.xxx.xxx.xxx'`

`Type` — EtherType of data
scalar

`Length` — Number of bytes in data vector
numeric

Parameters

`Data Size` — Number of bytes to extract
1500 (default) | numeric

Enter the data size (in bytes) for the data that you want to extract from an Ethernet packet.

See Also

See Also
`Create Ethernet Packet`

Topics
"Real-Time Transmit and Receive over Ethernet"

External Websites
www.iso.org

Introduced in R2008b

Filter Address

Filter Ethernet packets based on MAC address

Description

To filter network buffer packets by their MAC addresses, use the Filter Address block. See “Filter Type and Filter Address Blocks” on page 9-4 for cautions on setting the parameters for this block.

Ports

Input

`Network Buffer` — Network buffer containing packet data
scalar

The parameter is a reference to the network buffer.

Output

`Match [Addr (#)]` — Network buffer containing packets that match filter
vector

If you specify one MAC address, one port appears with the name `Match`. The block directs packets with this MAC address to port `Match`.

If you specify more than one MAC address, multiple ports appear, matched to the values in parameter **MAC Address**. The block directs packets with the first address to `Match Addr (1)`, with the second address to `Match Addr (2)`, and so on.

Dependency

The port count depends on how many MAC addresses appear in parameter **MAC Address**.

`Remainder` — Network buffer chain containing packets that do not match filter
vector of network buffers

Packets that do not meet the filter criteria appear at this port.

Dependency

To activate this port, clear **Drop non-matches**.

Parameters

`MAC Address` — MAC addresses to filter
`{[macaddr('00:00:00:00:00:00')]}` (default) | cell array

Enter a cell array that contains the MAC addresses for the filter.

`Drop non-matches` — Discard packets that do not match filter criteria
`'off'` (default) | `'on'`

To discard packets that do not match the filter criteria, select this parameter.

To output packets that do not match the filter criteria, clear this parameter.

`Filter on destination address` — Filter packets that match destination address
`'off'` (default) | `'on'`

To filter addresses for the source address, clear this check box (default).

To filter addresses for the destination address, select this check box.

See Also

See Also
`Filter Type`

Topics
“Filter Type and Filter Address Blocks” on page 9-4

"Real-Time Transmit and Receive over Ethernet"

External Websites

www.iso.org

Introduced in R2008b

Filter Type

Filter Ethernet packets based on EtherType

Description

To filter network buffer packets by their EtherType values, use the Filter Type block. See “Filter Type and Filter Address Blocks” on page 9-4 for cautions on setting the parameters for this block.

Ports

Input

`Network Buffer` — Network buffer containing packet data
scalar

The parameter is a reference to the network buffer.

Output

`Match Length` — Network buffer chain containing packets that match length filter
vector of network buffers

One port receives packets with EtherType values within `1–1500`.

Dependency

To activate this port, select parameter **Match Length (1-1500)**.

`Match [Type (#)]` — Network buffer chain containing packets that match filter
vector of network buffers

If you specify one Ethertype, one port appears with the name `Match`. The block directs packets with this EtherType to port `Match`.

If you specify more than one EtherType, multiple ports appear, matched to the values in parameter **EtherType**. The block directs packets with the first **EtherType** to `Match Type (1)`, with the second **EtherType** to `Match Type (2)`, and so on.

Dependency

The port count depends on how many EtherTypes appear in parameter **EtherType**.

`Remainder` — Network buffer chain containing packets that do not match filter

vector of network buffers

Packets that do not meet the filter criteria appear at this port.

Dependency

To activate this port, clear **Drop non-matches**.

Parameters

`Match Length (1-1500)` — Match packets with EtherType values within 1–1500

`'on'` (default) | `'off'`

To match packets whose **EtherType** values fall within the range 1–1500, select this check box.

`EtherType` — EtherTypes on which to filter

`[hex2dec('0000')]` (default) | vector

Enter a vector of EtherTypes on which you want to filter.

`Drop non-matches` — Discard packets that do not match filter criteria

`'off'` (default) | `'on'`

To discard packets that do not match the filter criteria, select this parameter.

To output packets that do not match the filter criteria, clear this parameter.

See Also

See Also

`Filter Type`

Topics

“Filter Type and Filter Address Blocks” on page 9-4
“Real-Time Transmit and Receive over Ethernet”

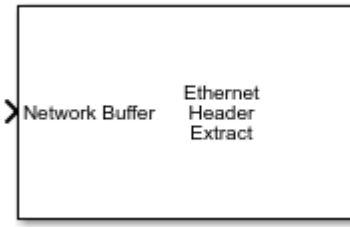
External Websites

www.iso.org

Introduced in R2008b

Header Extract

Extract header data from Ethernet packet
Library: Ethernet



Description

To extract the header data of network buffer packets, use the Header Extract block.

Ports

Input

`Network Buffer` — Network buffer containing packet data
scalar

The parameter is a reference to the network buffer.

Output

`Data` — Packet payload data
vector

Data Types: `uint8`

`Dst` — Ethernet address of packet destination
`'xxx.xxx.xxx.xxx'`

`Src` — Ethernet address of packet source
`'xxx.xxx.xxx.xxx'`

`Type` — EtherType of data
scalar

`Length` — Number of bytes in data vector
numeric

See Also

See Also
`Extract Ethernet Packet`

Topics
"Real-Time Transmit and Receive over Ethernet"

External Websites
www.iso.org

Introduced in R2008b

11

Network Buffer Library for Model-Based Ethernet Communications Support

Network Buffer Blocks

The Ethernet library includes a sublibrary, Network Buffers, that contains blocks for managing Ethernet network buffers. The blocks in this sublibrary are core blocks that you can use to create other subsystems.

The Ethernet drivers use a set of buffers, *Ethernet network buffers*, that it uses to store data that is sent and received over the network. The block organizes these buffers into several pools, each with different values of maximum data size. The buffers include information about the data itself. The block allocates the buffer pools during model initialization and does not change the buffer pools during model run time. When the block sends, receives, or processes data, it allocates a buffer. When the operation is done, it frees the buffer.

You can control the number of buffers allocated for each allowable value of data size by using the Buffer Mngmt block parameter **Buffer pool sizes**. Allocate enough buffers for the maximum number of data packets that you anticipate receiving, sending, or processing at one time. You can send and receive more data by allocating many more buffers. However, each allocation reserves more memory, which you cannot then use for other purposes. Running out of buffers means that no data can be sent and received until the block frees allocated buffers.

Monitor the buffer pool statistics at run time to find the optimal values that an application requires. To monitor the buffer pool statistics, select the **Display tuning information** check box in the Buffer Mngmt block parameters dialog box.

See Also

`Buffer Mngmt`

12

Network Buffer Library Blocks

Buffer Mngmt
Chain Size
Compose
Extract
Link
Manage
Merge
Split
Unlink

Buffer Mngmt

Initialize network buffer pools

Library

Simulink Real-Time Library for Ethernet

Description

To initialize network buffers, use the Buffer Mngmt block.

Block Parameters

This block has two tabs, **Main** and **Advanced**.

Main

Sample time

Enter the base sample time or a multiple of the base sample time (`-1` means sample time is inherited).

Advanced

Modify the values of the parameters in this tab only if you have a thorough understanding of the Ethernet protocol. Changing the values of these parameters can change the behavior of your system.

Buffer pool sizes (256, 512, 1024, 2048)

Enter a vector of the number of buffers for each pool size (`256`, `512`, `1024`, or `2048`).

Display tuning information

Select this check box to enable a display of statistical data collected during the run of the model.

See Also

Topics

"Network Buffer Blocks" on page 11-2

External Websites

www.iso.org

Introduced in R2008b

Chain Size

Determine the number of network buffers in the chain

Library

Simulink Real-Time Library for Ethernet

Description

To determine the number of buffers that are on the chain, use the Chain Size block.

See Also

Topics

"Real-Time Transmit and Receive over Ethernet"

External Websites

www.iso.org

Introduced in R2008b

Compose

Create a network buffer from raw input data

Library

Simulink Real-Time Library for Ethernet

Description

To create a network buffer, use the Compose block. This block creates a pointer to a network buffer.

See Also

Topics

"Real-Time Transmit and Receive over Ethernet"

External Websites

www.iso.org

Introduced in R2008b

Extract

Extract raw data from network buffer

Library

Simulink Real-Time Library for Ethernet

Description

To extract network buffer packets, use the Extract block.

Block Parameters

Packet size (-1: inherit)

Enter the packet size for the network buffer packet to extract. Enter `-1` (default) to inherit the packet size.

See Also

Topics

"Real-Time Transmit and Receive over Ethernet"

External Websites

www.iso.org

Introduced in R2008b

Link

Link vector of network buffers into a chain

Library

Simulink Real-Time Library for Ethernet

Description

To convert a vector of network buffer signals into a linked list of signals, use the Link block.

See Also

Topics

“Real-Time Transmit and Receive over Ethernet”

External Websites

www.iso.org

Introduced in R2008b

Manage

Output or buffer packets as indicated by the parameters

Library

Simulink Real-Time Library for Ethernet

Description

Output or buffer packets as indicated by the parameters.

Block Parameters

Chain size

Specify the queuing (output) behavior of the block as packets are received.

Value	Description
`inf`	Output all packets. No queuing occurs.
`0`	Delete all packets, no packets pass through.
Positive number, *C*	Pass through the first *C* packets
Negative number, *C*	Pass through the last *C* packets

Buffer size

Specify the buffering behavior of the block as packets are received.

Value	Description
`inf`	Buffer all remaining packets. Delete no packets.
`0`	Do not buffer packets. Delete all remaining packets.
Positive number, *B*	Buffer the remaining first *B* packets.

Value	Description
Negative number, *B*	Buffer the remaining last *B* packets.

Threshold

Enter a minimum threshold before which this block begins to output buffers.

Value	Description
0	Specifies no threshold.
Negative number,*T*	Delay passing buffer packets until *T* packets are buffered.
Positive number, *T*	Passes buffer packets if *T* packets are buffered.

See Also

Topics

"Real-Time Transmit and Receive over Ethernet"

External Websites

www.iso.org

Introduced in R2015a

Merge

Merge the incoming network buffer chains into one

Library

Simulink Real-Time Library for Ethernet

Description

To combine signal pointers to a linked list, use the Merge block.

Block Parameters

Number of inputs

Enter the number of network buffer signal pointers to combine into a linked list.

See Also

Topics

"Real-Time Transmit and Receive over Ethernet"

External Websites

www.iso.org

Introduced in R2008b

Split

Split a network buffer chain

Library

Simulink Real-Time Library for Ethernet

Description

To separate a linked list of buffer pointers into separate individual pointers, use the Split block.

Block Parameters

Number of outputs

Enter the number of pointers the input linked list should be separated into.

- 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 **Split in reverse order** parameter).
- If the number of buffers is less than **Number of outputs**, the block outputs zeros (0s) for the extra output ports.
- If the number of buffers is greater than **Number of outputs**, the block either deletes the extra buffers, or chains the remaining buffers together (depending on the setting of the **Allow chaining for last signal** parameter).

Split in reverse order

Select this check box to split out the network buffers in the reverse order in which they are received.

Allow chaining for last signal

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 **Number of outputs**.

Clear this check box to delete the remaining buffers.

See Also

Topics

"Real-Time Transmit and Receive over Ethernet"

External Websites

www.iso.org

Introduced in R2008b

Unlink

Unlink a chain into a vector of network buffers

Library

Simulink Real-Time Library for Ethernet

Description

To convert a linked list of signals into a vector of network buffer signal, use the Unlink block.

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.

See Also

Topics

"Real-Time Transmit and Receive over Ethernet"

External Websites

www.iso.org

Introduced in R2008b

13

Model-Based EtherCAT Communications Support

Modeling EtherCAT Networks

Ethernet for Control Automation (EtherCAT) is an open Ethernet network protocol for real-time distributed control, for example for automotive and industrial systems. The EtherCAT protocol provides:

- Deterministic and fast cycle times
- Inexpensive I/O module cost

EtherCAT networks consist of one master node and several slave nodes. The Simulink Real-Time EtherCAT sublibrary supports only the master node of an EtherCAT network. You cannot emulate slave nodes using the blocks in the EtherCAT sublibrary. However, you can use these blocks to prototype multiple EtherCAT networks with multiple Ethernet cards.

You model an EtherCAT network using one of the third-party EtherCAT configurators: the Beckhoff® ET9000, the Beckhoff TwinCAT®, or the Acontis EC-Engineer. To map the network model into a Simulink Real-Time model, become familiar with the following mappings:

- "EtherCAT Protocol Mapping" on page 13-28
- "EtherCAT Configurator Component Mapping" on page 13-29

Blocks and Tasks

At a minimum, each EtherCAT model must contain an EtherCAT Init block. The EtherCAT Init block contains a reference to an EtherCAT configuration. The file describes the network, including the device variables of the network. It also contains information identifying the Ethernet card for the EtherCAT network that the software accesses.

If you generate the configuration file using the Beckhoff ET9000 or the Beckhoff TwinCAT software, use the software to create at least one cyclic input/output task. Link this task to at least one input channel and one output channel on each slave device. If you generate the file using Acontis EC-Engineer, the software creates one default task linked to all slave device input/output channels.

When you know the input/output task rates, set the **Fixed-step size** in the Configuration Parameters dialog box to a value consistent with the following constraints:

- The task rates of all EtherCAT slave devices.
- The sample times of all other blocks in the Simulink model.

When you know the device variables that you are using in your model, add an EtherCAT PDO Receive or EtherCAT PDO Transmit block for every EtherCAT device variable. When you add these blocks to the model, the block obtains the list of device variables from the configuration file in the EtherCAT Init block. When you specify a device variable in the block dialog box, the software updates the block information with device variable information from the configuration file.

To transmit CANopen over EtherCAT (CoE) information through your network, add SDO Upload and SDO Download blocks to your model. The SDO blocks come in two types, synchronous and asynchronous. From the EtherCAT perspective, there is little difference in behavior of these types. The difference arises during the execution of the real-time application. The Sync SDO blocks halt execution while they wait for a response. The Async SDO blocks continue executing and poll the I/O module for a response. To avoid a CPU overload, use a slower execution rate for the SDO blocks than for the PDO blocks.

To track the state of the network or force the network into a particular state, add an EtherCAT Get State or EtherCAT Set State block.

Order of Network Events

Simulink Real-Time EtherCAT schedules network events according to the order of execution of the EtherCAT blocks. There are two major rate groups: one for the EtherCAT Sync SDO Upload and EtherCAT Sync SDO Download blocks, and another for the rest of the EtherCAT blocks.

1. EtherCAT Init (phase 1) — Reads data from EtherCAT variables from the last received frame into EtherCAT PDO Receive blocks.
2. Either of the following blocks, in arbitrary order:
   - EtherCAT PDO Receive — Processes data read from the last frame received from a slave device.
   - EtherCAT PDO Transmit — Buffers data to send in the next frame to a slave device.
3. Each of the following blocks, in arbitrary order:
   - EtherCAT Sync SDO Upload — Queues an SDO frame with new value, waits for response.

- EtherCAT Sync SDO Download — Queues an SDO frame with request for data, waits for response.
- EtherCAT Async SDO Upload — Queues an SDO frame with new value, checks for response, continues execution.
- EtherCAT Async SDO Download — Queues an SDO frame with request for data, checks for response, continues execution.

  Upload and download take at least three ticks of the fastest PDO cycle time to complete processing.
- EtherCAT Get State — Reads current state of EtherCAT network.
- EtherCAT Set State — Queues request to change current state of EtherCAT network.

4 EtherCAT Init (phase 2) — Sends the PDO frames, followed by the next available queued SDO frames.

See Also

`EtherCAT PDO Transmit` | `EtherCAT Set State` | `EtherCAT Async SDO Download` | `EtherCAT Async SDO Upload` | `EtherCAT Get State` | `EtherCAT Init` | `EtherCAT PDO Receive` | `EtherCAT Sync SDO Download` | `EtherCAT Sync SDO Upload`

More About

- "Fixed-Step Size Derivation" on page 13-27
- "EtherCAT Protocol Mapping" on page 13-28
- "EtherCAT Configurator Component Mapping" on page 13-29
- "Ethernet Chip Sets Compatible with EtherCAT" on page 13-30
- "EtherCAT Data Types" on page 13-33
- "EtherCAT Init Block DC Error Values" on page 13-34

Install EtherCAT Network and Configuration Software

To install the EtherCAT network and configuration software, execute the following steps. For requirements, see “EtherCAT Installation and Setup Requirements” on page 13-6.

1. Record the PCI bus and PCI slot for the existing Ethernet cards in the development computer. Include cards used to access your local area network and the card used to link the development and target computers.
2. Install a dedicated, EtherCAT compatible Ethernet card on the development computer.
3. Record the PCI bus and PCI slot for the new Ethernet card. Check the PCI bus and PCI slot for the existing cards.
4. Download or purchase the Beckhoff ET9000 EtherCAT Configurator software (`www.beckhoff.com`).
5. Install the Beckhoff ET9000 software on your development computer.

The next task is “Configure EtherCAT Network” on page 13-7.

External Websites

- www.beckhoff.com
- www.acontis.com/eng

EtherCAT Installation and Setup Requirements

For both the development and target computers, the EtherCAT I/O module has the following requirements:

- Each Ethernet card must be compatible with EtherCAT communication. For a list of supported chip sets, see “Ethernet Chip Sets Compatible with EtherCAT” on page 13-30.
- Each Ethernet card must use a static IP address and a nonroutable subnet.

  For information on setting up the dedicated Ethernet card, see your network administrator.
- On the target computer, install two Ethernet cards. Dedicate one card to linking the development and target computers. Dedicate the other to model-based EtherCAT communication.
- On the development computer, as a best practice, install two Ethernet cards in addition to your local area network card. Dedicate one card to linking the development and target computers. Dedicate the other to EtherCAT network configuration.

With only one card on the development computer, before configuring the EtherCAT network, unplug the Ethernet link cable and plug in the EtherCAT network cable. Before building and downloading the model, unplug the EtherCAT network cable, plug in the Ethernet link cable, disable the EtherCAT filter, and restart your development computer.

The Beckhoff ET9000 software runs only on a 32-bit Windows® compatible computer. Configure the development computer Ethernet card to enable only the Internet Protocol Version 4 (TCP/IPv4) driver. See the Beckhoff ET9000 documentation for information on manually creating an EtherCAT configuration file.

Configure EtherCAT Network

To configure the EtherCAT network using the Beckhoff ET9000 EtherCAT Configurator, execute the following steps. The model comes from example “EtherCAT® Communication with Beckhoff® Analog IO Slave Devices EL3062 and EL4002”.

Before configuring the network, carry out the steps in “Install EtherCAT Network and Configuration Software” on page 13-5.

Scan EtherCAT Network

To scan an EtherCAT network using the Beckhoff ET9000 EtherCAT Configurator program:

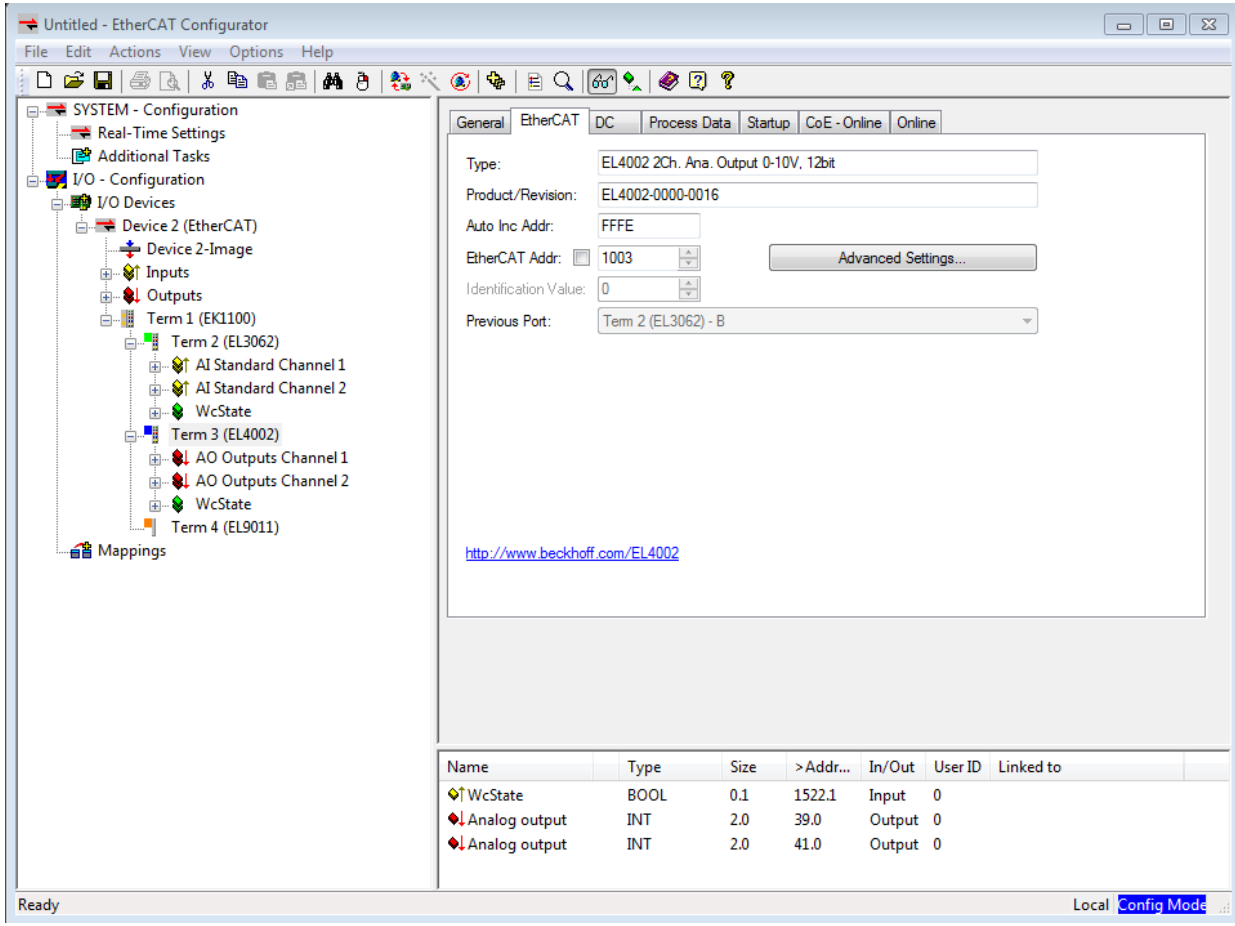
1. Connect your EtherCAT network to the development computer Ethernet port dedicated to EtherCAT. Turn on the network.
2. Start the Beckhoff ET9000 software from Windows.
3. Right-click **I/O Devices** and select **Scan Devices**.

   A dialog box opens stating that not all types of devices can be found automatically.
4. Click **OK**.

   A dialog box opens listing Ethernet devices on the development computer.
5. Select the check box next to the Ethernet device into which you plugged your EtherCAT network. If you do not see an Ethernet device identified as an EtherCAT device, check your EtherCAT network configuration and power supply.
6. Click **OK**.
7. In the dialog box, click **Yes**.

   An EtherCAT device hierarchy appears under **I/O Devices**. A dialog box opens asking if you want to activate free run mode.
8. Click **No**.
9. Expand **Term 1**, **Term 2**, and **Term 3**. then click the **EtherCAT** tab.

   The Beckhoff ET9000 window looks like this figure.



Configure EtherCAT Master Node Data

Before configuring the master node of an EtherCAT network, scan the network using the Beckhoff ET9000 EtherCAT Configurator program.

To configure the master node, execute the following steps.

Create EtherCAT Task

Before creating an EtherCAT task and configuring its time settings, scan the network using the Beckhoff ET9000 EtherCAT Configurator program.

To create and configure an EtherCAT task, execute the following steps.

1. Right-click **Additional Tasks** and select **Append Task**.

   A dialog box opens prompting you for a name and comments.

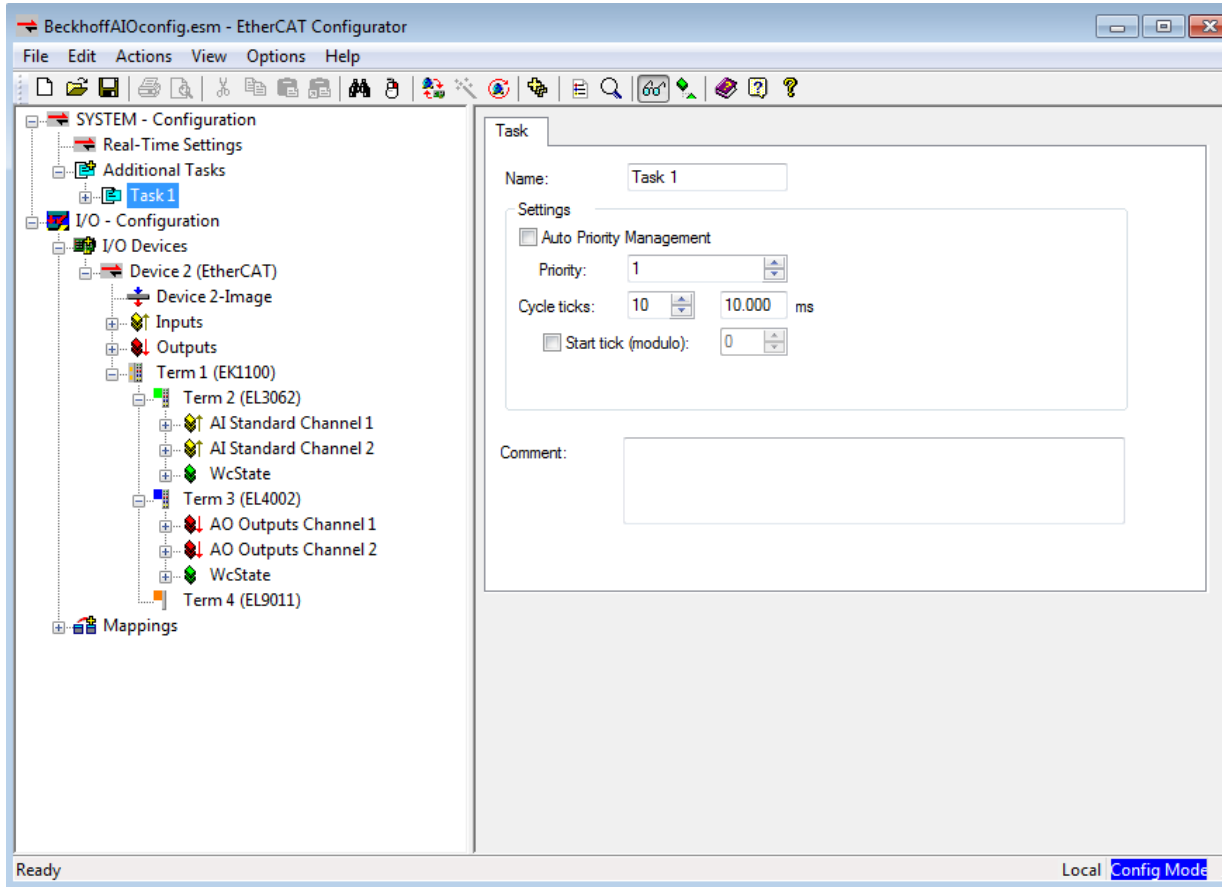
2. To accept the default name, here `Task 1`, click **OK**.

   An EtherCAT task hierarchy appears under **Additional Tasks**.

3. Click node **Task 1**, and then set the **Cycle ticks** box to `10` in the **Task** tab. This value sets the cycle time to `10` milliseconds, displayed in the box labeled **ms**.

   Note the cycle time in milliseconds. In the Configuration Parameters dialog box, you use the cycle time to calculate a value for the **Fixed-step size (fundamental sample time)** box.

   The Beckhoff ET9000 window looks like this figure.



Configure EtherCAT Task Inputs

To configure the inputs of an EtherCAT task, add and configure a task using the Beckhoff ET9000 EtherCAT Configurator program.

To configure the task inputs, execute the following steps.

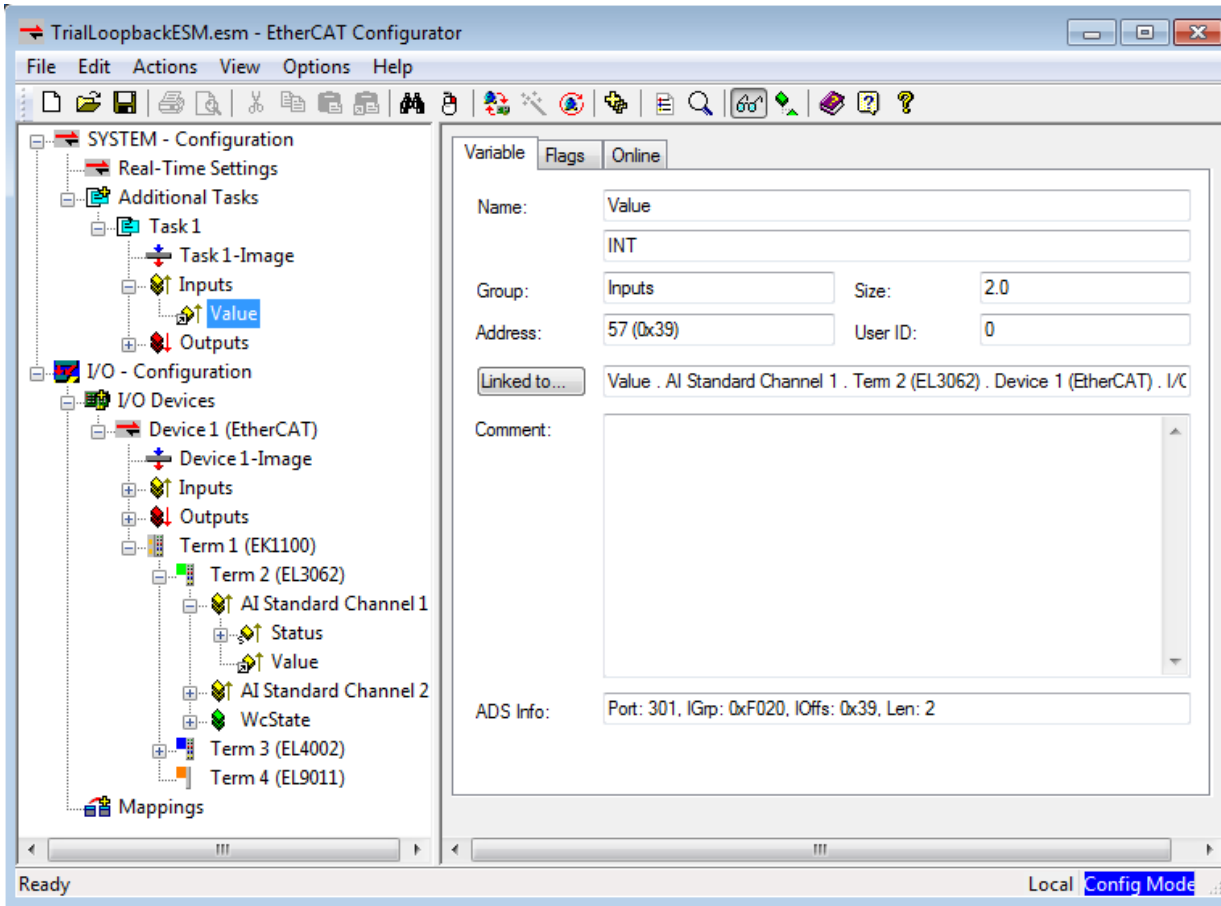
1. Expand nodes **Term 2** and **AI Standard Channel 1**.
2. Click and drag node **Value** under **AI Standard Channel 1** to **Inputs** under **Task 1**.
3. Click **Value** under **Inputs** and select the **Variable** tab.

4 Click the **Linked to** button.

A dialog box opens for attaching a variable to an input.

5 Select the first **Value** entry under **Term 2**, and then click **OK**.

The Beckhoff ET9000 window looks like this figure.



Configure EtherCAT Task Outputs

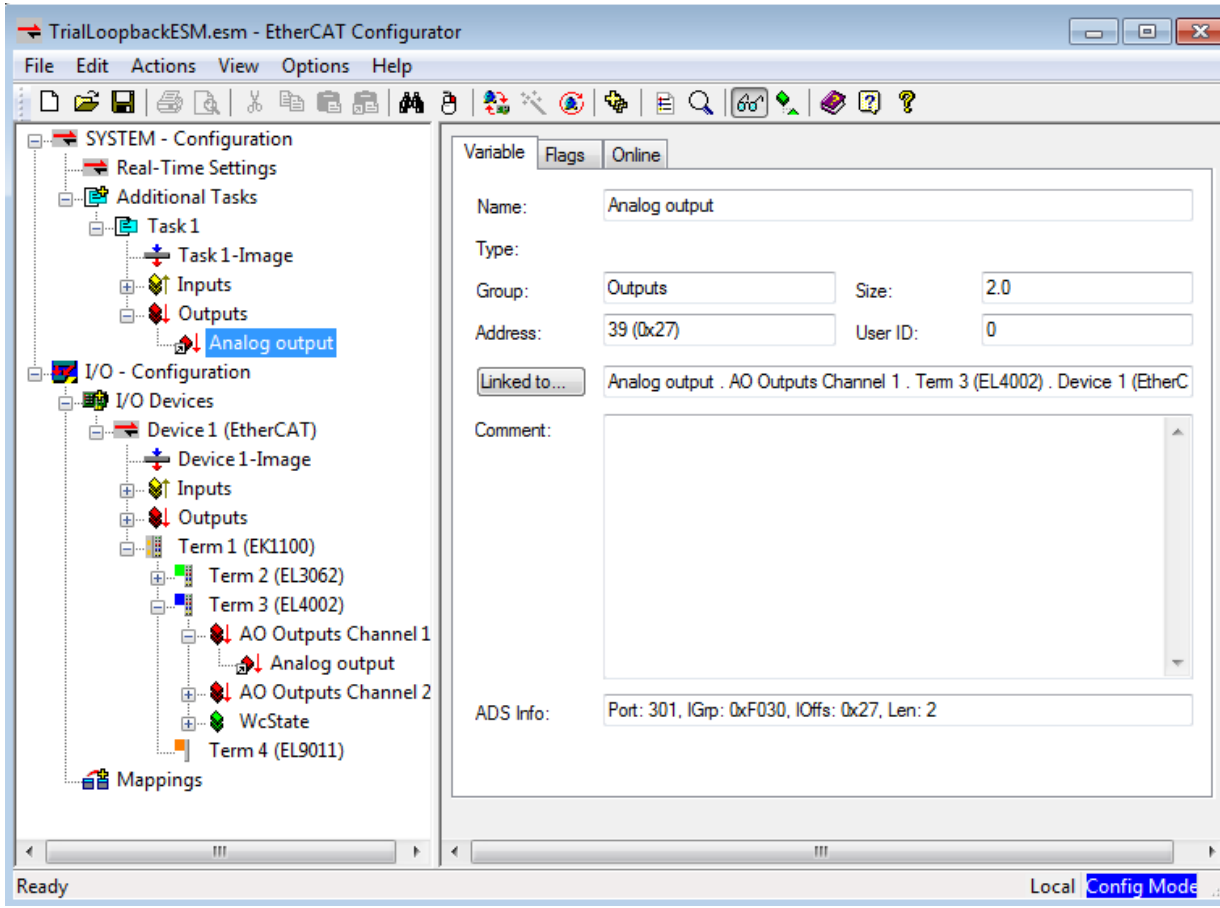
To configure the outputs of an EtherCAT task, add and configure a task using the Beckhoff ET9000 EtherCAT Configurator program.

To configure the task outputs, execute the following steps.

1. Expand nodes **Term 3** and **AO Outputs Channel 1**.
2. Click and drag node **Analog output** under **AO Outputs Channel 1** to **Outputs** under **Task 1**.
3. Click **Analog output** under **Outputs** and select the **Variable** tab.
4. Click the **Linked to** button.

   A dialog box opens for attaching a variable to an output.
5. Select the first **Analog output** entry under **Term 3** and click **OK**.

The Beckhoff ET9000 window looks like this figure.



Export and Save EtherCAT Configuration

The EtherCAT Network Information (ENI) file represents the master node of an EtherCAT network. To create the ENI file, scan and configure the network using the Beckhoff ET9000 EtherCAT Configurator program. To export the ENI file, execute the following steps.

1 Click node **Device 1 (EtherCAT)**, and then click the **EtherCAT** tab.

2 Click **Export Configuration File**.

3. In the file save dialog box, enter an XML file name, such as `BeckhoffAIOconfig.xml`, and then click **Save**. Do not name the ENI file required by a Simulink Real-Time model already named `modelname` with the name `modelname.xml`.
4. Click **File** > **Save**.
5. In the file save dialog box, enter an ESM file name, such as `BeckhoffAIOconfig.esm`, and then click **Save**.

To review or modify your configuration, open the ESM file using the Beckhoff ET9000 EtherCAT Configurator program. If you modify the configuration, save both the XML and ESM files. If the ESM file is not present, the Beckhoff ET9000 program cannot open the XML file. The same is true for the Beckhoff TwinCAT program.

The next task is “Install EtherCAT Network for Execution” on page 13-15.

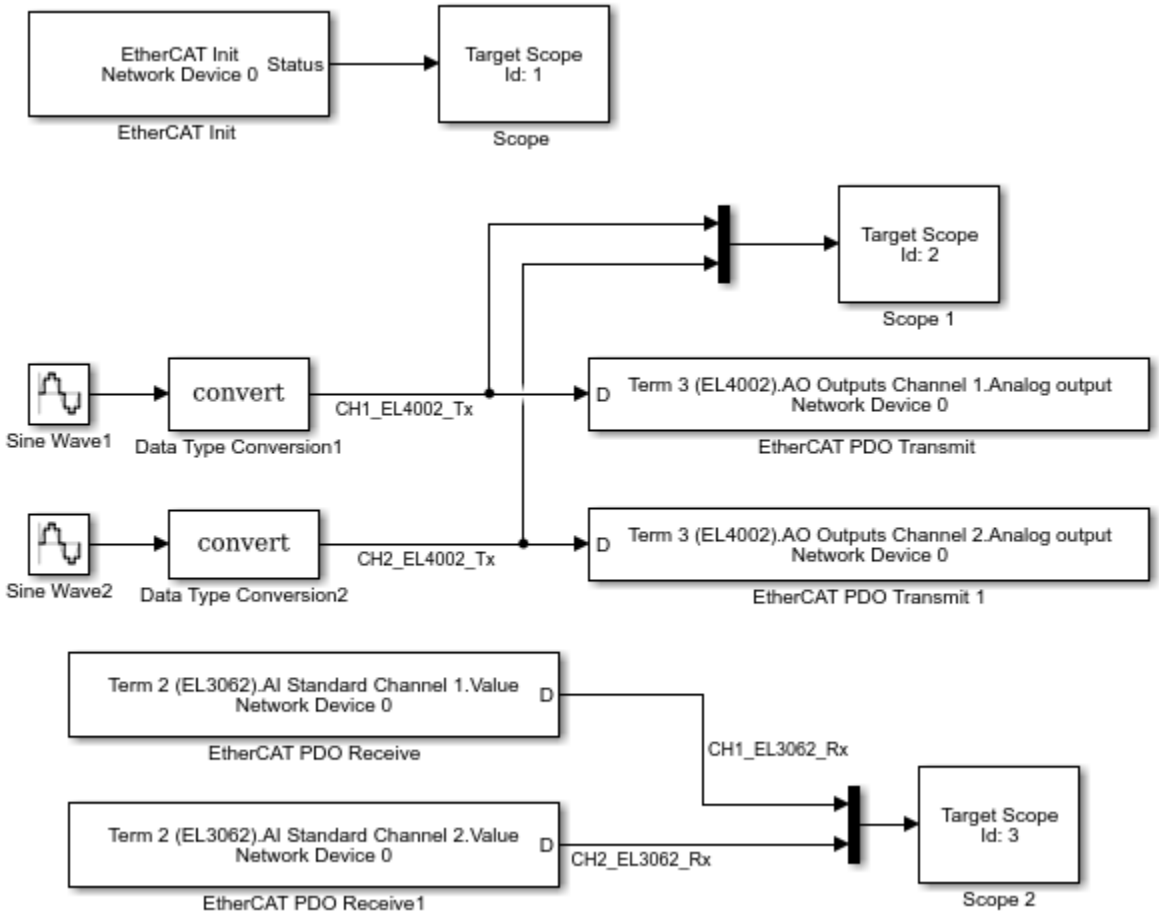
Install EtherCAT Network for Execution

To install the EtherCAT Network for execution using the target computer as master node, execute the following steps. For requirements, see “EtherCAT Installation and Setup Requirements” on page 13-6.

1. Record the PCI bus and PCI slot for the existing Ethernet cards in the target computer. For more on identifying and selecting Ethernet cards for linking the development and target computers, see “Ethernet Card Selection by Index”.
2. Install a dedicated, EtherCAT compatible Ethernet card on the target computer.
3. Record the PCI bus and PCI slot for the new Ethernet card. Check the PCI bus and PCI slot for the existing card. To select the Ethernet card required for the Ethernet link, update the Simulink Real-Time environment settings.
4. Connect your EtherCAT network to the target computer Ethernet port dedicated to EtherCAT. Turn on the network.

The next task is “Configure EtherCAT Master Node Model” on page 13-16.

Configure EtherCAT Master Node Model



Before configuring the model, carry out the steps in “Configure EtherCAT Network” on page 13-7.

To configure model `xpcEthercatBeckhoffAIO` for execution using the target computer as master node, execute the following steps.

Configure EtherCAT Init Block

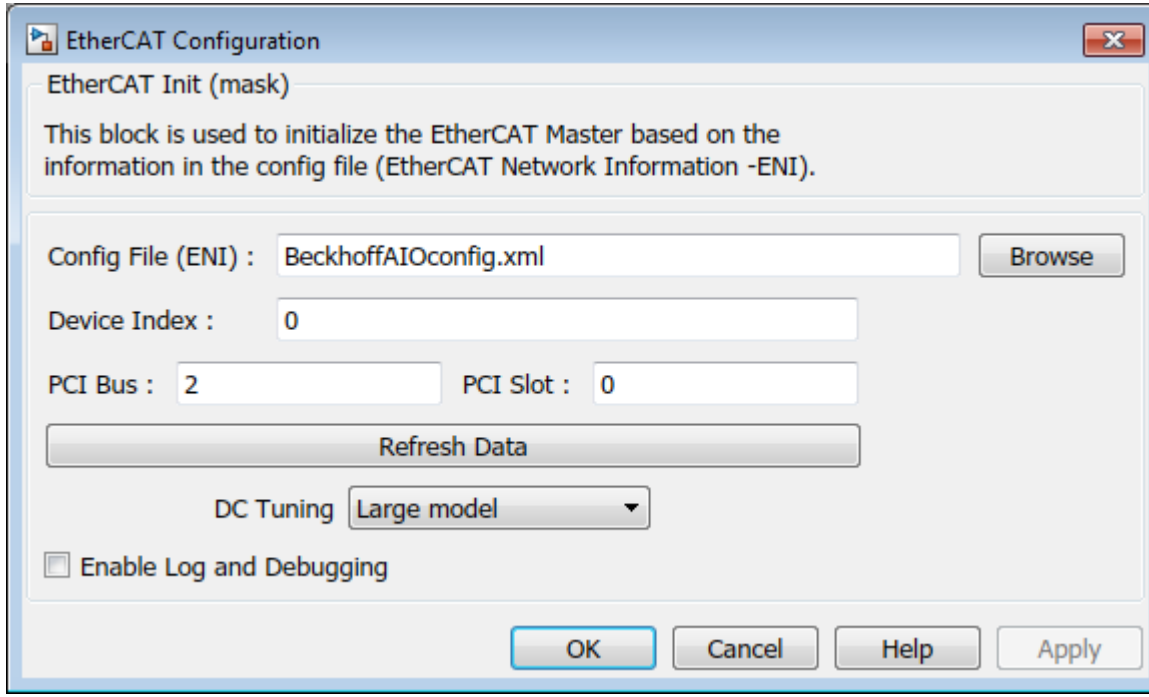
Before you use the EtherCAT Init block, configure the EtherCAT network using the Beckhoff ET9000 configurator. As part of this process, create and save an EtherCAT Network Information (ENI) file. See “Configure EtherCAT Network” on page 13-7.

If you configure EtherCAT distributed clocks in master shift mode, using the IEEE 1588 Sync Execution block in the same model produces a build error. To include EtherCAT distributed clocks and IEEE 1588 synchronized execution in the same model, use EtherCAT bus shift mode.

To configure the `EtherCAT Init` block of model `xpcEthercatBeckhoffAIO`, execute the following steps.

1. Open model `xpcEthercatBeckhoffAIO`.
2. Double-click the EtherCAT Init block.
3. At the **Config file (ENI)** text box, browse to the EtherCAT Network Information (ENI) file that you created when you configured the network (here, `'BeckhoffAIOconfig.xml'`). You can enter the file name with or without single quotes.
4. Take the default value `0` for parameter **Device index**.

   If the model includes more than one EtherCAT network, enter a unique **Device index** for each network. Enter the same value for all blocks in each network.
5. Enter the **PCI bus** and **PCI slot** you recorded while configuring the EtherCAT I/O module for execution. See “Install EtherCAT Network for Execution” on page 13-15.
6. Take the default value `Large model` for parameter **DC Tuning**.



7 To read the ENI file and store the data in the EtherCAT Init block, click **Refresh Data**.

8 Click **OK**.

Configure EtherCAT PDO Receive Blocks

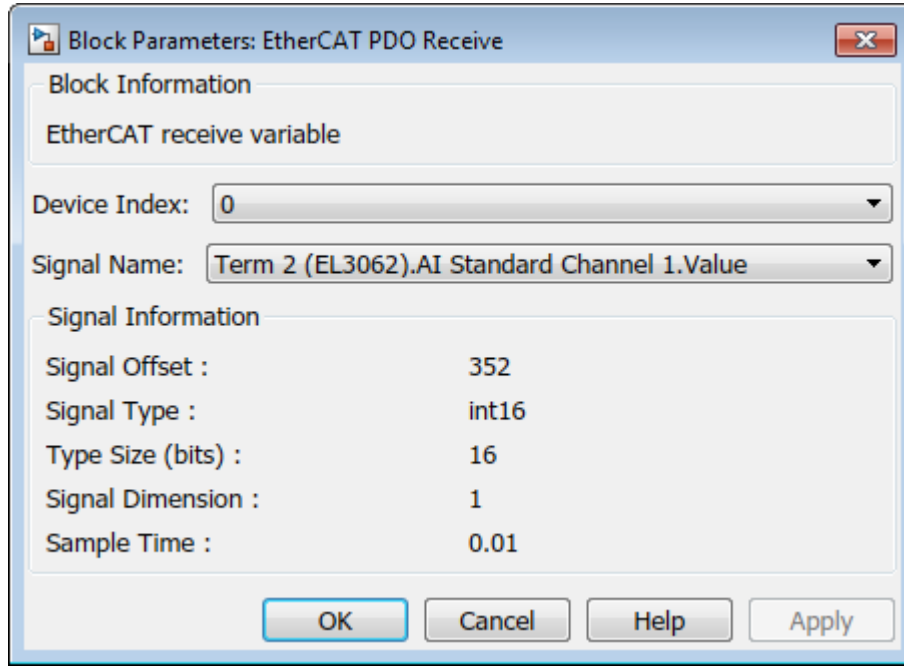
To configure the `EtherCAT PDO Receive` blocks of model `xpcEthercatBeckhoffAIO`, execute the following steps. You must have used the Beckhoff ET9000 configurator to configure the EtherCAT network.

1 Double-click the EtherCAT PDO Receive block labeled `EtherCAT PDO Receive`.

2 Set parameter **Device Index** to the value set in the `EtherCAT Init` block.

3 From the **Signal Name** list, select the EtherCAT network being accessed, here `Term 2 (EL3062).AI Standard Channel 1.Value`.

4 Note the value of parameter **Sample Time**, which is in seconds.



5 Click **OK**.

Execute steps 5–9 for the `EtherCAT PDO Receive` block labeled `EtherCAT PDO Receive 1`.

Configure EtherCAT PDO Transmit Blocks

To configure the `EtherCAT PDO Transmit` blocks of model `xpcEthercatBeckhoffAIO`, execute the following steps. You must have used the Beckhoff ET9000 configurator to configure the EtherCAT network.

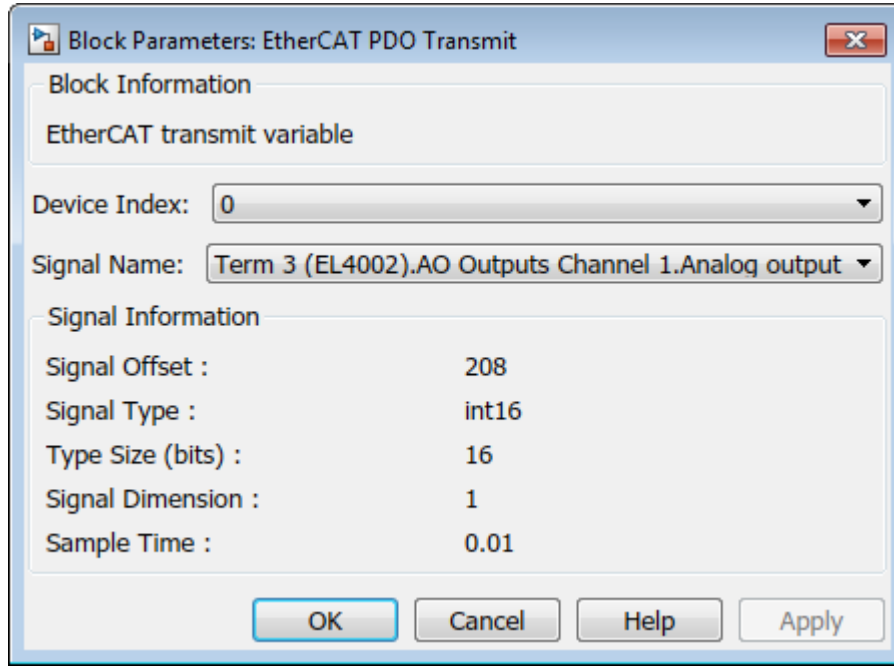
1 Open model `xpcEthercatBeckhoffAIO`.

2 Double-click the EtherCAT PDO Transmit block labeled `EtherCAT PDO Transmit`.

3 Set parameter **Device Index** to the value set in the `EtherCAT Init` block.

4 Select a **Signal Name** value consistent with the EtherCAT network being accessed, here `Term 3 (EL4002).AO Outputs Channel 1.Analog output`.

5 Note the value of parameter **Sample Time**, which is in seconds.



6 Click **OK**.

Execute steps 2–6 for the `EtherCAT PDO Transmit` block labeled `EtherCAT PDO Transmit 1`.

Configure EtherCAT Model Configuration Parameters

To configure the configuration parameters for model `xpcEthercatBeckhoffAIO`, execute the following steps. You must have used the Beckhoff ET9000 configurator to configure the EtherCAT network. For more information, see “Fixed-Step Size Derivation” on page 13-27.

1 Open model `xpcEthercatBeckhoffAIO`.

2 Calculate the greatest common divisor (GCD) of the **Sample Time** values for the EtherCAT tasks and for all source blocks in the model. In this case, the GCD is `0.010`.

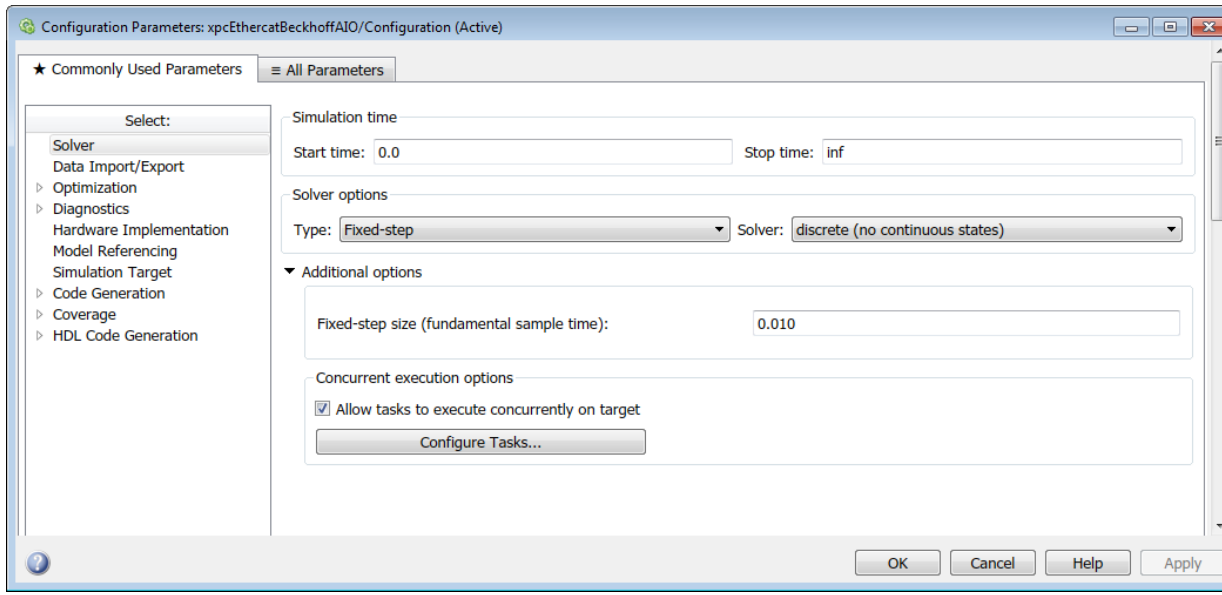
3 In the Model Editor, click **Simulation** > **Model Configuration Parameters** and click the **Solver** tab.

4 Set the **Type** parameter to `Fixed-step` and **Fixed-step size (fundamental sample time)** to one of the following:

- An integral divisor of the GCD value, in seconds.
- `auto`, if all other source blocks in the model have defined sample times.

In this case, set it to `0.010`.

The model configuration parameters dialog box looks like this figure.



5 Click **OK**.

The next tasks are building, downloading, and executing the EtherCAT master node model.

EtherCAT Distributed Clock Algorithm

In this section...

An EtherCAT network consists of a master node (the target computer) connected to an arbitrary number of slave nodes (devices). Each node contains a clock that controls its internal operation. When you enable distributed clocks, EtherCAT designates one clock in the network as the reference clock. The EtherCAT distributed clock (DC) algorithm then synchronizes the operation of multiple network nodes to the reference clock.

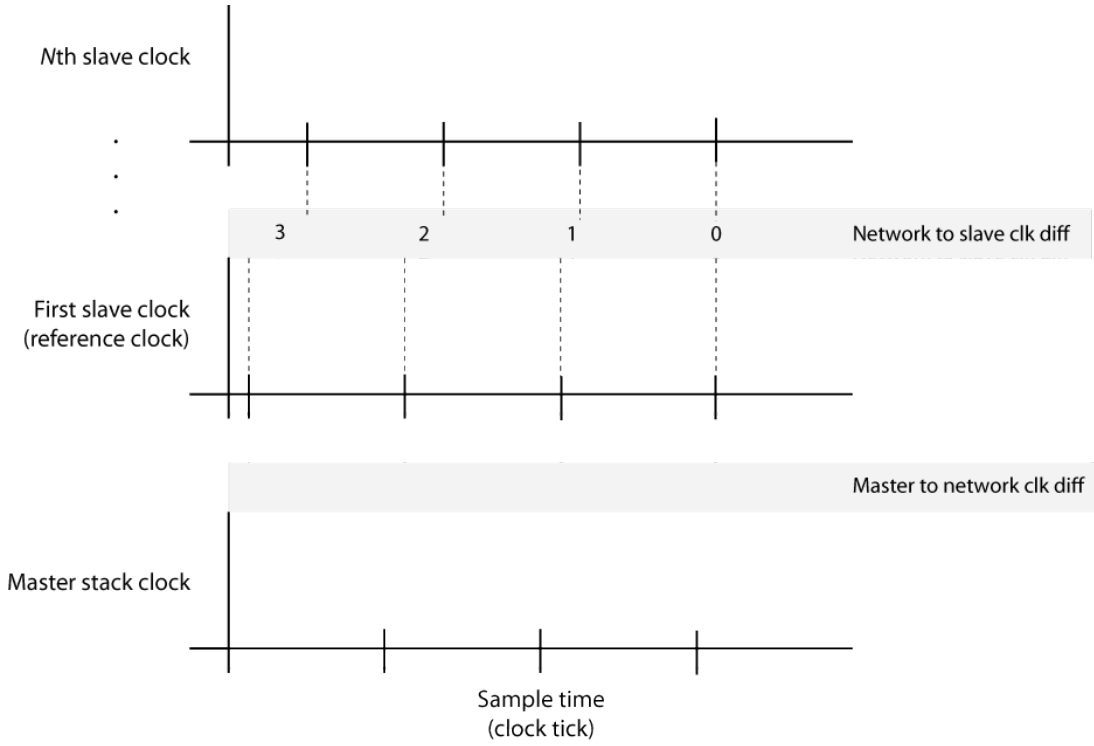
The DC algorithm operates in two phases. In phase 1, the algorithm aligns the clocks of DC-enabled network nodes other than the master node with the clock of the first DC-enabled slave node. In phase 2, the algorithm aligns the remaining unaligned clock with the reference clock.

Do not manually adjust the sample time of the real-time application in either master shift mode or bus shift mode.

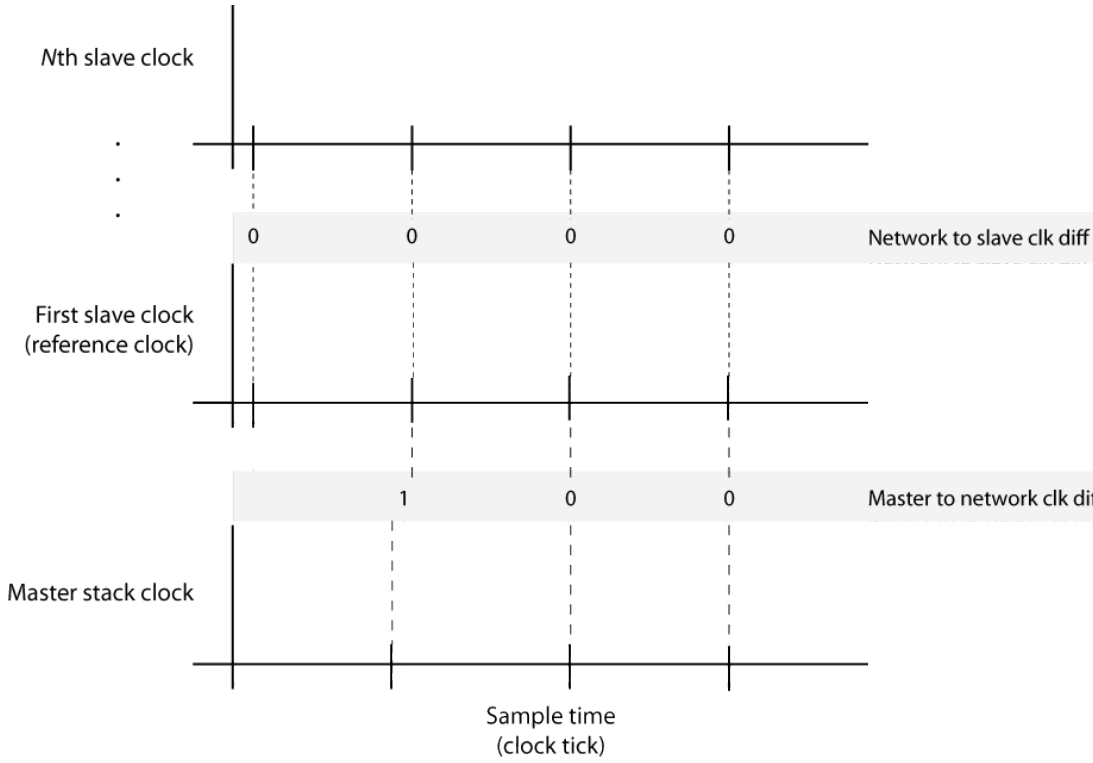
Master Shift Mode

In master shift mode, the reference clock is the clock of the first DC-enabled slave in the network.

In phase 1, the algorithm shifts the sample time of the network nodes to align with the clock of the first slave node. In that process, the EtherCAT Init block output value `NetworkToSlaveClkDiff` decreases to near zero.



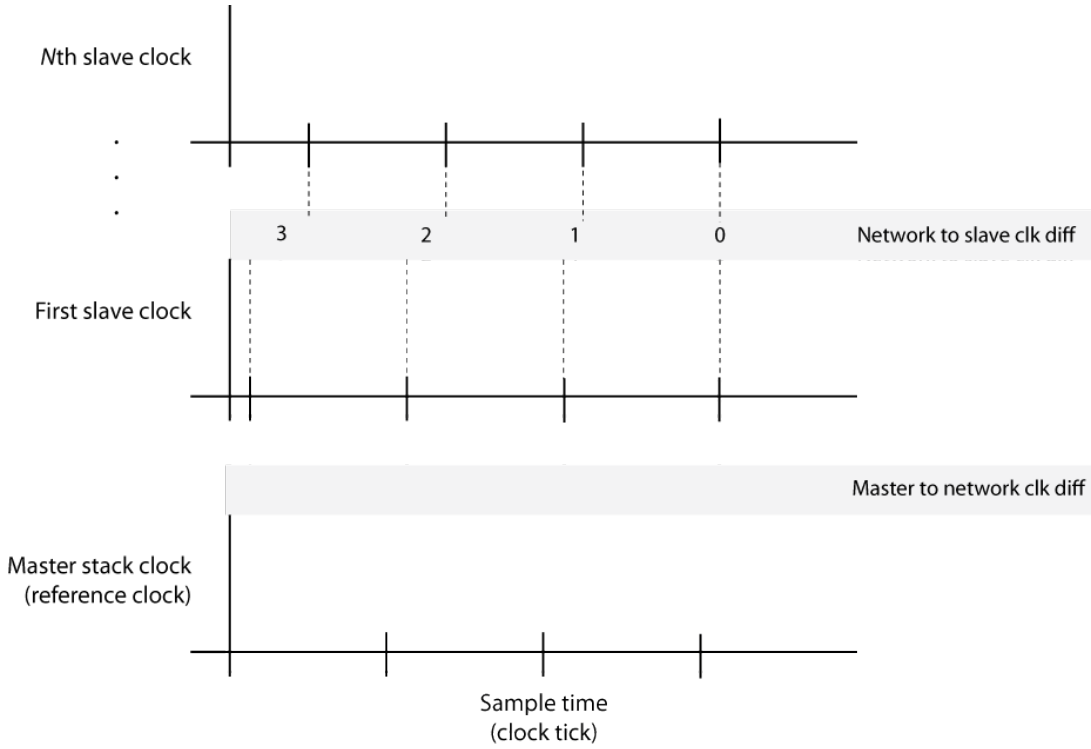
In phase 2, the algorithm shifts the sample time of the master stack running on the target computer to align with the first slave node clock. In that process, the EtherCAT Init block output value `MasterToNetworkClkDiff` decreases to near zero.



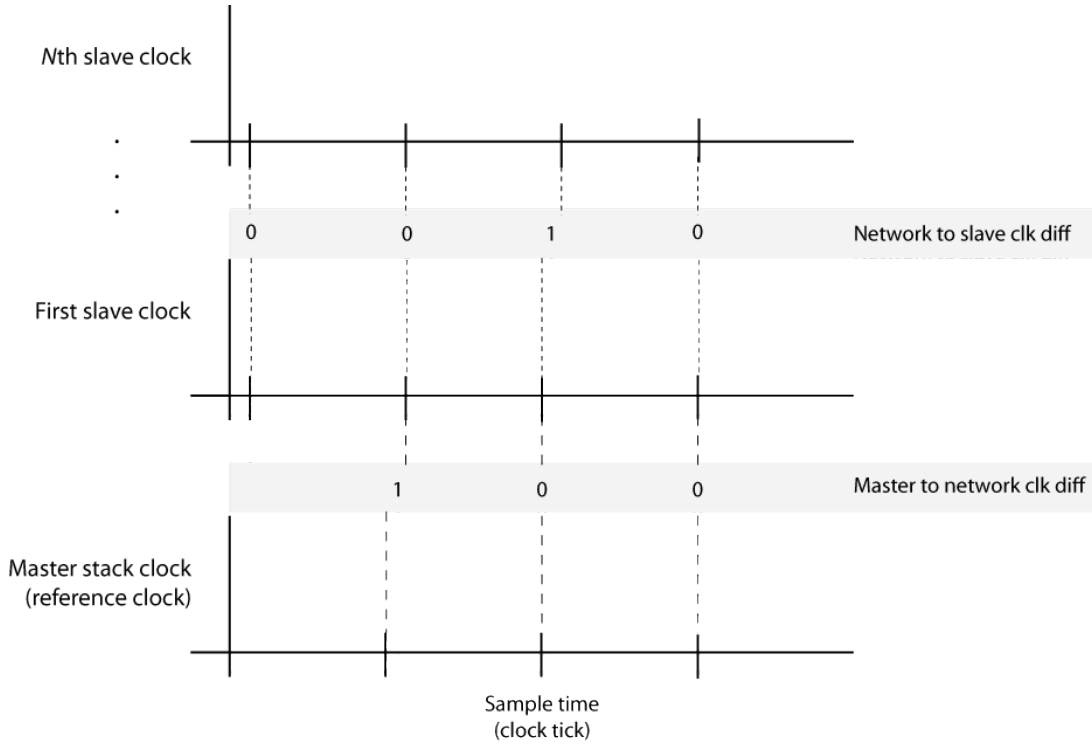
Bus Shift Mode

In bus shift mode, the reference clock is the clock of the master stack running on the target computer.

In phase 1, the algorithm shifts the sample time of the DC-enabled network nodes to align with the clock of the first DC-enabled slave node. In that process, the value `NetworkToSlaveClkDiff` decreases to near zero.



In phase 2, the algorithm shifts the sample time of the first DC-enabled slave node to align with the clock of the master stack. In that process, the value `MasterToNetworkClkDiff` decreases to near zero. The algorithm shifts the sample time of the other network nodes to stay aligned with the first slave node clock. In that process, the value of `NetworkToSlaveClkDiff` first increases, then decreases to near zero.



Limitations

If you configure EtherCAT distributed clocks in master shift mode, using the IEEE 1588 Sync Execution block in the same model produces a build error. To include EtherCAT distributed clocks and IEEE 1588 synchronized execution in the same model, use EtherCAT bus shift mode.

See Also

`EtherCAT Init`

More About

- “EtherCAT Init Block DC Error Values” on page 13-34

Fixed-Step Size Derivation

To configure the sample rate for an EtherCAT model, set the fixed-step size for the entire model and the sample rates for key blocks.

The fixed-step size determines the task rates of the EtherCAT tasks and the sample rates of the other source blocks in the model. Subject to the fixed step size value, the block type determines the sample rate groups: a comparatively slow rate for the synchronous SDO blocks, and another, faster rate for the rest of the blocks.

Using an EtherCAT network configurator, specify the EtherCAT task rate based on the requirements of the EtherCAT network. Choose the fixed-step size so that the GCD of the task rates and the block sample rates is an integer multiple of the fixed-step size.

The software sends all PDO data updates at the fastest specified task rate, even if you created multiple EtherCAT tasks running at different rates. The PDO read and write blocks run at the rate for the tasks containing the given EtherCAT variable.

If you know that the other source blocks have defined sample times, you can set **Fixed-step size** to `auto`. If one or more block rates are incompatible with the fixed sample time, there is an error during system update. If you do not encounter an error, from the Simulink **Display** menu, set **Sample Time > Colors** to reveal the block rates.

EtherCAT Protocol Mapping

EtherCAT supports several overlay protocols. Simulink Real-Time supports some of the protocols directly, provides others with minimal support, and ignores some others.

Overlay Protocol	Protocol Description	Support Type	Means of Support
CANopen over EtherCAT (CoE)	Implements CAN functionality using EtherCAT	Direct	Model CoE using SDO upload and download blocks.
Ethernet over EtherCAT (EoE)	Provides EtherCAT wrapper around Ethernet packets. EtherCAT acts as network switch	Minimal	Send wrapped EoE messages between separate slave devices.
File Access over EtherCAT (FoE)	Updates the EtherCAT board ROM	Ignored	Update the EtherCAT slave ROM using Beckhoff TwinCAT.
Functional Safety over EtherCAT (FSoE)	Sends asynchronous ‘safety’ messages over the network.	Ignored	
Servo over EtherCAT (SoE)	Wraps vendor-specific servo commands in a common protocol.	Ignored	

EtherCAT Configurator Component Mapping

The following table summarizes the mapping between third-party EtherCAT configurator components and Simulink Real-Time blocks and block attributes. For more information, see the documentation for the Beckhoff ET9000, the Beckhoff TwinCAT, or the Acontis EC-Engineer configurator.

EtherCAT Configurator Component		Simulink Real-Time Component
Beckhoff ET9000, TwinCAT	**Acontis EC-Engineer**	
Cycle ticks (task rate)	Cycle time	Sample time, Task rates
Scalars and vectors	Dimension	Dimension
BitSize	Byte size of type	Type Size
Data Type, BitSize	Data type	Signal Type
EtherCAT device variable linked to a variable in a task	All PDO variables included in default task, with no linking required.	EtherCAT PDO Receive **Signal Name**
Device variables in Process Image entity	EtherCAT PDO Receive or EtherCAT PDO Transmit block	EtherCAT PDO Receive or EtherCAT PDO Transmit block

See Also

`EtherCAT PDO Receive` | `EtherCAT PDO Transmit`

More About

- “EtherCAT Data Types” on page 13-33

Ethernet Chip Sets Compatible with EtherCAT

The Simulink Real-Time EtherCAT blocks support the Intel Gigabit Ethernet and Intel Fast Ethernet (10/100) chip sets. The Intel vendor ID is 0x8086.

Intel Gigabit Ethernet Chip Sets

Device ID	Chip Number	Chip Description
0x100E	02000	Intel PRO/1000 MT Desktop Adapter
0x1010	82546EB	Intel PRO/1000 MT Dual Port Server Adapter
0x1013	82541EI	Gigabit Ethernet Controller (Copper)
0x1019	82547EI	Gigabit Ethernet Controller (LOM)
0x1026	82545EP	Gigabit Ethernet Controller
0x104A	82566DM	Intel 82566DM Gigabit Ethernet
0x104D	82566MC	Intel 82566MC Gigabit Ethernet
0x105E	82579V	Intel PRO/1000 PT Dual Port Server Adapter
0x1075	82547EI	Gigabit Ethernet Controller
0x1076	82541EI	Gigabit Ethernet Controller
0x1078	82541ER	Gigabit Ethernet Controller
0x1079	82546EB	Dual Port Gigabit Ethernet Controller
0x107C	82541PI	Gigabit Ethernet Controller (Copper) rev 5
0x107D	82572EI	Intel 82572EI Gigabit Ethernet Controller
0x108B	PC82573V	Intel network controller (PCIE Gigabit Ethernet)
0x108C	82573E	Intel 82573E Gigabit Ethernet Controller (Copper)
0x109A	82573L	Intel PRO/1000 PL Network Adaptor
0x10A4	82571GB	Intel PRO/1000 PT Quad Port Server Adapter
0x10A7	82575EB	82575EB Gigabit Network Connection
0x10B9	82572GI	Intel PRO/1000 PT Desktop
0x10BC	82571GB	Intel PRO/1000 PT Low Profile Quad Port Server Adapter
0x10BD	82566DM	Intel 82566DM Gigabit Ethernet Adapter

Device ID	Chip Number	Chip Description
0x10C9	82576	82576 Gigabit ET Dual Port Server Adapter
0x10CE	82567V-2	Intel 82567V-2 Gigabit Network Connection
0x10D3	82574L	Intel 82574L Gigabit Ethernet Controller
0x10DE	02761028	Intel Gigabit network connection
0x10EA	82577LM	Intel 82577LM Gigabit LAN Controller
0x10EB	82577LC	Intel 82577 Gigabit Ethernet
0x10EF	82578DM	Intel 82578DM Gigabit Ethernet
0x10F0	82578DC	Intel 82578DC Gigabit Ethernet
0x10F5	82567LM	Intel 82567LM-2 Gigabit Network Connection
0x1501	82567V3	Intel 82567V3 Gigabit Network Connections
0x1502	0x04931028	Intel 82579LM Gigabit Network Card
0x1503	82579V	Gigabit Network Connection
0x150C	82583V	Intel 82583V Gigabit Ethernet Controller
0x150E	82580 QUAD	82580 Gigabit Network Connection
0x1521	I350	i350 Gigabit Network Connection
0x1526	82576	Intel Gigabit ET2 Quad Port Server Adapter
0x1527	82580 QUAD FIBRE	Intel Ethernet Server Adapter I340-F4
0x1533	210_COPPER	Intel I210 Gigabit Network Connection
0x1539	211AT	Intel Ethernet Controller I211-AT
0x153A	217LM	
0x153B	217V	
0x155A	218LM	
0x157B	210_COPPER_I	Intel I210 Gigabit Network Connection
0x1559	218V	
0xABB1		
0xABB2		

Intel Fast Ethernet (10/100) Chip Sets

Device ID	Chip Number	Chip Description
0x1039	10011734	LAN Controller: 82562ET, 82562EZ, 82562VE, 82562VM
0x103A	82801DB	LAN Controller with 82562ET/EZ (CNR)
0x1050	82562EZ	Pro/100 VE Network Connection
0x1059	82551QM	Fast Ethernet PCI/CardBus Controller
0x1092	27DA	PRO/100 VE Network Controller
0x1209	8255xER/IT	Fast Ethernet Controller for XP PC
0x1229	IE22	Intel PRO/100 M Desktop Adapter: 82557, 82558, 82559, 82550, 82551
0x2449	82559ER	82559ER Integrated 10Base-T/100Base-TX Ethernet Controller
0x27DC	336C1462	Intel PRO/100 VE Desktop Adapter

EtherCAT Data Types

The Simulink Real-Time EtherCAT blocks directly support the following EtherCAT data types. The software maps other EtherCAT data types to a byte array. The byte array requires explicit conversion using Byte Pack, Byte Unpack, or S-function blocks.

EtherCAT Data Type	Data Type Size (bits)	Converted Simulink Data Type
bit	1	uint8
bit8	8	uint8
bitarr	8 (bit array)	uint8
bitarr16	16 (bit array)	uint16
bitarr32	32 (bit array)	uint32
BOOL	1	Boolean
int8	8	int8
int16	16	int16
int32	32	int32
int64	64	int64
uint8	8	uint8
uint16	16	uint16
uint32	32	uint32
uint64	64	uint64
float	32	real32_T
double	64	real_T

EtherCAT Init Block DC Error Values

The Simulink Real-Time EtherCAT Init block returns the following EtherCAT distributed clock (DC) error values. The value `0` indicates that no error occurred.

Error Value	Description
1 (0x1)	Initialization function not called or not successful
2 (0x2)	Controller error — synchronization out of limit
3 (0x3)	Not enough memory
4 (0x4)	Hardware layer — (BSP) invalid
5 (0x5)	Hardware layer — error modifying the timer
6 (0x6)	Hardware layer — timer is not running
7 (0x7)	Hardware layer — function is called on wrong CPU
8 (0x8)	Invalid DC synchronization period length
9 (0x9)	Error DCM Controller SetVal is too small
10 (0xA)	Error DCM Controller — Drift between local timer and ref clock too high

EtherCAT Blocks

EtherCAT Init
EtherCAT Get Notifications
EtherCAT PDO Receive
EtherCAT PDO Transmit
EtherCAT Get State
EtherCAT Set State
EtherCAT Sync SDO Upload
EtherCAT Sync SDO Download
EtherCAT Async SDO Upload
EtherCAT Async SDO Download

EtherCAT Init

Initialize EtherCAT Master node with data in the EtherCAT Network Information (ENI) file

Library: EtherCAT

Description

The EtherCAT Init block initializes the EtherCAT master stack. The block specifies the Ethernet interface cards in the network.

Before you use this block, create and save an EtherCAT Network Information (ENI) file. You export the ENI file from one of these EtherCAT configurators: the Beckhoff ET9000, the Beckhoff TwinCAT, or the Acontis EC-Engineer. See “Configure EtherCAT Network” on page 13-7.

To find the ENI file, click **Browse**. To read the ENI file and store the data in the EtherCAT Init block, click **Refresh Data**.

The Simulink Real-Time software supports 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.

If you configure EtherCAT distributed clocks in master shift mode, using the IEEE 1588 Sync Execution block in the same model produces a build error. To include EtherCAT distributed clocks and IEEE 1588 synchronized execution in the same model, use EtherCAT bus shift mode.

Ports

Output

`Status` — Status information about the EtherCAT network

vector

The `Status` vector contains six values: `ErrVal`, `MasterState`, `DCErrVal`, `MasterToNetworkClkDiff`, `DCInitState`, and `NetworkToSlaveClkDiff`.

- `ErrVal` — Error status:
  - No error: `0`
  - Error: Value less than `0`.

  Because `ErrVal` shows the latest error status, the propagation of errors can hide the original error. To find the original error, add an `EtherCAT Get Notifications` block and use `SimulinkRealTime.etherCAT.filterNotifications` to print the status codes that the EtherCAT stack transmits.
- `MasterState` — Operating state of the EtherCAT network:

State	Value	Description
`INIT`	1	Initialization – The system finds slave devices and initializes the communication controller.
`PREOP`	2	Preoperational — The system uses the communication controller to exchange system-specific initialization data. In this state, the network cannot transmit or receive signal data.
`SAFEOP`	4	Safe operational — The network is running and ready for full operation. The master sends input data to the slave device. The slave device output remains in a safe state.
`OP`	8	Operational — The network is in full operation. The master sends input data to the slave device. The slave device responds with output data.

- `DCErrVal` — DC error status:
  - No DC Error: `0`

  - DC error: Value from "EtherCAT Init Block DC Error Values" on page 13-34.

  The value `0` appears both if the distributed clock is turned off and if no error occurs.

- `MasterToNetworkClkDiff` — Time difference, in nanoseconds, between the master stack clock and the clock on the first slave device that has enabled DC.
- `DCInitState` — Operating state of the distributed clock:
  - DC not enabled or not initialized: `0`
  - DC has been started: `1`
- `NetworkToSlaveClkDiff` — Time difference, in nanoseconds, between the clock on the first EtherCAT slave device and the least closely locked clock on the remaining slave devices.

  This value applies only to slave devices that have enabled DC. If only one device on the network has enabled DC, this value is `0`.

Data Types: `int32`

Parameters

`Config file (ENI)` — ENI file from the EtherCAT configurator

character vector

Specify the ENI file that you exported from the EtherCAT configurator.

You can specify the full path name or a partial path name. If you specify only the file name, the software searches for the file in the current folder and on the MATLAB path. If more than one file with that name exists on the path, MATLAB displays a message box where you select the file that you want.

Clicking **Browse** inserts a full, editable path name.

`Device index` — EtherCAT Ethernet card identifier

`0-15`

A unique integer in the range `0`–`15` that identifies the Ethernet card for an EtherCAT network.

For each EtherCAT network, the software generates a unique device index. The software inserts that device index as **Device index** into the EtherCAT Init block that represents the network.

`PCI bus` — PCI bus number of Ethernet card
`0` (default) | integer

Enter the PCI bus number for the Ethernet card.

`PCI slot` — PCI slot number of Ethernet card
`0` (default) | integer

Enter the PCI slot number for the Ethernet card.

`DC Tuning` — Distributed clock initialization parameter
`Large model` (default) | `Medium model` | `Small model`

Enter the distributed clock initialization parameter, one of these values:

- `Large model` (default) — Sends `16,000` timing initialization packets and allows `1` second of settling time. Provides best initial synchronization between multiple slaves that have DC enabled.
- `Medium model` — Sends `8,000` timing initialization packets and allows `0.3` seconds of settling time. The model reaches operational state about a second earlier than it does with the `Large model` setting.
- `Small model` — Sends `2,000` timing initialization packets and allows `0.2` seconds of settling time. The model reaches operational state earlier than it does with the other settings.

Monitor device synchronization at the moment that the model enters the operational state. Check that the devices are synchronized closely enough for your application.

`Enable Log and Debugging` — Access to debugging and logging block parameters
`off` (default) | `on`

Dependency

Selecting **Enable Log and Debugging** makes these parameters visible: **Log link layer error messages**, **Log master state changes**, **Log all state changes**, **Log base clock changes**, **Log master config changes**, and **Target log filename**.

`Log link layer error messages`, `Log master state changes`, `Log all state changes`, `Log base clock changes`, `Log master config changes` — Generate driver-level debug messages
`off` (default) | `on`

To generate driver-level messages for driver and network debugging, select these check boxes.

For a high-speed model, turning on these options can cause CPU overloads.

Dependency

To make these parameters visible, select **Enable Log and Debugging**.

`Target log filename` — Name of log file on target computer
character vector

Enter the name of the log file on the target computer, in single quotes. The default value is `'c:\dbglog.txt'`.

If the target computer does not have a usable disk partition, the software does not create the log file.

Dependency

To make these parameters visible, select **Enable Log and Debugging**.

See Also

See Also

"EtherCAT Init Block DC Error Values" on page 13-34 | `EtherCAT Get Notifications` | `SimulinkRealTime.etherCAT.filterNotifications`

Topics

"EtherCAT® Communication with Beckhoff® Analog IO Slave Devices EL3062 and EL4002"

"EtherCAT® Communication with Beckhoff® Digital IO Slave Devices EL1004 and EL2004"
"EtherCAT® Communication - Motor Velocity Control with Accelnet™ Drive"
"EtherCAT® Communication - Motor Position Control with an Accelnet™ Drive and Beckhoff® Analog IO Devices"
"Configure EtherCAT Network" on page 13-7
"Configure EtherCAT Master Node Model" on page 13-16

External Websites

www.ethercat.org
www.beckhoff.com
www.acontis.com/eng

Introduced in R2010b

EtherCAT Get Notifications

Collect notifications from the EtherCAT bus
Library: EtherCAT

Description

Collects notifications from the EtherCAT stack and presents them to the output as a 21-element vector of `int32`. At each time step, the block outputs what it has accumulated and clears itself for the next time step.

The vector contains the number of notifications in element `1`, followed by up to `20` notification codes. The maximum number of notifications is `20`. If the bus presents more than `20` notifications to the output, the block discards the newest notifications and presents the first `20` that were received.

Ports

Output

`Values` — Self-descriptive 21-element vector containing EtherCAT notification codes
`[Length 20 * Notification]`

- `Length` (`0` – `20`) — the number of notifications in the vector.
- `Notification` — a composite of a notification type and a specific value. The types are:
- - `EC_NOTIFY_GENERIC` [`0x00000000` (`0`)] — Represents state changes, such as:`0x00000001` (`1`) — `EtherCAT operational state change`.
  - `EC_NOTIFY_ERROR` [`0x00010000` (`65536`)] — Represents error states, such as `0x00010001` (`65537`):`cyclic command: working counter error`. Some describe changes in error state.

- `EC_NOTIFY_SCANBUS` [`0x00030000 (3*65536)`] — Represents ScanBus error states, such as `0x00030002 (196610):ScanBus mismatch`.
- `EC_NOTIFY_HOTCONNECT` [`0x00040000 (4*65536)`] — Represents hot connect states, such as `0x00040005 (262149):Slave disappears`.

To print the legal notification values and descriptions, call `SimulinkRealTime.etherCAT.filterNotifications` without an argument.

Data Types: `int32`

Parameters

`Device index` — EtherCAT Ethernet card identifier

`0` (default) | `0-15`

To associate a block with an EtherCAT network, copy the **Device index** value from the EtherCAT Init block representing that network into the **Device index** for the block.

`Sample time` — Sample time of block

`-1` (default) | numeric

Enter the base sample time or a multiple of the base sample time. Use the EtherCAT task sample time.

Tips

To collect notifications:

1. Add the EtherCAT Get Notifications block to your model.
2. Connect the EtherCAT Get Notifications block to an Outport block. If possible, make this Outport block Outport block `1`. If the EtherCAT Get Notifications block is connected to the first Outport block, the 21 notification signals appear in the first 21 columns `tg.OutputLog` matrix. Otherwise, you must select the columns with an offset.
3. Increase the value of **Signal logging data buffer size in doubles** by at least a factor of 100 in the **Simulink Real-Time Options** pane. The EtherCAT Get Notifications block can quickly increase the size of the output log.

4 To print the notifications for this model, pass the relevant 21 columns into the `SimulinkRealTime.etherCAT.filterNotifications` function.

See Also

See Also

`EtherCAT Init` | `SimulinkRealTime.etherCAT.filterNotifications`

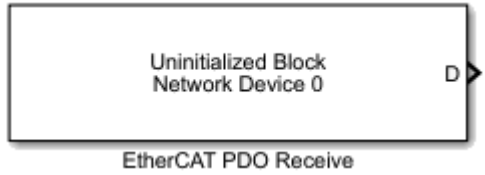
External Websites

www.ethercat.org
www.beckhoff.com
www.acontis.com/eng

Introduced in R2017a

EtherCAT PDO Receive

Receive data from slave device represented by process data object



Library

Simulink Real-Time Library for EtherCAT

Description

The EtherCAT PDO Receive 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 PDO Receive block mask is updated with the selected signal name.
- The signal information in the block parameter dialog box is updated to reflect the device variable.

Note: If an error occurs while the software parses the configuration file specified in the EtherCAT Init block, this block shows an error message.

Block Outputs

Name	Description
D	Data received from the EtherCAT slave device.

Block Parameters

Device index

A unique integer in the range 0 to 15 that identifies the Ethernet card for an EtherCAT network.

To associate a block with an EtherCAT network, copy the **Device index** value from the EtherCAT Init block representing that network into the **Device index** for the block.

The default value of **Device index** is 0.

Signal name

From the list, select the EtherCAT device variable name.

The block parameter dialog box updates the signal information to reflect the device variable that you selected.

Signal Information	Description
`Signal Offset`	Location in the process image from which the data is available after the execution of the `EtherCAT Init` block. This value is the EtherCAT configurator BitOffs − 80.
`Signal Type`	The Simulink data type for the EtherCAT data. For a mapping of Simulink data types to EtherCAT data types, see “EtherCAT Data Types” on page 13-33
`Type Size (bits)`	Size of the EtherCAT data type. See “EtherCAT Data Types” on page 13-33 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 rate is the execution rate of the EtherCAT task, as specified in the Beckhoff ET9000 EtherCAT configurator.

See Also

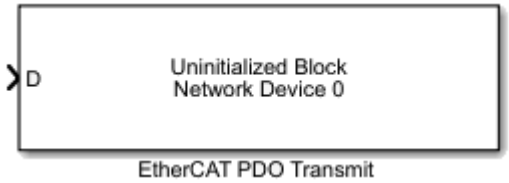
External Websites

www.ethercat.org
www.beckhoff.com
www.acontis.com/eng

Introduced in R2010b

EtherCAT PDO Transmit

Send data to slave device represented by process data object



Library

Simulink Real-Time Library for EtherCAT

Description

The EtherCAT PDO Transmit block transmits computed data to a particular variable in 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 PDO Receive block mask is updated with the selected signal name.
- The signal information in the block parameter dialog box is updated to reflect the device variable.

Note: If an error occurs while the software parses the configuration file specified in the EtherCAT Init block, this block shows an error message.

Block Inputs

Name	Description
D	Data to transmit to the EtherCAT slave device.

Block Parameters

Device index

A unique integer in the range `0` to `15` that identifies the Ethernet card for an EtherCAT network.

To associate a block with an EtherCAT network, copy the **Device index** value from the EtherCAT Init block representing that network into the **Device index** for the block.

The default value of **Device index** is `0`.

Signal name

From the list, select the EtherCAT device variable name.

The block parameter dialog box updates the signal information to reflect the device variable that you selected.

Signal Information	Description
`Signal Offset`	Location in the process image from which the data is available after the execution of the `EtherCAT Init` block. This value is the EtherCAT configurator BitOffs − 80.
`Signal Type`	The Simulink data type for the EtherCAT data. For a mapping of Simulink data types to EtherCAT data types, see “EtherCAT Data Types” on page 13-33
`Type Size (bits)`	Size of the EtherCAT data type. See “EtherCAT Data Types” on page 13-33 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 rate is the execution rate of the EtherCAT task, as specified in the Beckhoff ET9000 EtherCAT configurator.

See Also

External Websites

www.ethercat.org
www.beckhoff.com
www.acontis.com/eng

Introduced in R2010b

EtherCAT Get State

Get state of EtherCAT network

Library

Simulink Real-Time Library for EtherCAT

Description

The EtherCAT Get State block returns the state of the EtherCAT network.

State	Value	Description
`INIT`	1	Initialization – The system finds slave devices and initializes the communication controller.
`PREOP`	2	Preoperational — The system uses the communication controller to exchange system-specific initialization data. In this state, the network cannot transmit or receive signal data.
`SAFEOP`	4	Safe operational — The network is running and ready for full operation. The master sends input data to the slave device. The slave device output remains in a safe state.
`OP`	8	Operational — The network is in full operation. The master sends input data to the slave device. The slave device responds with output data.

Block Outputs

Name	Description
`State`	Reception from the EtherCAT network.

Block Parameters

Device index

A unique integer in the range 0 to 15 that identifies the Ethernet card for an EtherCAT network.

To associate a block with an EtherCAT network, copy the **Device index** value from the EtherCAT Init block representing that network into the **Device index** for the block.

The default value of **Device index** is 0.

Sample time

Enter the base sample time or a multiple of the base sample time (-1 means sample time is inherited).

See Also

External Websites

www.ethercat.org
www.beckhoff.com
www.acontis.com/eng

Introduced in R2010b

EtherCAT Set State

Set state of EtherCAT network

Library

Simulink Real-Time Library for EtherCAT

Description

The EtherCAT Set State block sets the state of the EtherCAT network. Each state has the corresponding integer:

State	Value	Description
`INIT`	1	Initialization – The system finds slave devices and initializes the communication controller.
`PREOP`	2	Preoperational — The system uses the communication controller to exchange system-specific initialization data. In this state, the network cannot transmit or receive signal data.
`SAFEOP`	4	Safe operational — The network is running and ready for full operation. The master sends input data to the slave device. The slave device output remains in a safe state.
`OP`	8	Operational — The network is in full operation. The master sends input data to the slave device. The slave device responds with output data.

Block Inputs and Outputs

Inputs

Name	Description
`New State`	Transmission to the EtherCAT network.

Outputs

Name	Description
`Prev State`	Previous state of the network.
`Error`	`0` if no error occurs. Otherwise, a nonzero value.

Block Parameters

Device index

A unique integer in the range `0` to `15` that identifies the Ethernet card for an EtherCAT network.

To associate a block with an EtherCAT network, copy the **Device index** value from the EtherCAT Init block representing that network into the **Device index** for the block.

The default value of **Device index** is `0`.

Timeout

Enter the number of seconds to wait for the EtherCAT network state to transition.

Set the timeout to `0` to return immediately.

Sample time

Enter the base sample time or a multiple of the base sample time (`-1` means sample time is inherited).

See Also

External Websites

www.ethercat.org
www.beckhoff.com
www.acontis.com/eng

Introduced in R2010b

EtherCAT Sync SDO Upload

Read data synchronously from slave device represented by service data object

Library

Simulink Real-Time Library for EtherCAT

Description

The EtherCAT Sync SDO Upload block reads a CANopen dictionary entry in the specified EtherCAT slave. The block then waits until it receives a response or until the timeout period is over.

The response to an operation can take several ticks of the main task sample time. Assign the synchronous blocks a sample time slower than the main task sample time.

Block Outputs

Name	Description
`Data`	Data received from the EtherCAT slave device.
`Error`	`0` if no error occurs. Otherwise, a nonzero value.

Block Parameters

Index

Specify the decimal index of the CANopen dictionary entry.

If you specify an invalid index, the block does not return an error or a timeout. The results are undefined.

Subindex

Specify the decimal subindex of the CANopen dictionary entry.

If you specify an invalid subindex, the block does not return an error or a timeout. The results are undefined.

Data Type

From the list, select the data type of the CANopen dictionary entry.

If you select a data type that does not match the type of the entry, the block returns an error.

Dimension

Specify the row and column dimension of the CANopen dictionary entry.

Enter a value of `1`. EtherCAT blocks support only scalars and vectors.

Device index

A unique integer in the range `0` to `15` that identifies the Ethernet card for an EtherCAT network.

To associate a block with an EtherCAT network, copy the **Device index** value from the EtherCAT Init block representing that network into the **Device index** for the block.

The default value of **Device index** is `0`.

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 the base sample time or a multiple of the base sample time (`-1` means sample time is inherited).

Timeout

Enter the number of milliseconds to wait for a response from the EtherCAT slave.

See Also

External Websites

www.ethercat.org
www.beckhoff.com
www.acontis.com/eng

Introduced in R2010b

EtherCAT Sync SDO Download

Transmit data synchronously to slave device represented by service data object

Library

Simulink Real-Time Library for EtherCAT

Description

The EtherCAT Sync SDO Download block writes to a CANopen dictionary entry in the specified EtherCAT slave. The block then waits until it receives a response or until the timeout period is over.

The response to an operation can take several ticks of the main task sample time. Assign the synchronous blocks a sample time slower than the main task sample time.

Block Inputs and Outputs

Inputs

Name	Description
`Data`	Input data for writing to the EtherCAT slave device.

Outputs

Name	Description
`Error`	`0` if no error occurs. Otherwise, a nonzero value.

Block Parameters

Index

Specify the decimal index of the CANopen dictionary entry.

If you specify an invalid index, the block does not return an error or a timeout. The results are undefined.

Subindex

Specify the decimal subindex of the CANopen dictionary entry.

If you specify an invalid subindex, the block does not return an error or a timeout. The results are undefined.

Data Type

From the list, select the data type of the CANopen dictionary entry.

If you select a data type that does not match the type of the entry, the block returns an error.

Dimension

Specify the row and column dimension of the CANopen dictionary entry.

Enter a value of 1. EtherCAT blocks support only scalars and vectors.

Device index

A unique integer in the range 0 to 15 that identifies the Ethernet card for an EtherCAT network.

To associate a block with an EtherCAT network, copy the **Device index** value from the EtherCAT Init block representing that network into the **Device index** for the block.

The default value of **Device index** is 0.

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 the base sample time or a multiple of the base sample time (`-1` means sample time is inherited).

Timeout

Enter the number of milliseconds to wait for a response from the EtherCAT slave.

See Also

External Websites

www.ethercat.org
www.beckhoff.com
www.acontis.com/eng

Introduced in R2010b

EtherCAT Async SDO Upload

Read data asynchronously from slave device represented by service data object

Library

Simulink Real-Time Library for EtherCAT

Description

The EtherCAT Async SDO Upload block requests a CANopen dictionary entry from the specified EtherCAT slave. It then immediately returns whatever value was returned from the device on an earlier call to the block.

Block Inputs and Outputs

Inputs

Name	Description
`Enable`	When `true`, the block uploads data.

Outputs

Name	Description
`Data`	Data received from the EtherCAT slave device.
`Status`	One of the following values: • `0` — Mailbox transfer object idle, transfer not running • `1` — Mailbox transfer object running, transfer not complete

Name	Description
	• 2 — Transfer successfully executed • 3 — Error occurred during transfer request

Block Parameters

Index

Specify the decimal index of the CANopen dictionary entry.

If you specify an invalid index, the block does not return an error or a timeout. The results are undefined.

Subindex

Specify the decimal subindex of the CANopen dictionary entry.

If you specify an invalid subindex, the block does not return an error or a timeout. The results are undefined.

Data Type

From the list, select the data type of the CANopen dictionary entry.

If you select a data type that does not match the type of the entry, the block returns an error.

Dimension

Specify the row and column dimension of the CANopen dictionary entry.

Enter a value of 1. EtherCAT blocks support only scalars and vectors.

Device index

A unique integer in the range 0 to 15 that identifies the Ethernet card for an EtherCAT network.

To associate a block with an EtherCAT network, copy the **Device index** value from the EtherCAT Init block representing that network into the **Device index** for the block.

The default value of **Device index** is 0.

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 the base sample time or a multiple of the base sample time (`-1` means sample time is inherited).

See Also

External Websites

www.ethercat.org
www.beckhoff.com
www.acontis.com/eng

Introduced in R2010b

EtherCAT Async SDO Download

Transmit data asynchronously to slave device represented by service data object

Library

Simulink Real-Time Library for EtherCAT

Description

The EtherCAT Async SDO Download block writes a CANopen dictionary entry in the specified EtherCAT slave. The block then immediately continues processing its input data.

Block Inputs and Outputs

Inputs

Name	Description
`Data`	Input data for writing to the EtherCAT slave device.
`Enable`	When `true`, the block downloads data.

Outputs

Name	Description
`Status`	One of the following values: • `0` — Mailbox transfer object idle, transfer not running • `1` — Mailbox transfer object running, transfer not complete

Name	Description
	• 2 — Transfer successfully executed • 3 — Error occurred during transfer request

Block Parameters

Index

Specify the decimal index of the CANopen dictionary entry.

If you specify an invalid index, the block does not return an error or a timeout. The results are undefined.

Subindex

Specify the decimal subindex of the CANopen dictionary entry.

If you specify an invalid subindex, the block does not return an error or a timeout. The results are undefined.

Data Type

From the list, select the data type of the CANopen dictionary entry.

If you select a data type that does not match the type of the entry, the block returns an error.

Dimension

Specify the row and column dimension of the CANopen dictionary entry.

Enter a value of 1. EtherCAT blocks support only scalars and vectors.

Device index

A unique integer in the range 0 to 15 that identifies the Ethernet card for an EtherCAT network.

To associate a block with an EtherCAT network, copy the **Device index** value from the EtherCAT Init block representing that network into the **Device index** for the block.

The default value of **Device index** is 0.

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 the base sample time or a multiple of the base sample time (`-1` means sample time is inherited).

See Also

External Websites

www.ethercat.org
www.beckhoff.com
www.acontis.com/eng

Introduced in R2010b

TCP, UDP

15

Real-Time TCP Communication Support

TCP Transport Protocol

The Simulink Real-Time software supports communication from the target computer to other systems or devices using Transmission Control Protocol (TCP). TCP provides ordered and error-checked packet transport.

TCP is a transport protocol layered on top of the Internet Protocol (IP). It is commonly known as TCP/IP.

- Stream — TCP is a *stream-oriented* protocol.

  TCP is a long stream of data that flows from one end of the network to the other. Another long stream of data flows in the other direction. The TCP stack at the transmitting end is responsible for breaking the stream of data into packets and sending those packets. The stack at the receiving end is responsible for reassembling the packets into a data stream using information in the packet headers.
- Connection — TCP is a *connection-based* protocol.

  In TCP, the two ends of the communication link must be connected throughout the communication.
- Error Detection — TCP detects errors.

  TCP packets contain a unique sequence number. The starting sequence number is communicated to the other side at the beginning of communication. The receiver acknowledges each packet, and the acknowledgment contains the sequence number so that the sender knows which packet was acknowledged. Therefore, packets lost on the way can be retransmitted. The sender knows that they did not reach their destination because the sender did not receive an acknowledgment. The receiver can reassemble in order packets that arrive out of sequence. Timeouts can be established, because the sender knows from the first few packets how long it takes to transmit a packet and receive its acknowledgment.

TCP communication is like a telephone conversation. A continuous connection is required, and two-way streaming data (the words spoken by each party) are exchanged.

When describing TCP, the words *Reliable* and *Unreliable* have a specific meaning.

Note: *Reliable* means that if a packet is not acknowledged, it is retransmitted. It does not mean that the protocol always succeeds.

Unreliable means that if too many packets are not acknowledged, the protocol can time out. It does not mean that the protocol packets usually fail to arrive.

You can construct a packet from Simulink data types such as `double`, `int8`, `int32`, `uint8`, or a combination of these data types. The Simulink Real-Time 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 preceding discussion applies to both communication with a shared Ethernet board and communication with a dedicated Ethernet board. Consider adding a dedicated Ethernet board for enhanced performance over communication using a shared Ethernet board. Shared TCP communication shares bandwidth with the link between the development and target computers.

See Also

`Byte Unpacking` | `Byte Packing` | `Byte Reversal/Change Endianess` | `TCP Receive` | `TCP Client Configure` | `TCP Send` | `TCP Server Configure`

External Websites

- www.ietf.org/rfc/rfc793.txt

TCP Troubleshooting

In this section...

TCP Blocks Run Only on Target Computer

The Simulink Real-Time TCP blocks function only when executed on the target computer. When simulated on the development computer, they do nothing.

Duplicate Subnet Calculated in Block

You can use a dedicated Ethernet card for TCP communication while using another card for communicating between the development and target computers. You can get the following error during model initialization:

```
The subnet in this block is the same as or is a subset of the subnet
calculated in ''block''. The block calculates the
subnet by ANDing the IP address bitwise with the subnet mask.
```

Check the IP address and subnet you assigned to the target computer Ethernet card in the configuration block. The TCP implementation requires that the two communication channels use separate subnets.

The block calculates the subnet by ANDing the IP address bitwise with the subnet mask for each card. For example, the following specifications result in the same subnet for both cards.

```
E1 (development-target): IP address:        192.168.0.25
                         Subnet mask:       255.255.255.0
                                            -------------
                         Calculated Subnet: 192.168.0.0

E2 (TCP):                IP address:        192.168.0.26
                         Subnet mask:       255.255.255.0
                                            -------------
```

```
                          Calculated Subnet: 192.168.0.0
```

Try a configuration such as the following:

```
E1 (development-target): IP address:        192.168.0.25
                         Subnet mask:       255.255.255.0
                                            -------------
                         Calculated Subnet: 192.168.0.0

E2 (TCP):                IP address:        192.168.0.26
                         Subnet mask:       255.255.255.2
                                            -------------
                         Calculated Subnet: 192.168.0.2
```

In some networks, the development computer must also be in the subnet where the TCP communication occurs. You can either add a second network card to the development computer or provide a gateway device to create a dedicated network for TCP communication.

Excluded Ports When Using Host-Target Connection

When you select the **Use host-target connection** parameter in the TCP configure blocks, you cannot use ports `22222` and `22223`. Simulink Real-Time reserves these ports for its own use.

Order of Operation of TCP Blocks

The real-time application must execute the TCP configure blocks before it executes the TCP Send or TCP Receive blocks.

As a best practice, connect the `Status` output of a TCP configure block to the `Enable` input of the associated TCP Send and TCP Receive blocks.

More About

- “TCP/IP and UDP Interface” (Instrument Control Toolbox)
- “TCP/IP Communication” (MATLAB)

16

TCP Blocks

IP Config
TCP Client
TCP Client Configure
TCP Receive
TCP Send
TCP Server
TCP Server Configure

IP Config

Initialize Ethernet network interface to use for IP communication in real-time applications
Library: TCP

Description

The IP Config block configures a dedicated Ethernet network for real-time operation.

The combination of **Local IP Address** and **Subnet mask** must be unique across all Ethernet cards in the target computer, including the card for communicating between the development and target computers. Distinguish cards by specifying a different subnet for each. The subnet is the IP address masked by the subnet mask.

Parameters

`Local IP Address` — IP address for the Ethernet interface
x.x.x.x

Enter the IP address for the dedicated Ethernet board.

The addresses `0.0.0.0` and `255.255.255.255` are invalid IP addresses.

`Subnet mask` — Subnet mask for interface
`255.255.255.0` (default) | *x.x.x.x*

Mask that designates a logical subdivision of a network.

`Gateway` — IP address for gateway interface
`0.0.0.0` (default) | *x.x.x.x*

The gateway must be within the network.

To indicate "no gateway used," enter `0.0.0.0` (the default). The address `255.255.255.255` is an invalid gateway IP address.

`PCI bus` — PCI bus number of Ethernet card
0 (default) | 0–31

Enter the PCI bus number for the Ethernet card.

`PCI slot` — PCI slot number of Ethernet card
0 (default) | 0–31

Enter the PCI slot number for the Ethernet card.

See Also

See Also
`SimulinkRealTime.target.getPCIInfo`

Topics
"TCP Troubleshooting" on page 15-4
"Real-Time UDP Troubleshooting" on page 17-15

Introduced in R2017a

TCP Client

Configure TCP client
Library: TCP

Description

Configure a TCP client application. You must have already configured a network interface for IP by the IP Config block.

Ports

Input

`Enable` — Connect block to remote Ethernet device

integer

If `Enable` is greater than zero, the block connects to the Ethernet device. Otherwise, the block does not connect.

Output

`Status` — Device returns a status of not connected or connected

`0 | 1`

The status value is one of:

- `0` — Not connected
- `1` — Connected

As a best practice, connect the `Status` output of a TCP configure block to the `Enable` input of the associated TCP Send and TCP Receive blocks.

Parameters

`Client IP address` — IP address of the client device that is being configured

`x.x.x.x`

If you are using the Ethernet connection between the development and target computers, this value must match the value of the `TcpIpTargetAddress` target setting. If you are using a dedicated Ethernet card, this value must match the **Local IP Address** parameter in the IP Config block for the network interface.

The addresses `0.0.0.0` and `255.255.255.255` are invalid IP addresses.

`Client local port` — IP port of the client device that is being configured
1–65535

The combination of **Client IP address** and **Client local port** must be unique.

Dependency

When you select the **Use host-target connection** parameter in the TCP configure blocks, you cannot use ports `22222` and `22223`. Simulink Real-Time reserves these ports for its own use.

`Remote server IP address` — IP address of the server device
`x.x.x.x`

Enter the IP address of the server to which you want to connect the client.

The addresses `0.0.0.0` and `255.255.255.255` are invalid IP addresses.

`Remote server port` — Port number of the server device
1–65535

Enter the port number of the server to which you want to connect the client.

Dependency

When you select the **Use host-target connection** parameter in the TCP configure blocks, you cannot use ports `22222` and `22223`. Simulink Real-Time reserves these ports for its own use.

See Also

See Also

`IP Config` | Target Settings Properties

Topics

"TCP Troubleshooting" on page 15-4

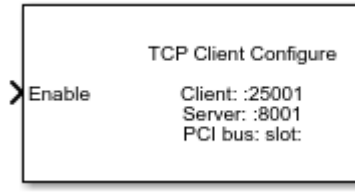
External Websites

www.ietf.org/rfc/rfc793.txt

Introduced in R2017a

TCP Client Configure

Configure a TCP client application that uses the specified Ethernet interface
Library: TCP



Description

Configure a TCP client application and initialize a network interface for the application.

The combination of **Client IP Address** and **Subnet mask** must be unique across all Ethernet cards in the target computer, including the card for communicating between the development and target computers. Distinguish cards by specifying a different subnet for each. The subnet is the IP address masked by the subnet mask.

The Simulink Real-Time TCP blocks function only when executed on the target computer. When simulated on the development computer, they do nothing.

Ports

Input

`Enable` — Connect block to remote Ethernet device

integer

If `Enable` is greater than zero, the block connects to the Ethernet device. Otherwise, the block does not connect.

Output

`Status` — Device returns a status of not connected or connected

0 | 1

The status value is one of:

- `0` — Not connected
- `1` — Connected

As a best practice, connect the `Status` output of a TCP configure block to the `Enable` input of the associated TCP Send and TCP Receive blocks.

Parameters

`Use host-target connection` — Use Ethernet connection between development and target computers
`'off'` (default) | `'on'`

Dependency

When you select this parameter, it deactivates the **Client IP address**, **Subnet mask**, **Gateway**, **PCI bus**, and **PCI slot** parameters and excludes the ports `22222` and `22223` from use by TCP.

`Client IP address` — IP address of the client device that is being configured
`x.x.x.x`

The addresses `0.0.0.0` and `255.255.255.255` are invalid IP addresses.

Dependency

To activate this parameter, clear the **Use host-target connection** parameter.

`Client local port` — IP port of the client device that is being configured
`1–65535`

The combination of **Client IP address** and **Client local port** must be unique.

Dependency

When you select the **Use host-target connection** parameter in the TCP configure blocks, you cannot use ports `22222` and `22223`. Simulink Real-Time reserves these ports for its own use.

`Subnet mask` — Subnet mask for interface

`255.255.255.0` (default) | *`x.x.x.x`*

Mask that designates a logical subdivision of a network.

Dependency

To activate this parameter, clear the **Use host-target connection** parameter.

`Gateway` — IP address for gateway interface

`0.0.0.0` (default) | *`x.x.x.x`*

Gateway to access a different subnet. The gateway must be within the network.

To indicate "no gateway used," enter `0.0.0.0` (the default). The address `255.255.255.255` is an invalid gateway IP address.

Dependency

To activate this parameter, clear the **Use host-target connection** parameter.

`Remote server IP address` — IP address of the server device

`x.x.x.x`

Enter the IP address of the server to which you want to connect the client.

The addresses `0.0.0.0` and `255.255.255.255` are invalid IP addresses.

`Remote server port` — Port number of the server device

1–65535

Enter the port number of the server to which you want to connect the client.

Dependency

When you select the **Use host-target connection** parameter in the TCP configure blocks, you cannot use ports 22222 and 22223. Simulink Real-Time reserves these ports for its own use.

`PCI bus` — PCI bus number of dedicated Ethernet card
0 (default) | 0–31

Enter the PCI bus number for the dedicated Ethernet card.

Dependency

To activate this parameter, clear the **Use host-target connection** parameter.

`Slot` — PCI slot number of dedicated Ethernet card
0 (default) | 0–31

Enter the PCI slot number for the dedicated Ethernet card.

Dependency

To activate this parameter, clear the **Use host-target connection** parameter.

Model Examples

See Also

See Also

`SimulinkRealTime.target.getPCIInfo` | `TCP Receive` | `TCP Send`

Topics

"TCP Troubleshooting" on page 15-4

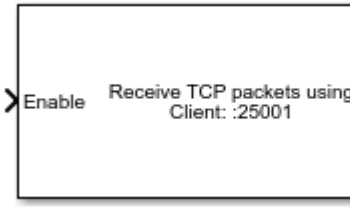
External Websites

www.ietf.org/rfc/rfc793.txt

Introduced in R2017a

TCP Receive

Receive data over TCP network from a remote device
Library: TCP



Description

Receive data sent from a remote client device to a server application on a target computer.

Ports

Input

`Enable` — Allow data reception

integer

When `Enable` > `0`, the block attempts to receive data sent to the remote device.

As a best practice, connect the `Status` output of a TCP configure block to the `Enable` input of the associated TCP Send and TCP Receive blocks.

Output

`Data` — Data that is received from the remote client

vector

The parameter **Receive width** determines the maximum size of the data vector.

Data Types: `uint8`

`Length` — Actual size of data vector

double

To test whether the number of data items exceeds the width of the data output port, use this value.

Parameters

`Receive using` — List of IP address and port pairs

`x.x.x.x:y`

This property is read-only.

The block receives the list of IP address and port pairs from the TCP configuration blocks in the model.

`Receive width` — Maximum expected length of data vector

1–65504

Maximum number of `uint8` values that the block expects to receive from the client device.

`Sample time` — Sample time of block

`-1` (default) | numeric

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

Model Examples

See Also

See Also

`TCP Client Configure` | `TCP Server Configure`

Topics

“TCP Troubleshooting” on page 15-4

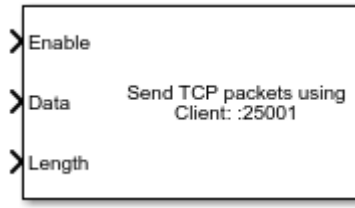
External Websites

www.ietf.org/rfc/rfc793.txt

Introduced in R2017a

TCP Send

Send data over TCP network to a remote device
Library: TCP



Description

Send data from a server application on a target computer to a remote client device.

Ports

Input

`Enable` — Allow data transmission
integer

When `Enable` > `0`, the block attempts to transmit data to the remote device.

As a best practice, connect the `Status` output of a TCP configure block to the `Enable` input of the associated TCP Send and TCP Receive blocks.

`Data` — Data to transmit over the TCP network
vector

Vector of length `Length` to transmit to the client device.

Data Types: `uint8`

`Length` — Length of data vector
double

Number of `uint8` values to transmit to the client device.

Output

`Status` — Device returns a status of not connected or connected
`0 | 1`

The status value is one of:

- `0` — Not connected
- `1` — Connected

Parameters

`Send using` — List of IP address and port pairs
`x.x.x.x:y`

This property is read-only.

The block receives the list of IP address and port pairs from the TCP configuration blocks in the model.

`Sample time` — Sample time of block
`-1` (default) | numeric

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

Model Examples

See Also

See Also
`TCP Client Configure` | `TCP Server Configure`

Topics
“TCP Troubleshooting” on page 15-4

External Websites

www.ietf.org/rfc/rfc793.txt

Introduced in R2017a

TCP Server

Configure TCP server application
Library: TCP

Description

Configure a TCP server application. This block assumes that a network interface has been configured for IP by the IP Config block.

Ports

Input

`Enable` — Connect block to remote Ethernet device
integer

If `Enable` is greater than zero, the block connects to the Ethernet device. Otherwise, the block does not connect.

Output

`Status` — Device returns a status of not connected or connected
`0 | 1`

The status value is one of:

- `0` — Not connected
- `1` — Connected

As a best practice, connect the `Status` output of a TCP configure block to the `Enable` input of the associated TCP Send and TCP Receive blocks.

Parameters

`Server IP address` — IP address of the server device that is being configured
`x.x.x.x`

If you are using the Ethernet connection between the development and target computers, this value must match the value of the `TcpIpTargetAddress` target setting. If you are using a dedicated Ethernet card, this value must match the **Local IP Address** parameter in the IP Config block for the network interface.

The addresses `0.0.0.0` and `255.255.255.255` are invalid IP addresses.

`Server port` — IP port of the server device that is being configured

1–65535

The combination of **Server IP address** and **Server port** must be unique.

Dependency

When you select the **Use host-target connection** parameter in the TCP configure blocks, you cannot use ports `22222` and `22223`. Simulink Real-Time reserves these ports for its own use.

See Also

See Also

`IP Config` | Target Settings Properties

Topics

"TCP Troubleshooting" on page 15-4

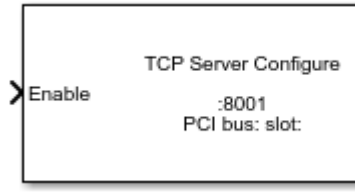
External Websites

www.ietf.org/rfc/rfc793.txt

Introduced in R2017a

TCP Server Configure

Configure TCP server application that uses the specified Ethernet interface
Library: TCP



Description

Configure a TCP server application and initialize a network interface for the application.

The combination of **Server IP Address** and **Subnet mask** must be unique across all Ethernet cards in the target computer, including the card for communicating between the development and target computers. Distinguish cards by specifying a different subnet for each. The subnet is the IP address masked by the subnet mask.

The Simulink Real-Time TCP blocks function only when executed on the target computer. When simulated on the development computer, they do nothing.

Ports

Input

`Enable` — Connect block to remote Ethernet device

integer

If `Enable` is greater than zero, the block connects to the Ethernet device. Otherwise, the block does not connect.

Output

`Status` — Device returns a status of not connected or connected

0 | 1

The status value is one of:

- `0` — Not connected
- `1` — Connected

As a best practice, connect the `Status` output of a TCP configure block to the `Enable` input of the associated TCP Send and TCP Receive blocks.

Parameters

`Use host-target connection` — Use Ethernet connection between development and target computers

`'off'` (default) | `'on'`

Dependency

Selecting the **Use host-target connection** parameter disables the **Server IP address**, **Subnet mask**, **PCI bus**, and **PCI slot** parameters and excludes the ports `22222` and `22223` from use by TCP.

`Server IP address` — IP address of the server device that is being configured

`x.x.x.x`

The addresses `0.0.0.0` and `255.255.255.255` are invalid IP addresses.

Dependency

To enable this parameter, clear the **Use host-target connection** parameter.

`Server port` — IP port of the server device that is being configured

`1–65535`

The combination of **Server IP address** and **Server port** must be unique.

Dependency

When you select the **Use host-target connection** parameter in the TCP configure blocks, you cannot use ports 22222 and 22223. Simulink Real-Time reserves these ports for its own use.

`Subnet mask` — Subnet mask for interface

`255.255.255.0` (default) | *`x.x.x.x`*

Mask that designates a logical subdivision of a network.

Dependency

To activate this parameter, clear the **Use host-target connection** parameter.

`Gateway` — IP address for gateway interface

`0.0.0.0` (default) | *`x.x.x.x`*

Gateway to access a different subnet. The gateway must be within the network.

To indicate "no gateway used," enter `0.0.0.0` (the default). The address `255.255.255.255` is an invalid gateway IP address.

Dependency

To activate this parameter, clear the **Use host-target connection** parameter.

`PCI bus` — PCI bus number of dedicated Ethernet card

0 (default) | 0–31

Enter the PCI bus number for the dedicated Ethernet card.

Dependency

To activate this parameter, clear the **Use host-target connection** parameter.

`Slot` — PCI slot number of dedicated Ethernet card

0 (default) | 0–31

Enter the PCI slot number for the dedicated Ethernet card.

Dependency

To activate this parameter, clear the **Use host-target connection** parameter.

Model Examples

See Also

See Also

`SimulinkRealTime.target.getPCIInfo` | `TCP Receive` | `TCP Send`

Topics

"TCP Troubleshooting" on page 15-4

External Websites

www.ietf.org/rfc/rfc793.txt

Introduced in R2017a

17

Real-Time UDP Communication Support

UDP Transport Protocol

The Simulink Real-Time software supports communication from the target computer to other systems or devices with User Datagram Protocol (UDP) packets. UDP is a transport protocol that 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.

UDP is a transport protocol layered on top of the Internet Protocol (IP). It is commonly known as UDP/IP.

- Packet — UDP is a *packet-oriented* protocol. You divide the data into packets and the protocol sends them to the receiver.
- Connectionless — UDP is a *connectionless* protocol. The protocol sends a packet to the receiver without checking to see if the receiver is ready to receive a packet. If the receiver is not ready, the packet is lost.
- No Error Detection— UDP does not support error detection. The protocol sends packets and does not track them. If packets arrive out of sequence, or are lost in transmission, the receiving end (or the sending end) does not know.

UDP is like sending letters by mail, without a return address. If the other party is not found, or the letter is lost in transit, it is discarded.

When describing UDP, the words *reliable* and *unreliable* have a specific meaning.

- *Reliable* means that the protocol is not guaranteed to succeed. It does not mean that the protocol always succeeds.
- *Unreliable* means that protocol packets can fail to arrive without the system detecting that the packets did not arrive. It does not mean that the protocol packets usually fail to arrive.

UDP continues to receive packets as long as the receiver is active and processes data as quickly as it arrives.

UDP is a commonly used transport layer because of its lightweight nature. When used from Simulink Real-Time, UDP gives the real-time application a good chance of succeeding in real-time execution. Also, the datagram nature of UDP is optimal for sending samples of data from the real-time application generated by the Simulink Coder software. If the real-time application cannot process the data as quickly as it arrives, only the most recent packet is used. The earlier packets are ignored.

You can construct a packet from Simulink data types such as `double`, `int8`, `int32`, `uint8`, or a combination of these data types. The Simulink Real-Time 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 preceding information applies to communication with a shared Ethernet board and communication with a dedicated Ethernet board. Consider adding a dedicated Ethernet board for enhanced performance over communication using a shared Ethernet board. Shared UDP communication shares bandwidth with the link between the development and target computers.

See Also

`Byte Unpacking` | `Byte Packing` | `Byte Reversal/Change Endianess` | `UDP Configure` | `UDP Receive` | `UDP Send`

More About

- "Target to Host Transmission using UDP"
- "Target to Target Transmission using UDP"
- "UDP Communication Setup" on page 17-11
- "UDP and Variable-Size Signals" on page 17-13

UDP Data Exchange with Shared Ethernet Board

In this section...

This example shows how to set up two-way data exchange with an Ethernet board that is shared with the connection between the development and target computers. Using this configuration, you can communicate between two Simulink Real-Time systems, between the Simulink Real-Time and Simulink products, or between two Simulink models. When one or both of the systems are running as a non-real-time Simulink model, be sure to set the sample time.

This example does not require a UDP Configure block because the example uses the connection between the development and target computers. To perform real-time UDP data transfer with a dedicated Ethernet board, see "Target to Target Transmission using UDP".

The example models are named `udpsendreceiveA` and `udpsendreceiveB`. Replace the port and IP address examples with ports and addresses as required by your network.

Data Transferred

The models transfer two different data sets between them, one data set from `udpsendreceiveA` to `udpsendreceiveB` and another data set in the opposite direction.

For this example, the inputs are generated with Simulink Constant blocks that use the MATLAB random number function (`rand`). The Simulink Coder software uses this function during code generation to generate random numbers. To generate the vector of `uint8` (3x3), use the MATLAB function:

```
uint8(255 * rand(3,3))
```

because `255` is the maximum value for an unsigned 8-bit integer. The other values are generated similarly.

`udpsendreceiveA` to `udpsendreceiveB`

The UDP data send is 75 bytes wide. The data to transfer is in the following formats.

- `[3 3]` of `uint8` (9 bytes)
- `[1 1]` of `uint16` (2 bytes)
- `[2 4]` of `double` (64 bytes)

When packed, the data is aligned on 1-byte boundaries.

`udpsendreceiveB` to `udpsendreceiveA`

The UDP data to be sent is 79 bytes wide. The data to transfer is in the following formats.

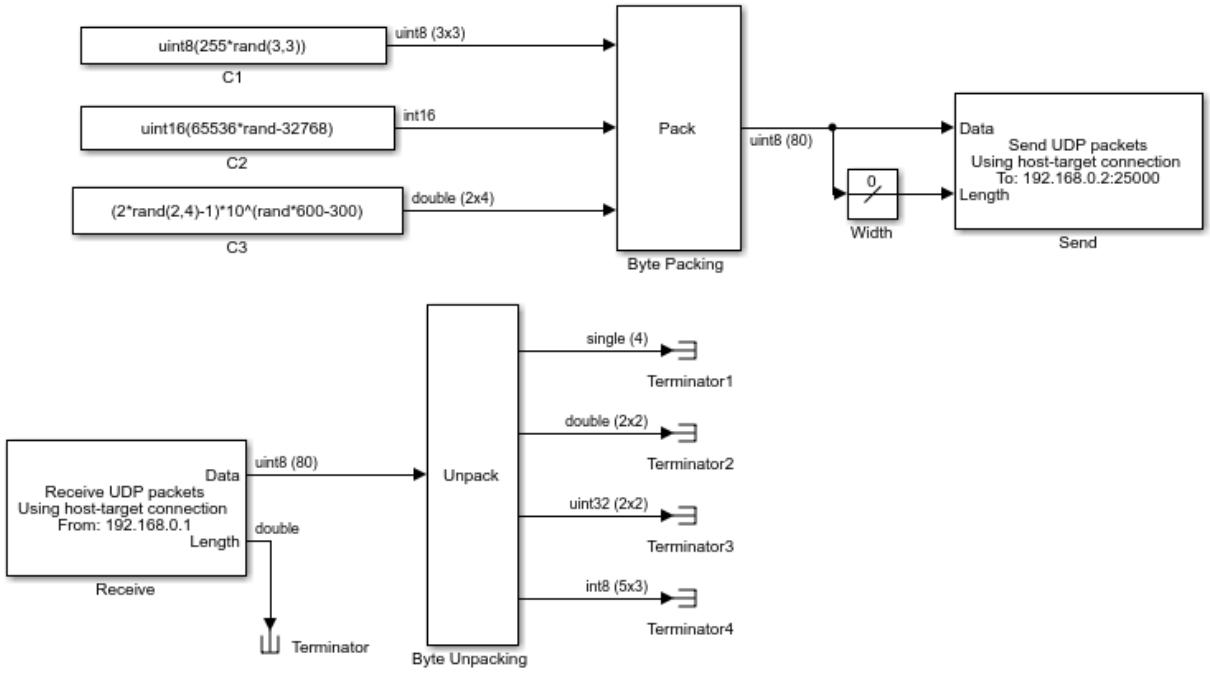
- `[4 1]` of `single` (16 bytes)
- `[2 2]` of `double` (32 bytes)
- `[2 2]` of `uint32` (16 bytes)
- `[5 3]` of `int8` (15 bytes)

When packed, the data is aligned on 2-byte boundaries. A zero-valued pad byte is added during packing.

Set Up `udpsendreceiveA`

The final `udpsendreceiveA` is shown in the figure.

The tables list the parameters for the send and receive sides of the model.



`udpsendreceiveA` Send Side

Block	Parameter	Value
Byte Packing	**Output port (packed) data type**	`'uint8'`
	Input port (unpacked) data types (cell array)	`{'uint8', 'uint16', 'double'}`
	Byte alignment	1
UDP Send	**Local IP address**	`Use host-target connection`
	Local port	`-1` (autoselect)
	To IP address	`192.168.0.2`
	To port	25000
	Sample time (-1 for inherited)	`0.01`

- The `Length` input port receives the output of a Width block that calculates the width of the signal connected to the `Data` port.
- The Byte Packing block settings match the Byte Unpacking block of `udpsendreceiveB`.

`udpsendreceiveA` Receive Side

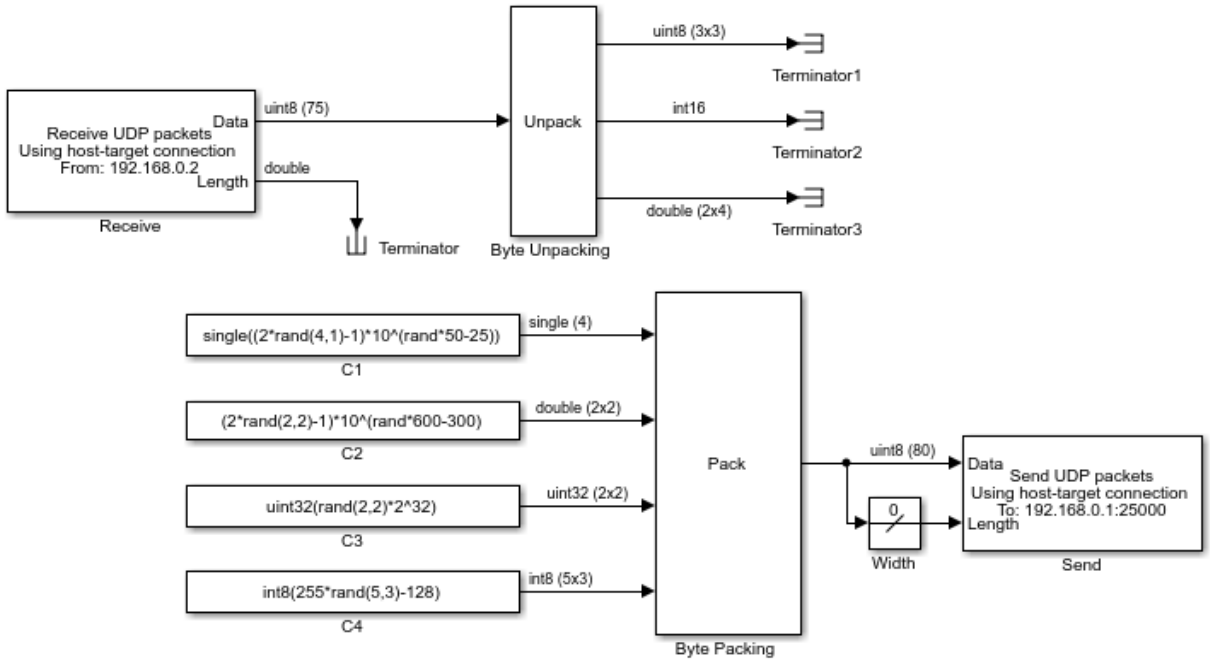
Block	Parameter	Value
UDP Receive	**Local IP address**	`Use host-target connection`
	Local port	`25000`
	Receive width	`80`
	Receive from any source	`off`
	From IP address	`192.168.0.1`
	Sample time (-1 for inherited)	`0.01`
Byte Unpacking	**Output port (unpacked) data types (cell array)**	`{'single', 'double', 'uint32', 'int8'}`
	Output port (unpacked) dimensions (cell array)	`{4, [2 2], [2 2], [5 3]}`
	Byte alignment	`2`

- The second output port of the UDP Receive block is sent into a terminator. You can use this output to determine when a packet has arrived. The same is true for the outputs of the Byte Unpack block, which in a real model would be used in the model.
- The **Receive width** of the UDP Receive block matches the output port width of the Byte Packing block in `udpsendreceiveB`.
- The Byte Unpacking block settings match the settings of the Byte Packing block of `udpsendreceiveB`.
- The number of unpacked bytes is `79`. The byte alignment is `2`, so the Byte Unpacking block assumes that the input vector includes a pad `0` to align the vector on an even-numbered boundary.

Set Up `udpsendreceiveB`

The final `udpsendreceiveB` model is shown in the figure.

The tables list the parameters for the receive side and the send side of the model.



`udpsendreceiveB` Receive Side

Block	Parameter	Value
UDP Receive	**Local IP address**	`Use host-target connection`
	Local port	`25000`
	Receive width	`75`
	Receive from any source	`off`
	From IP address	`192.168.0.2`
	Sample time (-1 for inherited)	`0.01`
Byte Unpacking	**Output port (unpacked) data types (cell array)**	`{'uint8', 'int16', 'double'}`

Block	Parameter	Value
	Output port (unpacked) dimensions (cell array)	`{[3 3], 1, [2 4]}`
	Byte alignment	1

- The second output port of the UDP Receive block is sent into a terminator. You can use this output to determine when a packet has arrived. The same is true for the outputs of the Byte Unpack block, which in a real model would be used in the model.
- The **Receive width** of the UDP Receive block matches the output port width of the Byte Packing block in `udpsendreceiveA`.
- The Byte Unpacking block settings match the Byte Packing block in `udpsendreceiveA`.

`udpsendreceiveB` Send Side

Block	Parameter	Value
Byte Packing	**Output port (packed) data type**	`'uint8'`
	Input port (unpacked) data types (cell array)	`{'single', 'double', 'uint32', 'int8'}`
	Byte alignment	2
UDP Send	**Local IP address**	`Use host-target connection`
	Local port	`-1` (autoselect)
	To IP address	`192.168.0.1`
	To port	25000
	Sample time (-1 for inherited)	`0.01`

- The `Length` input port receives the output of a Width block that calculates the width of the signal connected to the `Data` port.
- The Byte Packing block settings match the settings of the Byte Unpacking block of `udpsendreceiveA`.
- The number of unpacked bytes is `79`. The byte alignment is `2`, so the Byte Packing block pads the output vector with `0` to align on an even-numbered boundary.

See Also

`Byte Packing` | `Byte Unpacking` | `UDP Configure` | `UDP Receive` | `UDP Send`

More About

- "Target to Host Transmission using UDP"
- "Target to Target Transmission using UDP"
- "UDP Transport Protocol" on page 17-2
- "UDP Communication Setup" on page 17-11
- "UDP and Variable-Size Signals" on page 17-13

UDP Communication Setup

The infrastructure provided in the Simulink Real-Time Library for UDP communication consists mainly of two blocks: a UDP Send block and a UDP Receive block. These blocks are in the Simulink Real-Time Library, available from the Simulink Library under **Simulink Real-Time**. You can also access them from the MATLAB command line by typing:

```
slrtlib
```

The blocks are located under the `Real-Time UDP` heading in the library. The UDP Send block takes as input a vector of type `uint8`, which it sends. The UDP Receive block outputs a vector of `uint8`. To convert arbitrary Simulink data types into this vector of `uint8`, use a Byte Packing block. To convert a vector of `uint8`s back into arbitrary Simulink data types, use a Byte Unpacking block.

If you are using a dedicated Ethernet port for UDP communication, use a UDP Configure block to configure the Ethernet interface.

You can have up to 32 UDP blocks in a model—UDP Send and UDP Receive blocks combined in arbitrary order, plus the optional UDP Configure block.

To communicate with *big-endian* architecture systems, use the Byte Reversal/Change Endianess block. Your model does not need this block for communicating between 80x86-based computer systems running either the Simulink Real-Time kernel or the Microsoft® Windows operating system.

The blocks work from within the Simulink environment and from a real-time application running under the Simulink Real-Time system. Be cautious about transmitting data between a Simulink simulation and a real-time application, or using two Simulink models. A Simulink model is not a real-time model and can run several times faster or slower than a real-time application. Set the sample time of the UDP Send and UDP Receive blocks and the sample time of the Simulink model so that the blocks can communicate.

- You cannot configure two UDP Receive blocks with the same local port. For example, two UDP Receive blocks cannot have the same local port and different IP addresses.
- You cannot configure two UDP Send blocks with the same local port. For example, two UDP Send blocks cannot have the same local port and different IP addresses.

See Also

`Byte Packing` | `Byte Unpacking` | `UDP Configure` | `UDP Receive` | `UDP Send`

More About

- "Target to Host Transmission using UDP"
- "Target to Target Transmission using UDP"
- "UDP Transport Protocol" on page 17-2

UDP and Variable-Size Signals

The Simulink Real-Time UDP sublibrary does not directly support variable-size signals. The UDP Send block input port accepts only fixed-size signals.

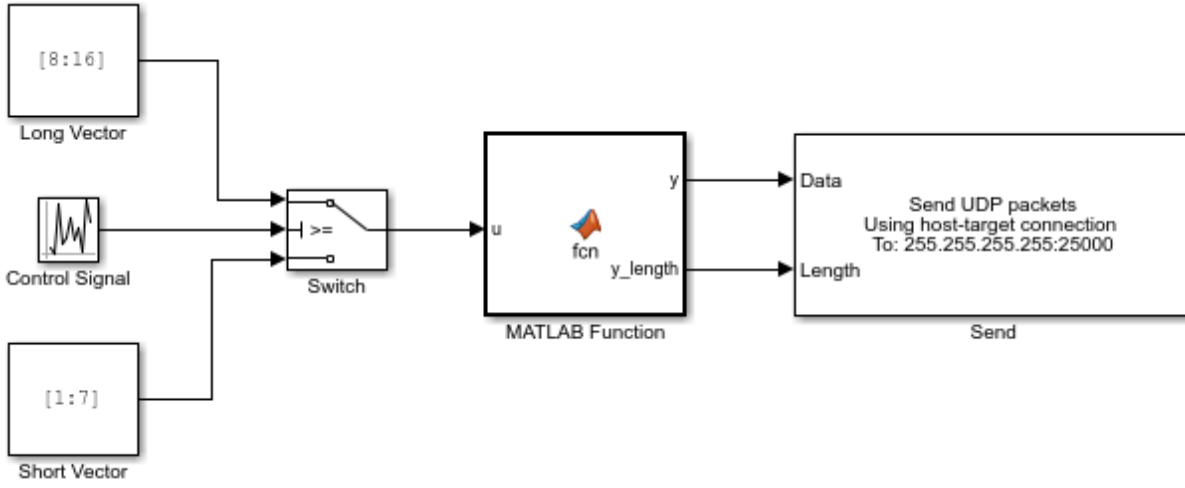
To send variable-size signals though UDP, determine the maximum number of elements of a fixed-size input signal that you expect to connect to the block. Then use the second input, `Length`, to specify the number of elements of this input signal to send through UDP.

This example configures the MATLAB Function block to accept a variable-size signal and maps that signal to a fixed-size output signal. It outputs the number of relevant elements. You can output the fixed-size output signal and number of elements to the inputs of the UDP Send block.

1. To accept a variable-size input signal, create a MATLAB Function block.
2. In the MATLAB Function block, enter code like the following code. 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
```

3. In the MATLAB Function Editor, select **Tools** > **Edit Data/Ports**. In **Ports and Data Manager**, select the data `u`, and then select the corresponding **Variable size** check box.
4. 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`.
5. Provide a variable-size signal source for the MATLAB Function block.



See Also

MATLAB Function | `UDP Send`

Real-Time UDP Troubleshooting

In this section...

Duplicate Subnet Calculated in Block

You can use a dedicated Ethernet card for TCP communication while using another card for communicating between the development and target computers. You can get the following error during model initialization:

```
The subnet in this block is the same as or is a subset of the subnet
calculated in ''block''. The block calculates the subnet by ANDing the
IP address bitwise with the subnet mask.
```

Check the IP address and subnet you assigned to the target computer Ethernet card in the configuration block. The UDP implementation requires that the two communication channels use separate subnets.

The block calculates the subnet by ANDing the IP address bitwise with the subnet mask for each card. For example, the following specifications result in the same subnet for both cards.

```
E1 (development-target): IP address:        192.168.0.25
                         Subnet mask:       255.255.255.0
                                            -------------
                         Calculated Subnet: 192.168.0.0

E2 (RT-UDP):             IP address:        192.168.0.26
                         Subnet mask:       255.255.255.0
                                            -------------
                         Calculated Subnet: 192.168.0.0
```

Try a configuration such as the following:

```
E1 (development-target): IP address:        192.168.0.25
                         Subnet mask:       255.255.255.0
                                            -------------
                         Calculated Subnet: 192.168.0.0
```

```
E2 (RT-UDP):            IP address:        192.168.0.26
                        Subnet mask:       255.255.255.2
                                           -------------
                        Calculated Subnet: 192.168.0.2
```

In some networks, the development computer must also be in the subnet where the TCP communication occurs. You can either add a second network card to the development computer or provide a gateway device to create a dedicated network for TCP communication.

Excluded Ports When Using Development-Target Computer Connection

When you use the same IP address as the communication channel between the development and target computers, you cannot use ports `22222` and `22223`. Simulink Real-Time reserves these ports for its own use.

More About

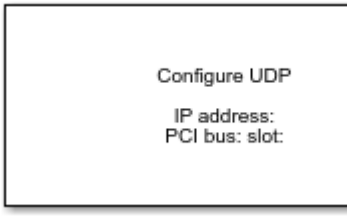
- "TCP/IP and UDP Interface" (Instrument Control Toolbox)

Real-Time UDP Blocks

UDP Configure

Initialize Ethernet network interface to use for UDP communication in real-time applications

Library: Real-Time UDP



Description

The Real-Time UDP Configuration block configures a dedicated Ethernet network for real-time UDP operation.

The combination of **Local IP Address** and **Subnet mask** must be unique across all Ethernet cards in the target computer, including the card for communicating between the development and target computers. Distinguish cards by specifying a different subnet for each. The subnet is the IP address masked by the subnet mask.

Parameters

`Local IP Address` — IP address for the Ethernet interface
`x.x.x.x`

Enter the IP address for the dedicated Ethernet board.

The addresses `0.0.0.0` and `255.255.255.255` are invalid local IP addresses.

`Subnet mask` — Subnet mask for interface
`255.255.255.0` (default) | *`x.x.x.x`*

Mask that designates a logical subdivision of a network.

`Gateway` — IP address for gateway interface
`0.0.0.0` (default) | *`x.x.x.x`*

The gateway must be within the network.

To indicate "no gateway used," enter `0.0.0.0` (the default). The address `255.255.255.255` is an invalid gateway IP address.

`PCI bus` — PCI bus number of Ethernet card
0 (default) | 0–31

Enter the PCI bus number for the Ethernet card.

`PCI slot` — PCI slot number of Ethernet card
0 (default) | 0–31

Enter the PCI slot number for the Ethernet card.

Model Examples

See Also

See Also
`SimulinkRealTime.target.getPCIInfo` | `UDP Receive` | `UDP Send`

Topics
"UDP Transport Protocol" on page 17-2
"Real-Time UDP Troubleshooting" on page 17-15

Introduced in R2016b

UDP Receive

Receive data over UDP network from a remote device
Library: Real-Time UDP

Description

The UDP Receive block receives data over a UDP network from a remote device. It can receive data by using the connection between the development and target computers or by using a dedicated Ethernet card. If you use a dedicated Ethernet card, add a UDP Configure block to your model.

The parameter **Local IP address** applies only when the block executes on a target computer. If your model is running in Simulink on the development computer, you can use this block to transmit data to a remote device. In this case, the Windows operating system determines the network connection.

Ports

Output

`Data` — Data received
vector

Vector of `uint8` containing data received over the UDP network. If no new packet is received, the data values are held. To determine whether a new packet has been received, use the **Length** output port.

Data Types: `uint8`

`Length` — Number of bytes received

double

Number of bytes in the new packet received, otherwise `0`. If more bytes are received than can be output through the receive port with width defined by **Receive width**, the excess bytes are discarded.

Parameters

`Local IP address` — Destination IP address for receiving data

`Use host-target connection` (default)

When **Local IP address** is set to `Use host-target connection`, the block uses the connection between the development and target computers. Otherwise, the block uses the value that you set in the **Local IP address** parameter of the UDP Configure block.

`Local port` — Destination UDP port through which to receive data

1–65535

Specifies UDP port through which to receive data.

`Receive width` — Width of `Data` output vector

1–65504

Determines the width of the `Data` output vector. If this value is less than the number of bytes in the received packet, the excess bytes are discarded.

`Receive from any source` — Causes receiver to accept data from any IP address

`on` (default) | `off`

When **Receive from any source** is `on`, the block receives data from any accessible IP address. When it is `off`, the block receives data from only the address that you specify in **From IP address**.

Dependency

To make the **From IP address** parameter visible, clear **Receive from any source**.

`From IP address` — Source from which to receive data

`0.0.0.0` (default) | *`x.x.x.x`*

Enter a valid IP address as a dotted decimal character vector, for example, `10.10.10.3`. You can also use a MATLAB expression that returns a valid IP address as a character vector.

The default address, `0.0.0.0`, causes the block to accept UDP packets from any accessible device. If you set **From IP address** to a specific IP address, only packets arriving from that IP address are received.

The address `255.255.255.255` is an invalid IP address.

Dependency

To make this parameter visible, clear **Receive from any source**.

`Sample time (-1 for inherited)` — Sample time of block
`-1` (default) | numeric

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

Model Examples

See Also

See Also

`Byte Reversal/Change Endianess` | `Byte Unpacking` | `UDP Configure` | `UDP Send`

Topics

"UDP Transport Protocol" on page 17-2
"Real-Time UDP Troubleshooting" on page 17-15

Introduced in R2016b

UDP Send

Send data over UDP network to a remote device
Library: Real-Time UDP

Description

The UDP Send block sends data over a UDP network to a remote device. The block can send data by using the connection between the development and target computers or by using a dedicated Ethernet card. If you use a dedicated Ethernet card, add a UDP Configure block to your model.

The parameter **Local IP address** applies only when the block executes on a target computer. If your model is running in Simulink on the development computer, you can use this block to transmit data to a remote device. In this case, the Windows operating system determines the network connection.

To broadcast to all devices, set **To IP address** to `255.255.255.255`, otherwise set **To IP address** to a valid IP address.

Ports

Input

`Data` — Data to transmit

vector

Vector of `uint8` containing data to transmit over the UDP network. To determine how many bytes of data to transmit, use the **Length** input port.

Data Types: `uint8`

`Length` — Number of bytes of data to transmit

double

Determines the number of bytes of data to transmit. Specify the width of the `Data` vector as the maximum number of bytes that you expect to transmit.

Parameters

`Local IP address` — Source IP address for sending data

`Use host-target connection` (default)

When **Local IP address** is set to `Use host-target connection`, the block uses the connection between the development and target computers. Otherwise, the block uses the value that you set in the **Local IP address** parameter of the UDP Configure block.

`Local port` — Source UDP port through which to transmit data

`1–65535 | -1`

Specifies local UDP port through which to transmit data.

The value `-1` means that the block transmits using any available port.

`To IP address` — IP address of target device

`255.255.255.255` (default) | *`x.x.x.x`*

Specifies IP address of target device. To broadcast, send to `255.255.255.255`.

`To port` — UDP port of target device

`1–65535`

Specify the UDP port of target device. With **To IP address**, this parameter defines the destination of the data transmission.

`Sample time (-1 for inherited)` — Sample time of block

`-1` (default) | numeric

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

Model Examples

See Also

See Also

`Byte Reversal/Change Endianess` | `Byte Packing` | `UDP Configure` | `UDP Receive`

Topics

“UDP Transport Protocol” on page 17-2
“Real-Time UDP Troubleshooting” on page 17-15

Introduced in R2016b

Parallel Ports, PTP, SAE J1939, Shared Memory

19

Parallel Ports

Using Parallel Ports

In this section...

Introduction

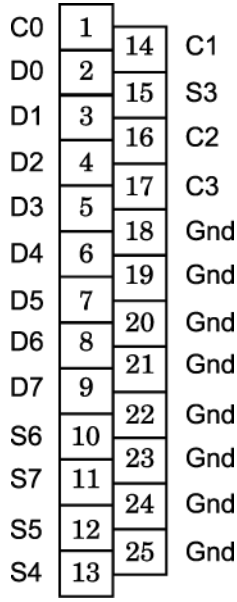
Most target computers have a parallel port that you can use for various devices. The Simulink Real-Time block library provides blocks that enable you to use the parallel ports of a target computer for digital input and output, and source interrupts.

Caution: The parallel port is part of the motherboard on many computers. Be careful when configuring the port and when connecting external devices to the port. Incorrect connections to the port can damage your computer.

The Simulink Real-Time 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				
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 Simulink Real-Time model as follows:

1. Select **Simulation > Model Configuration Parameters**.
2. Under node **Code Generation**, select node **Simulink Real-Time 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`).

- **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, call the function `SimulinkRealTime.target.getPCIInfo` with the `'verbose'` option. For example:

   ```
   tg = slrt;
   getPCIInfo(tg, '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.

Parallel Port Blocks

Parallel Port Digital Input

Parallel Port Digital Input block

Library

Simulink Real-Time Library for Parallel Port

Scaling Input to Output

I/O Module Input	Block Output Data Type	Scaling
TTL	Double (**Format**: `8 1-bit Channels`)	Double: TTL low = 0.0 TTL high = 1.0
	uint8 (**Format**: `One 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 target computer 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`
- `ox278`
- `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
```

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 (`-1` means sample time is inherited).

Introduced in R2007a

Parallel Port Digital Input Status Bits

Parallel Port Digital Input Status Bits block

Library

Simulink Real-Time Library for Parallel Port

Scaling Input to Output

I/O Module Input	Block Output Data Type	Scaling
TTL	Double (**Format**: `5 1-bit Channels`)	Double: TTL low = 0.0 TTL high = 1.0
	uint8 (**Format**: `One 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 target computer 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`
- `ox278`
- `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
```

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 (`-1` means sample time is inherited).

Introduced in R2007a

Parallel Port Digital Output

Parallel Port Digital Output block

Library

Simulink Real-Time Library for Parallel Port

Scaling Output to Input

I/O Module Output	Block Input Data Type	Scaling
TTL	Double (**Format**: `8 1-bit Channels`)	Double: < 0.5 = TTL low > 0.5 = TTL high
	uint8 (**Format**: `One 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 target computer 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`
- `ox278`
- `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
```

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 replicated as the initial value over the channel vector. 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 replicated over the channel vector. If you specify 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 replicated over the channel vector. 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 (`-1` means sample time is inherited).

Introduced in R2007a

Parallel Port Digital Output Control Bits

Parallel Port Digital Output Control Bits block

Library

Simulink Real-Time Library for Parallel Port

Scaling Output to Input

I/O Module Output	Block Input Data Type	Scaling
TTL	Double (**Format**: `4 1-bit Channels`)	Double: < 0.5 = TTL low > 0.5 = TTL high
	uint8 (**Format**: `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 target computer 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`
- `ox278`
- `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
```

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 replicated as the initial value over the channel vector. 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 replicated over the channel vector. If you specify 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 replicated over the channel vector. 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 (`-1` means sample time is inherited).

Introduced in R2007a

Precision Time Protocol

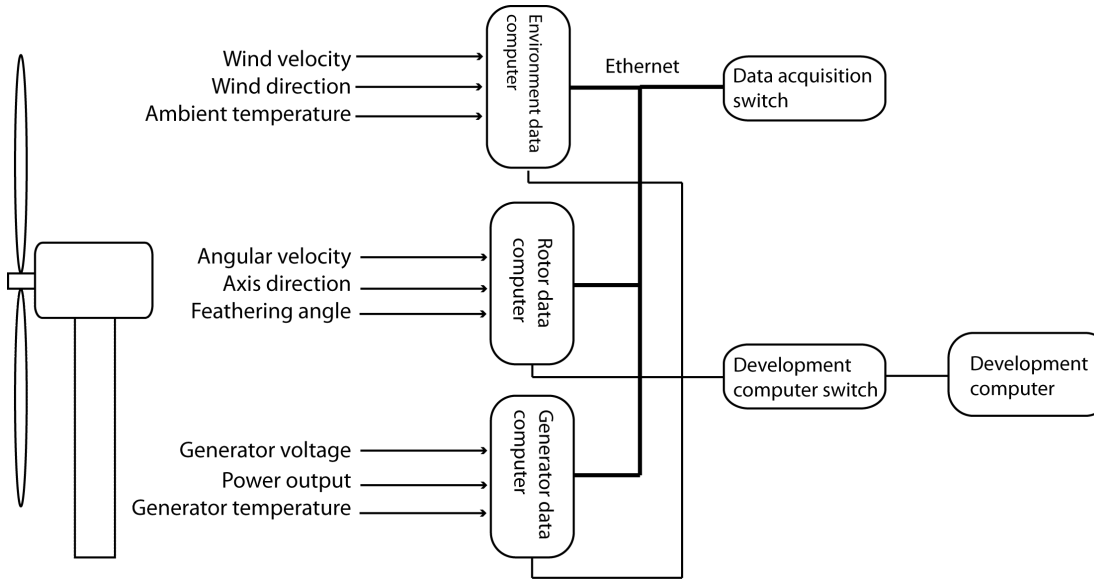
Precision Time Protocol

Measurement and control systems increasingly use distributed system technologies. To distribute measurement or control tasks over interconnected computing devices, such systems maintain a system-wide sense of time. Simulink Real-Time uses the Precision Time Protocol (PTP) to synchronize the PTP clock of each computer to a reference time. The PTP clock of a Simulink Real-Time target computer is the clock on the PTP network card.

PTP (IEEE 1588) is a protocol that synchronizes PTP clocks throughout a computer network. The current version of PTP (IEEE 1588-2008) describes a hierarchical master-slave architecture for clock distribution.

By design, this protocol is more accurate for local systems than the Network Time Protocol (NTP) and more robust than the Global Positioning System (GPS). On a local area network, the protocol achieves PTP clock accuracy in the submicrosecond range, making it suitable for distributed measurement. When you use this protocol to synchronize Simulink Real-Time applications across multiple target computers, it can synchronize execution to under 10 μs.

Suppose that you are designing a control system for a wind power plant. To determine the plant parameters, you attach sensors that acquire the data shown in the diagram.



To record the data and timestamps, you connect the sensors to a set of data acquisition target computers. You interconnect the data acquisition computers through an Ethernet network and a switch that supports the PTP protocol (a PTP transparent clock or boundary clock). To access the data and timestamps, you connect the target computers to a development computer through another Ethernet network and switch. On the development computer, you run MATLAB to do the data analysis, including:

- Sorting by time the data recorded on the different computers to analyze the event sequence over time.
- Filtering sensor data that have invalid (unsynchronized) timestamps.
- Integrating values of measured data collected at the same time from sensors connected to different computers.

To synchronize the target computer PTP clocks, you create a Simulink Real-Time model for each target computer. Each model uses the following PTP blocks:

- IEEE 1588 Ethernet — Run PTP protocol with Raw Ethernet as transport protocol. This block communicates with the corresponding blocks on the other target computers and determines the time offset that synchronizes them.
- IEEE 1588 Read Parameter — Output a Precision Time Protocol parameter value. Of the possible output values, you select `PTP time (nanosecond)`.

For debugging, you can configure a separate IEEE 1588 Read Parameter block to read other values, such as `Protocol state`.

- IEEE 1588 Sync Execution — Synchronize model execution to Precision Time Protocol clock. You can now make measurements at the same time step.
- IEEE 1588 Sync Status — Output the synchronization status of the Precision Time Protocol. When the value is `true`, the data timestamps are synchronized to the required precision.

As a best practice, for each model, you enclose the sensor block and the IEEE 1588 Read Parameter and IEEE 1588 Sync Status blocks in an Atomic Subsystem block. By using the Atomic Subsystem block, you bring the PTP timestamp as close as possible to the time of the data measurement.

Finally, you build and download the real-time applications to each target computer, run the applications, and collect and analyze the results at each valid timestamp. You use the results to design a control system for the wind power generator.

See Also

Atomic Subsystem | `IEEE 1588 Read Parameter` | `IEEE 1588 Ethernet` | `IEEE 1588 Sync Execution` | `IEEE 1588 Sync Status`

More About

- “Synchronize Timestamps Across Data-Gathering Network” on page 21-5
- “IEEE® 1588™ Precision Time Protocol - Execution Synchronization”
- “Data Acquisition and Data Analysis Example Description” on page 21-18
- “Prerequisites, Limitations, and Unsupported Features” on page 21-30

External Websites

- standards.ieee.org

Synchronize Timestamps Across Data-Gathering Network

This example shows a data acquisition target computer that transmits timestamped data to a second target computer that analyzes the data.

Required Products: Simulink®, Simulink Real-Time™

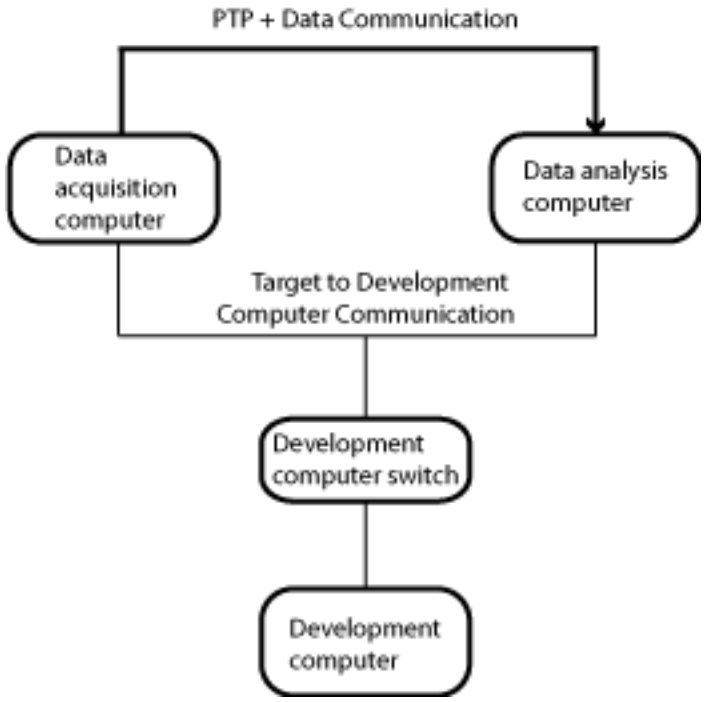
Other Requirements:

- One Windows® development computer with an Ethernet card.
- Two Speedgoat target computers.
- At least one Intel® 82574 Ethernet card on each target computer
- One Ethernet card on each target computer dedicated to communication between the development and target computers.
- One Ethernet switch.
- Four crossover Ethernet cables.

The real-time applications use Precision Time Protocol blocks to synchronize the Intel 82574 Ethernet card PTP clocks and the kernel clocks for each computer. You can log the PTP timestamps from both computers and use them to associate transmitted data with reference data.

Configure Hardware

Your Speedgoat target machines include at least two Ethernet cards installed, one of them an Intel 82574 Ethernet card. Use the Intel 82574 Ethernet card for the PTP network. Use the other to connect the two target computers through the Ethernet switch to the development computer. The network looks like this figure.



Required Information

To configure the network and your models for this example, collect the following information for each target computer:

- Identifiers
- *PTP card:* Device name, PCI bus and slot numbers, MAC address that you assign to the PTP card that is transmitting non-PTP data
- *COM card:* Device name, PCI bus and slot numbers, Ethernet index of the card

You can find the built-in MAC address of the PTP card from a source such as a bill of materials or a label on the hardware. You can find the device name and the PCI bus and slot numbers by using SimulinkRealTime.target.getPCIInfo. You can find the Ethernet index of the communication card by using SimulinkRealTime.getTargetSettings.

Example Information

TargetPC1

Identifier — TargetPC1

PTP card

- Device name — `Intel 82574L`
- PCI bus — `5`
- PCI slot — `0`
- MAC Address — [EEPROM}

COM card

- Device name — `Intel 82579LM`
- Ethernet index — `0`
- PCI bus — `0`
- PCI slot — `25`
- MAC address — N/A

TargetPC2

Identifier — TargetPC2

PTP card

- Device name — `Intel 82574L`
- PCI bus — `0`
- PCI slot — `52`
- MAC Address — `68:05:CA:31:B9:EF`

COM card

- Device name — `Intel 82541GI_LF`
- Ethernet index — `0`
- PCI bus — `16`
- PCI slot — `4`
- MAC address — N/A

Hardware Configuration

1. Connect an Ethernet cable between the PTP card in `TargetPC1` and the PTP card in `TargetPC2`. This connection creates the PTP network. To configure the IEEE 1588 Ethernet blocks in the two real-time applications, you must have the PCI bus and PCI slot of these cards.
2. Connect an Ethernet cable from the Comm card in `TargetPC1` to the Ethernet switch.
3. Connect an Ethernet cable from the Comm card in `TargetPC2` to the switch.
4. Connect an Ethernet cable from the switch to the development computer. These connections complete the communication network between the development computer and the target computers.
5. Start the two target computers. Use slrtexplr to connect to them. If you cannot establish communication with a target computer, verify the Ethernet index assigned to the communication port.

Configure Real-Time Applications

In this example, the system contains two real-time applications running on separate target computers. The data acquisition application transmits PTP and non-PTP data to the data analysis application. To configure the data acquisition application, you must have the MAC address of the PTP card that is installed in the data analysis target computer. The data analysis application transmits only PTP data to the data acquisition application. To configure the data analysis application, you can use the MAC address stored in the EEPROM of the PTP card in the data acquisition target computer.

Configure Data Acquisition Application

To configure this application, first perform the steps in Configure Hardware. Collect the PCI bus, PCI slot, and role of the Ethernet cards installed in the target computers.

The data acquisition application is `ex_ptp_sync_src` (matlab:open_system(docpath(fullfile(docroot, 'toolbox', 'xpc', 'examples', 'ex_ptp_sync_src')))). It runs on `TargetPC1`. To configure the application:

Set Configuration Parameters

1. Open `ex_ptp_sync_src`.
2. In the Configuration Parameters dialog box, open the **Solver** pane.
3. Verify that **Type** is `Fixed-step`.

4. Verify that **Fixed-step size (fundamental sample time)** is set to an explicit value and not to `auto` (best practice). Use the same sample time as in the data analysis application.
5. Verify that **Allow tasks to execute concurrently on target** is set.
6. Take the defaults for the other settings.
7. Open the Simulink Real-Time Options pane.
8. Verify that **Build for default target computer** is cleared.
9. Verify that **Specify target computer name** is `TargetPC1`.
10. Verify that **Execution mode** is `Real-Time`.
11. Take the defaults for the other settings.

Configure PTP Blocks

1. Open the IEEE 1588 Ethernet block
2. Open the General pane.
3. From the information for the `TargetPC1` PTP card, enter the values `5` and `0` for **PCI bus** and **PCI slot**.
4. Verify that the IEEE 1588 Ethernet **Sample time** value is a multiple of the **Fixed-step size (fundamental sample time)** value.
5. Verify that **Sample time** has the same value as in the data analysis model.
6. Open the **Network parameters** pane.
7. From the information for the `TargetPC1` PTP card, in the **Source MAC address** box, select `EEPROM`.
8. Verify that **Destination MAC address** is `Standard PTP multicast`.
9. Open the **Clock parameters** pane.
10. Verify that **Timescale (epoch)** is `PTP (1970-01-01)`.
11. Verify that **Delay measurement mechanism** is `Request-response`.
12. Set the **Slave only** check box. This setting prevents the software from making this node the master PTP clock node.
13. Open the **Time intervals** pane.
14. Verify that **Announce interval (second)**, **Sync interval (second)**, and **Min delay or pdelay request interval (second)** are at least three times the **Sample time** value.

15 Verify that the intervals are integral multiples of **Sample time**.

16 Verify that the intervals have the same settings as in the data analysis model.

17 In the remaining top-level PTP blocks, verify that the **Sample time** value matches that in the IEEE 1588 Ethernet block.

18 Open PTP Clock-Data Subsystem. For each block in the subsystem, verify that the **Sample time** value matches that in the IEEE 1588 Ethernet block.

Configure Data Communication Blocks

1 From the information for the `TargetPC2` PTP card, in the Create Ethernet Packet block dialog box, set **Destination MAC** to `macaddr('68:05:CA:31:B9:EF')`.

2 Set **EtherType (use 0 for length)** to `hex2dec('0010')`. Using this type distinguishes the data-specific messages from the PTP-specific messages, which use the same Ethernet card.

3 In the Ethernet Tx block, verify that the **Sample time** value matches that in the IEEE 1588 Ethernet block.

4 Save the updated model in your working folder. You cannot build and run a real-time application in the examples folder.

Configure Data Analysis Application

To configure this application, first perform the steps in Configure Hardware. Collect the PCI bus, PCI slot, and role of the Ethernet cards that are installed in the target computers. To configure the PTP card in the data analysis target computer, use the MAC address that you specified in the Create Ethernet Packet block of the data acquisition application.

The data analysis application is `ex_ptp_sync_sink` (matlab:open_system(docpath(fullfile(docroot, 'toolbox', 'xpc', 'examples', 'ex_ptp_sync_sink')))). It runs on `TargetPC2`. To configure the application:

Set Configuration Parameters

1 Open `ex_ptp_sync_sink`.

2 In the Configuration Parameters dialog box, open the **Solver** pane.

3 Verify that **Type** is `Fixed-step`.

4 Verify that **Fixed-step size (fundamental sample time)** is set to an explicit value (best practice) and not to `auto`. Use the same sample time as in the data acquisition application.

5. Verify that **Allow tasks to execute concurrently on target** is set.
6. Take the defaults for the other settings.
7. Open the **Simulink Real-Time Options** pane.
8. Verify that **Build for default target computer** is cleared.
9. Verify that **Specify target computer name** is `TargetPC2`.
10. Verify that **Execution mode** is `Real-Time`.
11. Take the defaults for the other settings.

Configure PTP Blocks

1. Open the IEEE 1588 Ethernet block.
2. Open the **General** pane.
3. From the information for the `TargetPC2` PTP card, enter the values `52` and `0` for **PCI bus** and **PCI slot**.
4. Verify that the IEEE 1588 Ethernet Sample time value is a multiple of the Fixed-step size (fundamental sample time) value.
5. Verify that Sample time has the same value as in the data acquisition IEEE 1588 Ethernet block.
6. Open the **Network parameters** pane.
7. In the **Source MAC address** box, select `Specify`.
8. From the information for the `TargetPC2` PTP card, in the **Specify source MAC address** text box, enter macaddr('68:05:CA:31:B9:EF'). You can enter an arbitrary MAC address in this box, provided it is unique in the PTP network.
9. Verify that **Destination MAC address** is `Standard PTP multicast`.
10. Open the **Clock parameters** pane.
11. Verify that **Timescale (epoch)** is `PTP (1970-01-01)`.
12. Verify that **Delay measurement mechanism** is `Request-response`.
13. Verify that **Slave only** is cleared. This setting allows the software to make this node the master PTP clock node.
14. Open the **Time intervals** pane.
15. Verify that **Announce interval (second)**, **Sync interval (second)**, and **Min delay or pdelay request interval (second)** are at least three times the Sample time value.

16 Verify that the intervals are integral multiples of **Sample time**.

17 Verify that the intervals have the same settings as in the data acquisition model.

18 In the remaining top-level PTP blocks, verify that the **Sample time** value matches that in the IEEE 1588 Ethernet block.

19 Open PTP Clock-Data Subsystem. For each block in the subsystem, verify that the **Sample time** value matches that in the IEEE 1588 Ethernet block.

Configure Data Communication Blocks

1 Open PTP Clock-Data Subsystem

2 Open the Ethernet Rx block.

3 Open the **Rx** pane and verify that the **Sample time** value matches that in the IEEE 1588 Ethernet block.

4 Open the **Filter** pane.

5 Verify that **Filter criteria** is `Specify types to match`.

6 Verifty that **Receive these types (vector of types 0-65535)** is `[hex2dec('0010')]`.

7 Save the updated model in your working folder. You cannot build and run a real-time application in the examples folder.

Build, Download, and Run Real-Time Applications

In this example, the data acquisition application builds and is downloaded to `TargetPC1`. The data analysis application builds and is downloaded to `TargetPC2`. To run these applications, first perform the steps in Configure Real-Time Applications. MATLAB® and Simulink Real-Time Explorer must be running in your working folder.

Build, download, and run the applications:

1 Start `TargetPC1` and `TargetPC2`.

2 In the Explorer Targets pane, connect to `TargetPC1` and `TargetPC2`.

3 In your working folder, open `ex_ptp_sync_src` and `ex_ptp_sync_sink`.

4 Build and download `ex_ptp_sync_src` to `TargetPC1`.

5 Build and download `ex_ptp_sync_sink` to `TargetPC2`.

6 In the Explorer **Applications** pane, for `TargetPC1/ex_ptp_sync_src` and `TargetPC2/ex_ptp_sync_sink`, change the property **Stop Time** to `Inf`.

7. In the Explorer **Applications** pane, start `ex_ptp_sync_src` and `ex_ptp_sync_sink`.
8. For both applications, waveform data starts streaming in the target scopes labeled `Data`. However, the timestamps displayed as signal `Time` are not initially valid. The two applications go through the following sequence of `Sync Status` and `State` values:

1. Initialization:

- `ex_ptp_sync_src State → 4` (LISTENING)
- `ex_ptp_sync_src Sync Status → 0` (not synchronized)
- `ex_ptp_sync_sink State → 4` (LISTENING)
- `ex_ptp_sync_sink Sync Status → 0` (not synchronized)

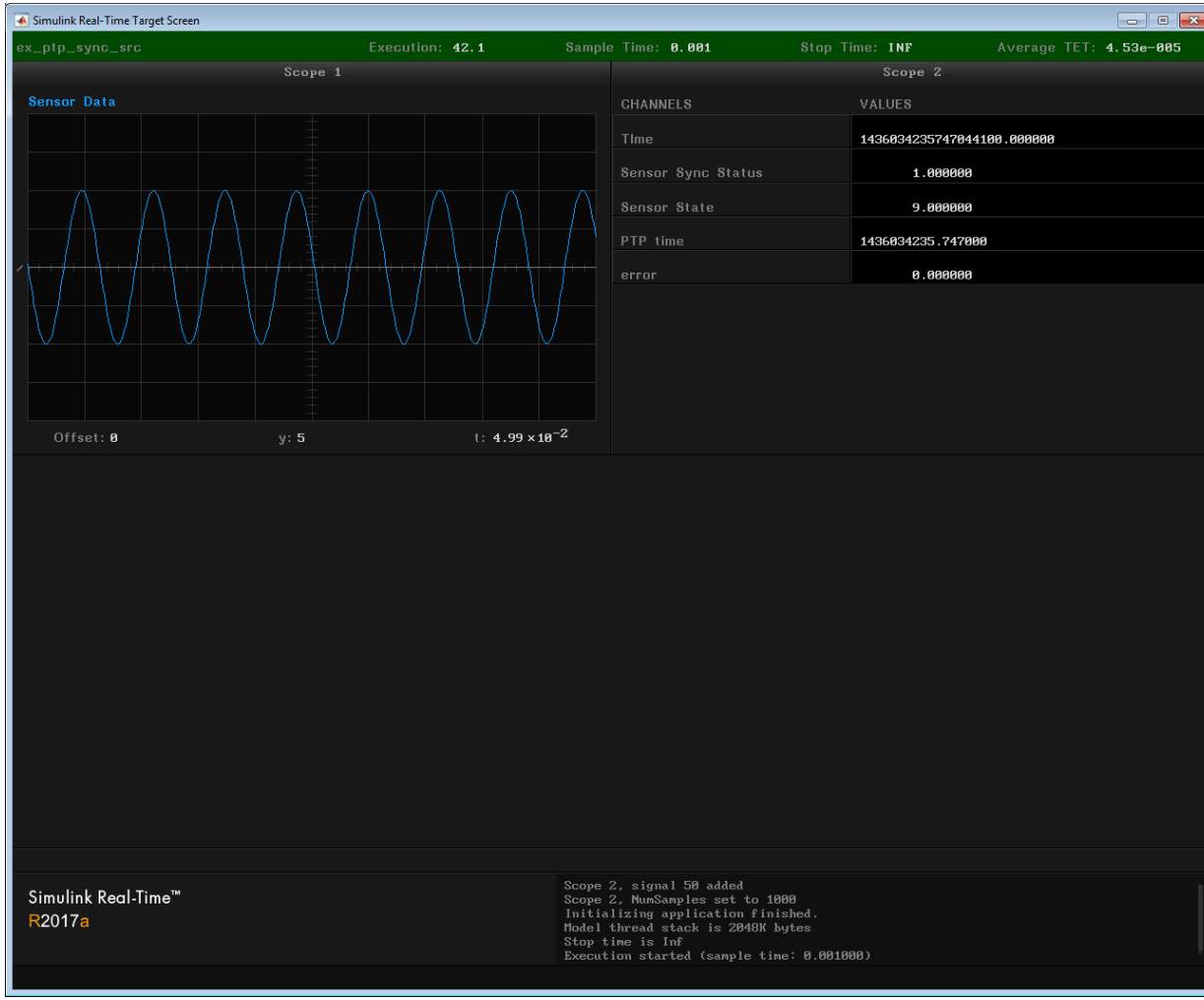
2. Master allocation and synchronization

- `ex_ptp_sync_sink State → 6` (MASTER)
- `ex_ptp_sync_sink Sync Status → 1` (synchronized)

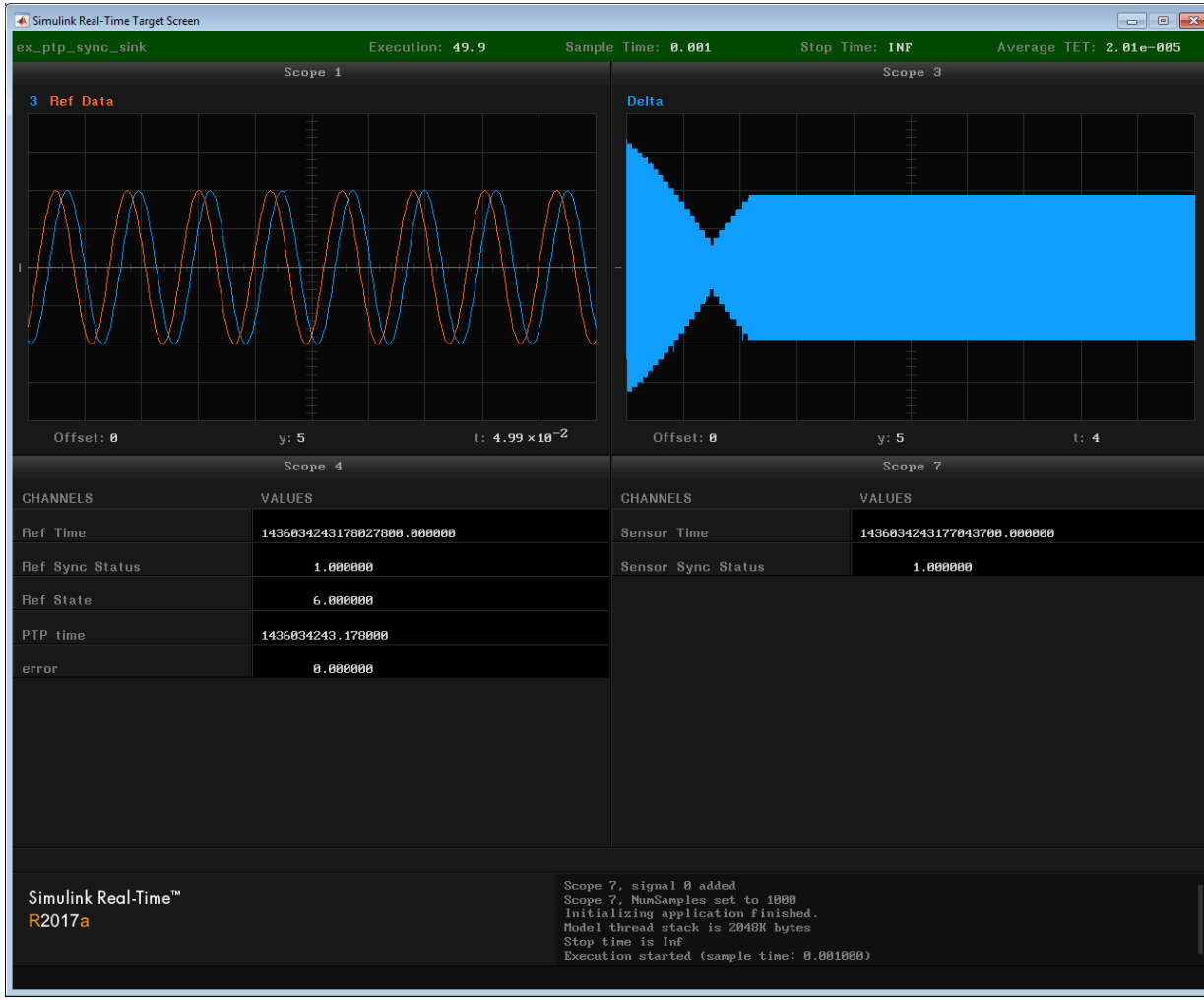
3. Slave allocation and synchronization

- `ex_ptp_sync_src State → 9` (SLAVE)
- `ex_ptp_sync_src Sync Status → 1` (synchronized)

For the data acquisition node (the slave node), the final state looks like this figure.



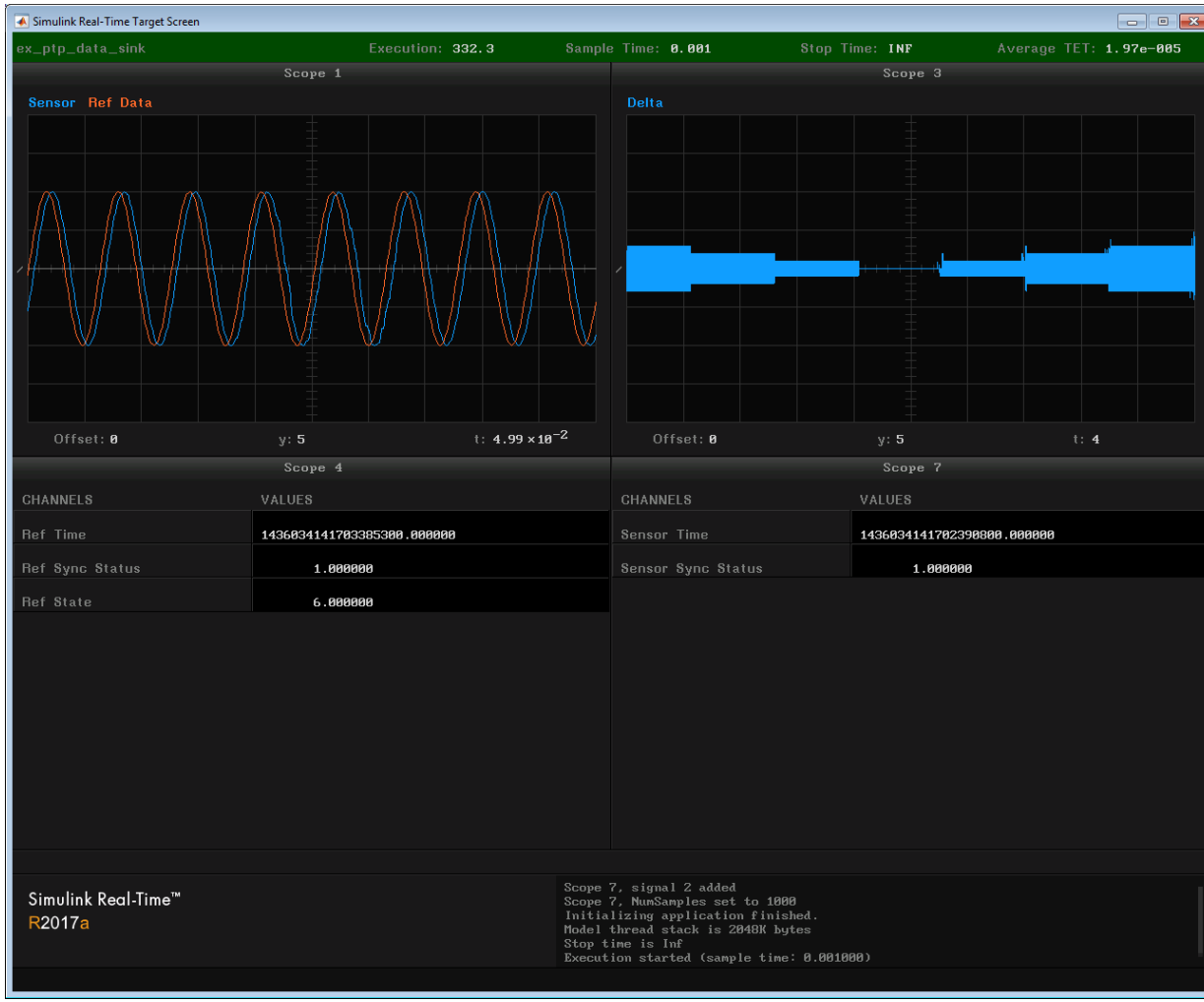
For the data analysis node (the master node), the final state looks like this figure.



For this example, the sensor and reference Sine Wave blocks are set to the same frequency and amplitude, but start at arbitrary times. The difference in start time causes a phase difference between the sine waves. The phase difference appears on the `Delta` scope as a waveform that settles to a constant amplitude.

The phase difference is constant because the IEEE 1588 Sync Execution blocks synchronize the kernel clocks on the two target computers. If you do not include these

blocks, the kernel clocks of the two target computers drift apart. As a result, the `Delta` waveform shows a beat frequency.



With the IEEE 1588 Sync Execution block, you can make measurements across multiple target computers at a synchronized time step. However, the kernel interrupt clock

controller can shorten some time steps up to 10% of the fundamental sample time, resulting in a CPU overload.

See Also

matlab: open_system(docpath(fullfile(docroot, 'toolbox', 'xpc', 'examples', 'ex_ptp_sync_src'))) | matlab: open_system(docpath(fullfile(docroot, 'toolbox', 'xpc', 'examples', 'ex_ptp_sync_sink'))) | matlab: open_system(docpath(fullfile(docroot, 'toolbox', 'xpc', 'examples', 'ex_ptp_data_src'))) | matlab: open_system(docpath(fullfile(docroot, 'toolbox', 'xpc', 'examples', 'ex_ptp_data_sink'))) | `IEEE 1588 Sync Execution` | `IEEE 1588 Ethernet` | `IEEE 1588 Read Parameter` | `IEEE 1588 Sync Status`

More About

- “Data Acquisition and Data Analysis Example Description” on page 21-18
- “IEEE® 1588™ Precision Time Protocol - Execution Synchronization”
- “Precision Time Protocol” on page 21-2

Data Acquisition and Data Analysis Example Description

This example features a data acquisition target computer that transmits timestamped sensor data to a second target computer. The second target computer processes the sensor data and displays the data and the difference between the sensor data and reference data. Both target computers connect to a development computer. The development computer runs Simulink and Simulink Real-Time Explorer to build, download, and run real-time applications on each target computer. The real-time applications use Precision Time Protocol (PTP) blocks to synchronize the PTP clocks and kernel clocks on each computer.

In this section...
"Data Acquisition Application" on page 21-18
"Data Analysis Application" on page 21-21

Data Acquisition Application

The data acquisition application is a PTP slave node that acquires data from a sensor, which a Sine Wave block represents. The application transmits the sensor data to the data analysis application.

Top-Level Model

The PTP initialization block and the blocks that create and transmit the Ethernet packet are in the top-level model:

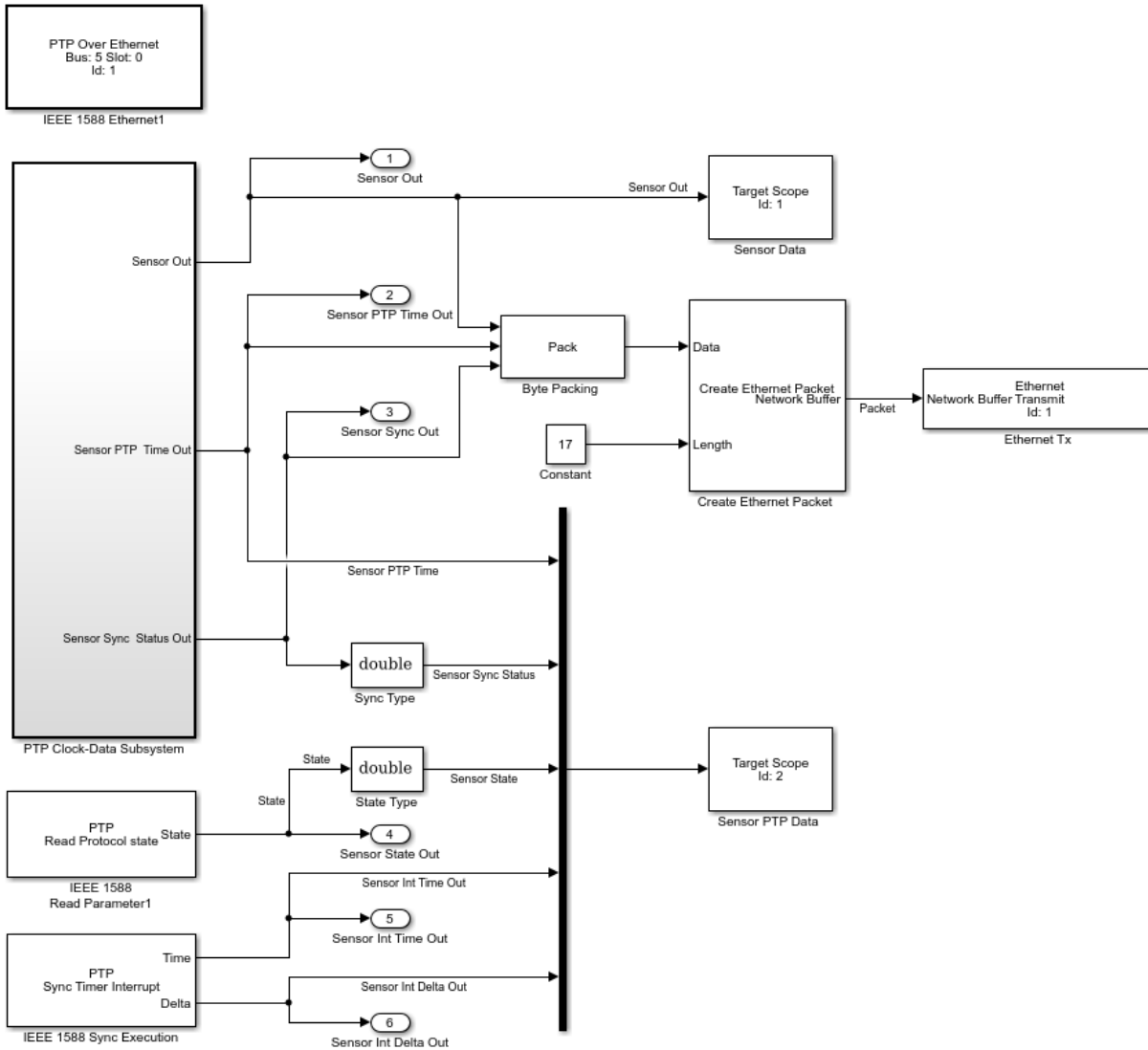
- IEEE 1588 Ethernet — Configures the PTP network card clock as a slave clock.
- IEEE 1588 Read Parameter — Shows when the PTP clock has been allocated as a slave clock (value 9). Configured as `Read Protocol state`.
- IEEE 1588 Sync Execution — Aligns the kernel clocks across multiple target computers. The block output shows the difference between the following times:
  - The PTP time at the real-time interrupt
  - The nearest PTP time that is a multiple of the fundamental sample time
- Byte Packing — Packs the sensor data, timestamp, and synchronization status into an Ethernet packet.

- Create Ethernet Packet — Addresses the Ethernet packet to the MAC address of the data analysis application.
- Ethernet Tx — Transmits the Ethernet packet to the data-analysis target computer.

  The Ethernet Tx block sends data packets through the same Ethernet connection as the PTP blocks use to send PTP messages. To distinguish data packets from PTP messages, the model assigns to the data packets Ethernet type `hex2dec('0010')`. This Ethernet type is different from the default Ethernet type of PTP packets (`hex2dec('88F7')`).
- Outport — For data logging purposes, the top-level model propagates the major signals to Outport blocks.

For debugging purposes, the top-level model includes two real-time Scope blocks:

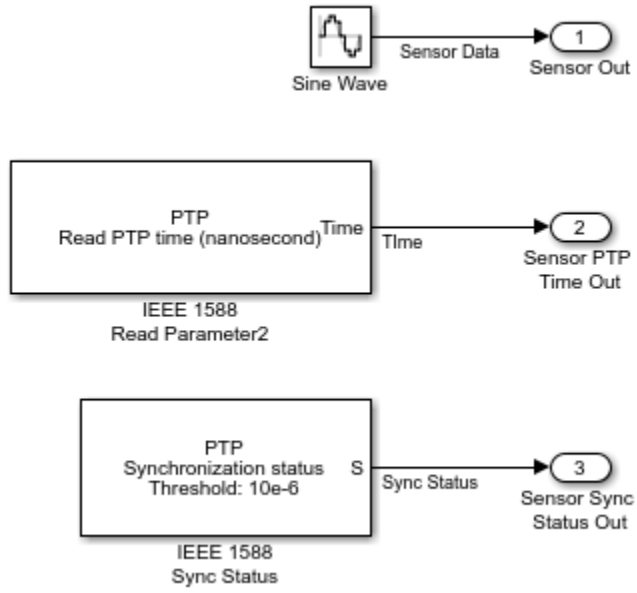
- `Sensor Data` — Displays the sensor data in a graphic scope.
- `Sensor PTP Data` — Displays the PTP time, PTP synchronization status, PTP state, and synchronization delta of the data acquisition model.


PTP Over Ethernet
Bus: 5 Slot: 0
Id: 1
IEEE 1588 Ethernet1
1
Sensor Out
Sensor Out
Target Scope
Id: 1
Sensor Data
Sensor Out
2
Sensor PTP Time Out
Pack
Byte Packing
Data
Create Ethernet Packet
Network Buffer
Length
Packet
Ethernet
Network Buffer Transmit
Id: 1
Ethernet Tx
3
Sensor Sync Out
Sensor PTP Time Out
17
Constant
Create Ethernet Packet
Sensor PTP Time
Sensor Sync Status Out
double
Sensor Sync Status
Sync Type
PTP Clock-Data Subsystem
State
double
Sensor State
State Type
Target Scope
Id: 2
Sensor PTP Data
State
PTP
Read Protocol state
State
4
Sensor State Out
IEEE 1588
Read Parameter1
Sensor Int Time Out
5
Sensor Int Time Out
Time
PTP
Sync Timer Interrupt
Delta
Sensor Int Delta Out
6
Sensor Int Delta Out
IEEE 1588 Sync Execution

Atomic Subsystem

The PTP timestamp must align as closely as possible with the data source. For better alignment, the model wraps the sensor data block and the lower-level PTP blocks in an atomic subsystem:

- Sine Wave — Represents sensor data.
- IEEE 1588 Read Parameter — Generates the timestamp, configured as `PTP Time (nanosecond)`.
- IEEE 1588 Sync Status — Generates the synchronization status. When the PTP clock is synchronized with the master PTP clock, the block output becomes `1`.



Data Analysis Application

The data analysis application is a PTP master node that gets sensor data from an emulator, a Sine Wave block. The application gets reference data from an emulator, a Sine Wave block, and sensor data from an Ethernet Rx block. The application calculates the difference between the reference data and the sensor data.

Top-Level Model

The PTP initialization block and the blocks that receive and process the data are in the top-level model:

- IEEE 1588 Ethernet — Configures the PTP network card clock as a master clock.
- IEEE 1588 Read Parameter — Shows when the PTP clock has been allocated as a master clock (value `6`). Configured as `Read Protocol state`.
- IEEE 1588 Sync Execution — Aligns the kernel clocks across multiple target computers. The block output shows the difference between the following times:
  - The PTP time at the real-time interrupt
  - The nearest PTP time that is a multiple of the fundamental sample time
- Extract Ethernet Packet — Extracts the Ethernet packet that is carrying the sensor data.
- Byte Unpacking — Unpacks the sensor data, timestamp, and synchronization status from the Ethernet packet.
- Sum — Calculates the difference between the sensor data and the reference data.

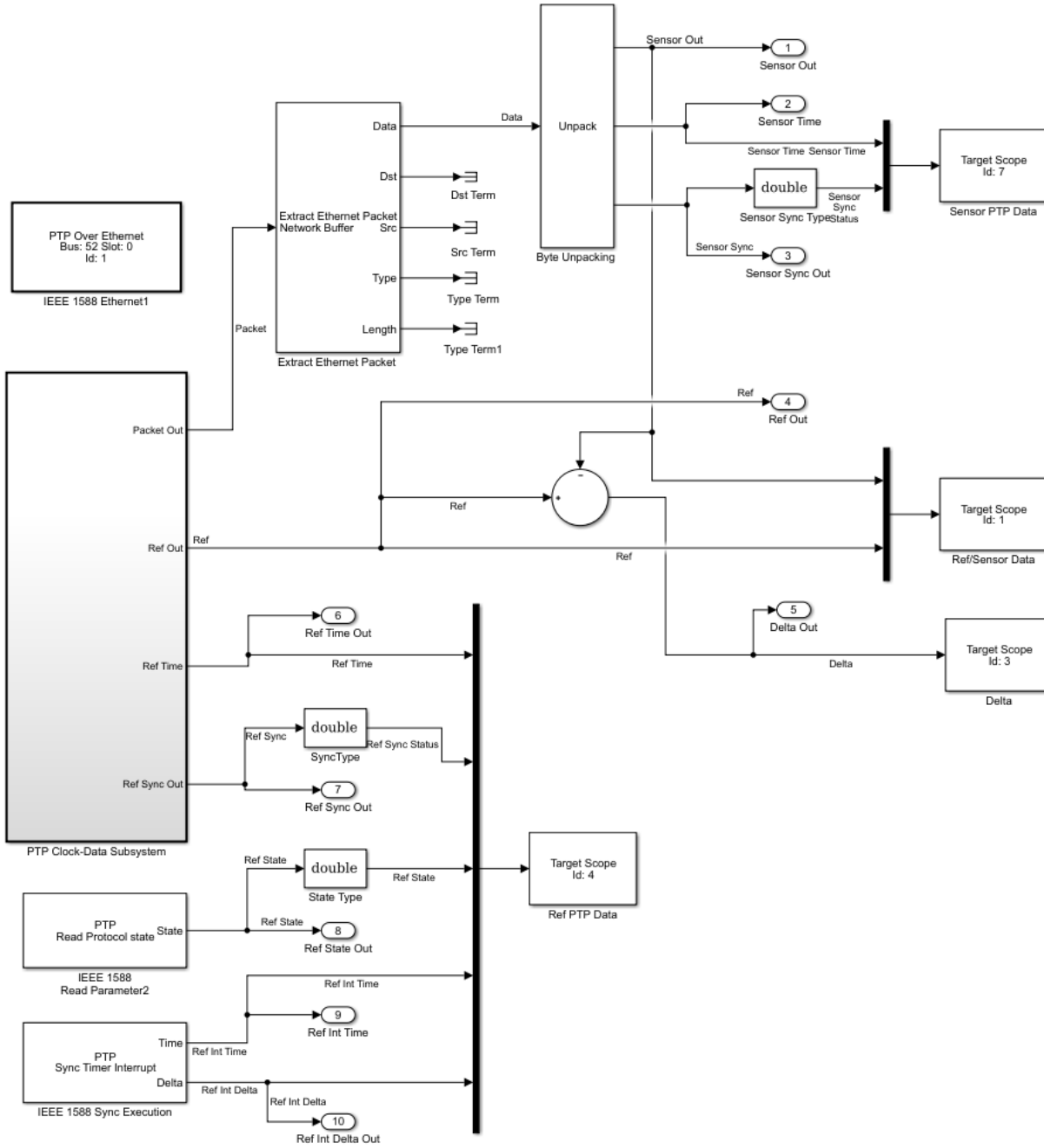
  The Sum block provides input data for further processing. For example, you can plot the sensor data, reference data, and difference against the timestamp to assess the real-time behavior. You can also feed the difference data back through a control system to change an actuator setting at the data acquisition site.
- Outport — For data logging purposes, the top-level model propagates the major signals to Outport blocks.

For debugging purposes, the top-level model includes four real-time Scope blocks:

- `Ref/Sensor Data` — Displays the reference data and the sensor data together in a graphic scope.
- `Delta` — Displays the difference between the reference data and the sensor data in a graphic scope.

  The `Delta` scope is configured with a long sample time. It captures long-period differences between the sensor and reference data. If the frequency, phase, and amplitude differences are constant, the scope displays a rectangular area. If the differences are periodic, the scope displays a beat frequency.
- `Ref PTP Data` — Displays the PTP time, PTP synchronization status, PTP state, and synchronization delta of the data analysis model.

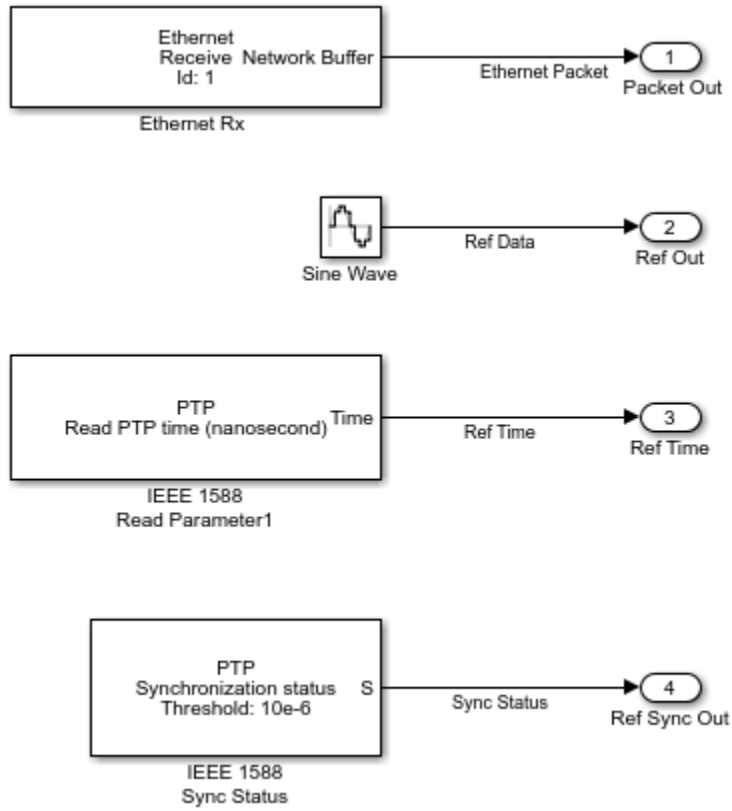
- `Sensor PTP Data` — Displays the PTP time, PTP synchronization status, and synchronization delta of the data acquisition model.


PTP Over Ethernet
Bus: 52 Slot: 0
Id: 1
IEEE 1588 Ethernet1
Extract Ethernet Packet
Network Buffer
Data
Dst
Src
Type
Length
Dst Term
Src Term
Type Term
Type Term1
Extract Ethernet Packet
Packet
Data
Unpack
Byte Unpacking
Sensor Out
1
Sensor Out
2
Sensor Time
Sensor Time Sensor Time
double
Sensor Sync Type
Sensor Sync Status
Sensor Sync
3
Sensor Sync Out
Target Scope
Id: 7
Sensor PTP Data
Packet Out
Ref Out
Ref Time
Ref Sync Out
PTP Clock-Data Subsystem
Ref
4
Ref Out
Ref
Ref
Target Scope
Id: 1
Ref/Sensor Data
6
Ref Time Out
Ref Time
Ref Time
5
Delta Out
Delta
Target Scope
Id: 3
Delta
Ref Sync
double
SyncType
Ref Sync Status
Ref Sync
7
Ref Sync Out
Ref State
double
State Type
Ref State
Target Scope
Id: 4
Ref PTP Data
PTP
Read Protocol state
State
IEEE 1588
Read Parameter2
Ref State
8
Ref State Out
Ref Int Time
9
Ref Int Time
Time
PTP
Sync Timer Interrupt
Delta
IEEE 1588 Sync Execution
Ref Int Time
Ref Int Delta
Ref Int Delta
10
Ref Int Delta Out

Atomic Subsystem

The PTP timestamp must align as closely as possible with the Ethernet receiver. For better alignment, the model wraps the blocks representing the reference data source and the lower-level PTP blocks in an atomic subsystem:

- Sine Wave — Represents reference data.
- IEEE 1588 Read Parameter — Generates the timestamp, configured as `PTP Time (nanosecond)`.
- IEEE 1588 Sync Status — Generates the synchronization status. When the PTP clock is synchronized with the master PTP clock, the block output becomes `1`.
- Ethernet Rx — Receives sensor data from the acquisition target computer. The configured block filters out all packets except packets of Ethernet type `hex2dec('0010')`. The default Ethernet type of PTP packets is `hex2dec('88F7')`.



See Also

matlab: open_system(docpath(fullfile(docroot, 'toolbox', 'xpc', 'examples', 'ex_ptp_sync_src'))) | matlab: open_system(docpath(fullfile(docroot, 'toolbox', 'xpc', 'examples', 'ex_ptp_sync_sink'))) | `IEEE 1588 Sync Execution` | `IEEE 1588 Ethernet` | `IEEE 1588 Read Parameter` | `IEEE 1588 Sync Status`

More About

- “Synchronize Timestamps Across Data-Gathering Network” on page 21-5
- “IEEE® 1588™ Precision Time Protocol - Execution Synchronization”

Precision Time Protocol Troubleshooting

To troubleshoot your model, first familiarize yourself with the PTP standard, and then with the specialized requirements of the Simulink Real-Time implementation.

In this section...

Modeling

- Configure all PTP nodes in a network with the same **Delay measurement mechanism**. If you configure a slave node with a different setting from the master node, the slave node enters the `FAULTY` state
- Configure all PTP nodes in a network with the same **Timescale** or **Arbitrary timescale epoch** value. If you configure the master and slave nodes with differing timescales, the representation of time in time-of-day format differs for the two nodes.
- Configure all nodes in the PTP networks with the same **Announce interval** and **Announce receipt timeout**. Differing values of these parameters in a PTP network can lead to unpredictable behavior.
- Avoid using inherited sample time everywhere in your model. Inherited sample time occurs throughout your model when you make the following settings:

  - **Fixed step size** → `auto` in the Configuration Parameters dialog box
  - **Sample time** → `-1` in all of the blocks of your model.

  The sample time that Simulink calculates can be greater than the PTP message transmission intervals, resulting in an unusable model.
- The PTP configuration subsystems include configuration blocks for the associated transport protocol. If you use a separate Ethernet card for data transmission, include a separate network configuration block. Assign it a **Device ID** different from the one already in use by the PTP configuration block. Multiple network configuration blocks with the same **Device ID** cause a build error.
- The PTP Over Ethernet block creates PTP messages with **Ether type** set to `hex2dec('88F7')`. To use the same Ethernet card for PTP as for data transmission:

  - In the Create Ethernet Packet block, set **Ether type** to a nonzero value that is different from `hex2dec('88F7')` (for example, `hex2dec('0010')`.
  - In the Ethernet Rx block, set **Filter criteria** to `Specify types to match`. Set **Receive these types** to the value that you set in the Create Ethernet Packet block (for example, `[hex2dec('0010')]`).
- If you include more than one slave node in the network, configure the master node to use the standard PTP multicast address for transmitting messages. The master node must transmit the same synchronization message to all the slaves.
- Using the IEEE 1588 Sync Execution block to make measurements across multiple target computers at the same simulation step can lead to a CPU overload. Also, the kernel interrupt clock controller can shorten some time steps up to 10% of the model fundamental sample time.

  If you get CPU overloads, consider decreasing the value of the **Proportional gain** parameter of the IEEE 1588 Sync Execution block or increasing the sample time of your real-time application.
- If you use the IEEE 1588 Sync Execution block in your model, configuring EtherCAT distributed clocks in master shift mode in the same model produces a build error. To include IEEE 1588 synchronized execution and EtherCAT distributed clocks in the same model, use EtherCAT bus shift mode.

Accuracy

- The synchronization accuracy depends upon the value of **Sync interval**. The smaller the value, the more accurate the synchronization. If your model fails to meet your required synchronization accuracy, try decreasing the value of **Sync interval**.
- You can use IEEE 1588 Sync Execution block to synchronize two PTP models with differing fundamental sample times. Their execution is synchronous at a PTP time equal to the least common multiple of the two rates.

Faulty States

- When a slave node enters the `FAULTY` state, look for one of the following conditions:
  - The slave node is configured with a different **Delay measurement mechanism** setting from the master node setting.
  - The slave node model sample time setting is greater than the master node **Sync interval** setting.

  - The slave node **Announce interval** setting is shorter than the master node **Announce interval** setting.
  - The slave is not receiving a response from the master to delay request messages sent by the slave. This behavior occurs, for example, if the slave node is configured to use a delay measurement mechanism setting different from the master node setting.
- If the master node fails to read a required timestamp from the Ethernet card due to contention for the timestamp register, the transmission can fail. After a master node fails five consecutive times to transmit a `Follow_Up`, `Delay_Resp`, `Pdelay_Resp`, or `Pdelay_Resp_Follow_Up` message to its slave nodes, it enters the `FAULTY` state. Try the following:

  - Reduce the number of slave nodes in the network.
  - Shorten the sample time for the subsystem that represents the master node. The master node cycles through the slave messages faster and reads the timestamp register more often.
  - Increase the **Min delay or pdelay request interval** of the slave nodes. The slave nodes generate messages less often.
  - Connect a peer-to-peer transparent PTP clock between the master and slave nodes. Set **Delay measurement mechanism** to `Peer-delay` for all of the nodes. The peer-to-peer transparent PTP clock has a separate timestamp register for each port, taking the load off the master node.

    For more information, see IEEE Std 1588-2008 Clause 10.

More About

- “Prerequisites, Limitations, and Unsupported Features” on page 21-30
- “Synchronize Timestamps Across Data-Gathering Network” on page 21-5
- “IEEE® 1588™ Precision Time Protocol - Execution Synchronization”

External Websites

- standards.ieee.org

Prerequisites, Limitations, and Unsupported Features

The Simulink Real-Time implementation of PTP enforces specific requirements and limitations.

In this section...

Prerequisites

- PTP functionality is available only with a Speedgoat target computer. If you have not installed the Speedgoat library, attempting to build a real-time application with PTP causes a build error.
- Simulink Real-Time supports PTP only with Intel 82574 Ethernet cards. To check that you have the required card, start your target computer. In the Command Window, type:

  ```
  tg = slrt;
  getPCIInfo(tg, 'Ethernet')
  ```

  Check that you see an entry like this entry:

  ```
  Intel                    82574L
       Bus 5, Slot 0, IRQ 10
       Ethernet controller
       VendorID 0x8086, DeviceID 0x10d3, SubVendorID 0x15bd,
             SubDeviceID 0x100a
       Released in: R2010a
       Notes: Intel 8254x Gigabit Ethernet series
  ```

Limitations

- The PTP network card clock acts as PTP clock. Only one clock is allowed per node.
- Run the model in `Real-Time` execution mode, not in `Freerun` mode or driven by an external interrupt. In the latter two cases, the PTP message transmission intervals can violate the PTP standard.

- You can include only one PTP configuration block in a model. You can run only one real-time application on a target computer. If you have installed multiple PTP Ethernet cards on your target computer, you can use only one of them for PTP at a time. You can use the other PTP Ethernet cards for non-PTP purposes.
- The PTP message transmission intervals (**Announce interval**, **Sync interval**, and **Min delay or pdelay request interval**) must be greater than the block sample time. Too small a message transmission interval causes a model update error.
- Simulink Real-Time can transmit PTP messages only at a multiple of the block sample time. If a transmission interval is not a multiple of the block sample time, PTP transmits the messages at the nearest multiple to the specified transmission time. As a best practice, specify all transmission intervals as integral multiples of the block sample time.
- The specification requires that a PTP node issue messages within ±30% of the message transmission intervals at least 90% of the time. To meet this requirement, specify message transmission intervals (**Announce interval**, **Sync interval**, and **Min delay or pdelay request interval**) at least three times the base sample time.
- The following factors limit accuracy:

  - Network protocol stack delay fluctuation
  - Network technology component delay fluctuations (switches, routers)
  - Clock timestamp accuracy
  - Clock oscillator stability

  Choose components that minimize these factors. For example, you can use a transparent or boundary PTP clock to increase synchronization accuracy.

  - A transparent PTP clock tracks the amount of time a PTP message takes to go through the device. It passes that information to nodes receiving the message.
  - A boundary PTP clock has multiple PTP ports that can act as a master clock or a slave clock.
- Some systems require a PTP time source that is traceable to an International Atomic Time (TAI) clock, such as a GPS signal. To support traceability, acquire a third-party grandmaster PTP clock that provides this capability. In that case, a Simulink Real-Time target computer running PTP acts only as a slave clock.

Unsupported Features

- Simulink Real-Time supports only PTP version 2, as defined in IEEE Std 1588-2008. If a Simulink Real-Time PTP node receives a PTP version 1 message as defined in IEEE Std 1588-2002, it ignores it.
- The Simulink Real-Time implementation of PTP does not support the following functionality defined in IEEE Std 1588-2008:
  - PTP variance computation, as described in IEEE Std 1588-2008 Clause 7.6.3.
  - PTP nodes configured as one-step PTP clocks, as described in IEEE Std 1588-2008 Clause 3.1.21.

    You can configure the master node as a two-step clock only, as described in IEEE Std 1588-2008 Clause 3.1.47.
  - PTP management messages, as described in IEEE Std 1588-2008 Clause 15.

    A Simulink Real-Time PTP node cannot transmit a PTP management message. When a Simulink Real-Time PTP node receives a PTP management message, it ignores it.
  - PTP signaling messages, as described in IEEE Std 1588-2008 Clause 13.12.

    A Simulink Real-Time PTP node cannot transmit a PTP signaling message. When a Simulink Real-Time PTP node receives a PTP signaling message, it ignores it.
  - Optional features, as described in IEEE Std 1588-2008 Clause 16 and Clause 17.

Related Examples

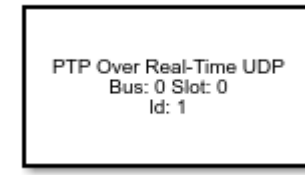
- “Precision Time Protocol Troubleshooting” on page 21-27

22

Precision Time Protocol Blocks

IEEE 1588 Real-Time UDP

Execute IEEE 1588 Precision Time Protocol
Library: IEEE 1588



Description

IEEE 1588 Real-Time UDP executes the PTP protocol, using UDP to send and receive the protocol messages. The block communicates with the corresponding blocks on the other target computers, determines the time offset that synchronizes them, and adjusts the time offset.

Parameters

General

`Device ID` — Ethernet board identifier
1 - 8

From the list, select a unique number to identify the Ethernet board.

`PCI bus` — PCI bus number of Ethernet card
0 (default) | integer

Enter the PCI bus number for the Ethernet card.

`PCI slot` — PCI slot number of Ethernet card
0 (default) | integer

Enter the PCI slot number for the Ethernet card.

`Sample time (-1 for inherited)` — Sample time of block
-1 (default) | numeric

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

Network Parameters

`IP address of port` — IP address of PTP clock board
x.x.x.x

IP address of the Ethernet board, or node, carrying the PTP clock.

The addresses `0.0.0.0` and `255.255.255.255` are invalid IP addresses.

`Subnet mask` — Subnet mask for interface
`255.255.255.0` (default) | *x.x.x.x*

Mask that designates a logical subdivision of a network.

`Gateway` — IP address for gateway interface
`0.0.0.0` (default) | *x.x.x.x*

The gateway must be within the network.

To indicate "no gateway used," enter `0.0.0.0` (the default). The address `255.255.255.255` is an invalid gateway IP address.

`Source IP address of receive packets (set to 0.0.0.0 to receive all)` — IP address in receive blocks
`0.0.0.0` (default) | *x.x.x.x*

IP address in UDP Receive blocks. The default value (`0.0.0.0`) specifies that the node is to receive all packets sent to the ports assigned to PTP messages (ports `319` and `320`).

Use a specific value for one-to-one communication. If the node is a master PTP clock node, use a specific value only if exactly one slave is connected to the master clock node.

The address `255.255.255.255` is an invalid IP address.

`Destination IP address of transmit packets` — IP address in transmit blocks
`Standard PTP Multicast (224:0:1:129, 224:0:0:107)` (default) | *x.x.x.x*

IP address in UDP Transmit blocks. Specifies the IP address of the other PTP computers or devices to which to send the PTP packets. Select one of:

- `Standard PTP Multicast (224:0:1:129, 224:0:0:107)` (default) — Default standard multicast IP address assigned to PTP. If you select this option, the PTP packets are broadcast to all computers listening on the PTP ports (ports `319` and `320`). The destination IP addresses are:
  - `224.0.1.129` for non-peer-delay measurement mechanism messages (`Announce`, `Sync`, `Follow_up`, `Delay_Req`, `Delay_Resp`)
  - `224.0.0.107` for peer-delay measurement mechanism messages (`Pdelay_Req`, `Pdelay_Resp`, `Pdelay_Resp_Follow_up`)
- `Specify` — Explicitly specify the destination IP address.

Dependency

Selecting `Specify` makes the **Specify destination IP address** parameter visible.

`Specify Destination IP address` — IP address of transmit packets
`255.255.255.255` (default) | *`x.x.x.x`*

The default value (`255.255.255.255`) specifies that the node is to broadcast the packets to all listening nodes of the network. Use a specific value for one-to-one communication. If the node is a master PTP clock node, use a specific value only if exactly one slave is connected to the master clock node.

Dependency

To make this parameter visible, set **Destination IP address of transmit packets** to `Specify`.

Clock Parameters

`Timescale (epoch)` — Origin point of the PTP timescale
`PTP (1970-01-01)` (default) | `GPS (1980-06-01)` | `NTP (1900-01-01)` | `Specify`

Specify the origin point of the PTP timescale. Select one of:

- `PTP (1970-01-01)` — Precision Time Protocol standard epoch, starting January 1, 1970.

- `GPS (1980-06-01)` — Global Positioning System standard epoch, starting June 1, 1980.
- `NTP (1900-01-01)` — Network Time Protocol standard epoch, starting January 1, 1900.
- `Specify` — Explicitly specify the timescale epoch.

Dependency

Selecting `Specify` makes the **Arbitrary timescale epoch (yyyy mm dd hh)** parameter visible.

`Arbitrary timescale epoch [yyyy mm dd hh]` — Explicit origin point for PTP timescale

`[1970 01 01 00]` (default) | `[yyyy mm dd hh]`

Specify the origin point for the PTP timescale, in year, month, day, and hour.

Dependency

To make this parameter visible, set **Timescale (epoch)** to `Specify`.

`Delay measurement mechanism` — Method of measuring link delays

`Request-response` (default) | `Peer-delay`

Specify the method of measuring link delays. Configure all PTP network nodes to use the same link delay measurement mechanism.

For more information, see IEEE Std Clause 7.5.4.

`Slave only` — Node that cannot be allocated as master PTP clock

`off` (default) | `on`

When you select this check box, you cannot allocate the PTP Ethernet card that this block represents as a master PTP clock.

In **Slave only** mode, the values of the advanced parameters (**Priority 1**, **Clock class**, **Clock accuracy**, and **Priority 2**) are set to their highest values. When the parameters have these settings, all of the other nodes must have the same configuration. If a node

has a different configuration, the Best Master Clock Algorithm (BMCA) cannot choose the node as best master clock. If the BMCA selects a **Slave only** node as best clock, the node remains in the `LISTENING` state.

`Show advanced configuration parameters` — Enable low-level PTP configuration parameters
`off` (default) | `on`

For more information, see IEEE Std 1588-2008.

Dependency

Selecting this check box makes advanced configuration parameters visible: **Domain number**, **Current UTC offset**, **Priority 1**, **Clock class**, **Clock accuracy**, and **Priority 2**.

`Domain number` — Domain number of PTP network
0 (default) | 0–127

Specify the domain number of the PTP network to which the node belongs.

A Simulink Real-Time PTP node can belong to only one PTP domain at a given time. If the node receives a PTP message with a different domain number, it ignores it. For more information, see IEEE Std Clause 7.1.

Dependency

To make this parameter visible, select the **Show advanced configuration parameters** check box.

`Current UTC offset` — Current offset from Coordinated Universal Time
35 (default) | integer

The current UTC offset, in seconds.

If you specify a nonzero value, that value is considered valid. The `UTCOffsetValid` flag is set to `true`. Otherwise, the flag is set to `false`. For more information, see IEEE Std 1588-2008 Clause 7.2.3.

Dependency

To make this parameter visible, select the **Show advanced configuration parameters** check box.

`Priority 1` — Priority of PTP node

128 (default) | 0–255

Specify an integer value encoding the priority of the PTP node in the network. When the value is `0`, the node has the highest priority. When it is `255`, the node has the lowest priority.

To assess the quality of two PTP clocks, the Best Master Clock Algorithm compares the following parameters, in order:

1. **Priority 1**
2. **Clock class**
3. **Clock accuracy**
4. **Priority 2**

For each parameter, the algorithm selects the clock with the smaller value as the best clock. If all four parameters are equal for both clocks, the algorithm compares the MAC addresses of the nodes.

For more information, see IEEE Std 1588-2008 Clause 7.6.2.2.

Dependency

To make this parameter visible, select the **Show advanced configuration parameters** check box.

`Clock class` — Clock class designator

248 (default) | 0–255

Specify a nonreserved integer value. If **Clock class** is less than `128`, the node cannot enter the `SLAVE` state. If **Clock class** is less than `128` and the node is not selected as the best clock, the node enters the `PASSIVE` state.

If you specify a reserved integer value, the block produces an error during model update. For more information, see IEEE Std 1588-2008 Clause 7.6.2.4. For a list of reserved and nonreserved **Clock class** values, see IEEE Std 1588-2008 Table 5.

Dependency

To make this parameter visible, select the **Show advanced configuration parameters** check box.

`Clock accuracy` — Accuracy code for clock
`hex2dec('FE')` (default) | `0–254`

Specify a nonreserved integer value. For more information, see IEEE Std 1588-2008 Clause 7.6.2.5. For a list of reserved and nonreserved **Clock accuracy** values, see IEEE Std 1588-2008 Table 6.

Dependency

To make this parameter visible, select the **Show advanced configuration parameters** check box.

`Priority 2` — Secondary priority of PTP node
`128` (default) | `0–255`

Specify secondary priority of PTP node. When the value is `0`, the node has the highest priority. When it is `255`, the node has the lowest priority.

For more information, see IEEE Std 1588-2008 Clause 7.6.2.3.

Dependency

To make this parameter visible, select the **Show advanced configuration parameters** check box.

Time Intervals

`Announce interval (second)` — Period of master node Announce message
`2` (default) | numeric

The period, in seconds, of an Announce message transmitted by a node in master state.

For more information, see IEEE Std 1588-2008 Clause 9.5.8.

`Sync interval (second)` — Period of master node Sync message
`0.1` (default) | numeric

The period, in seconds, of a Sync message transmitted by a node in master state.

For more information, see IEEE Std 1588-2008 Clause 9.5.9.

`Min delay or pdelay request interval (second)` — Period of slave node request message
`0.1` (default) | numeric

Period of delay request message or of peer-delay request message transmitted by a node in the slave state. When the delay measurement mechanism is `Request-response`, the block transmits delay request messages. When the mechanism is `Peer-delay`, it transmits peer-delay request messages.

For more information, see IEEE Std 1588-2008 Clauses 9.5.11 and 9.5.13.

`Announce receipt timeout (in announce intervals)` — Timeout for Announce message response
3 (default) | integer

Specifies the number of announce intervals a node not in the master state has to wait without receiving an announce message. After the timeout passes, the node enters the master state.

For more information, see IEEE Std 1588-2008 Clause 9.2.6.11.

Model Examples

See Also

See Also
`SimulinkRealTime.target.getPCIInfo`

Topics

"Synchronize Timestamps Across Data-Gathering Network" on page 21-5

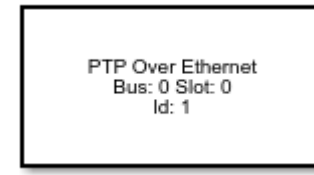
External Websites

standards.ieee.org

Introduced in R2015b

IEEE 1588 Ethernet

Execute IEEE 1588 Precision Time Protocol
Library: IEEE 1588



Description

IEEE 1588 Ethernet executes the PTP protocol, using raw Ethernet to send and receive the protocol messages. The block communicates with the corresponding blocks on the other target computers, determines the time offset that synchronizes them, and adjusts the time offset.

Parameters

General

`Device ID` — Ethernet board identifier
1 - 8

From the list, select a unique number to identify the Ethernet board.

`PCI bus` — PCI bus number of Ethernet card
0 (default) | integer

Enter the PCI bus number for the Ethernet card.

`PCI slot` — PCI slot number of Ethernet card
0 (default) | integer

Enter the PCI slot number for the Ethernet card.

`Sample time (-1 for inherited)` — Sample time of block
-1 (default) | numeric

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

Network Parameters

`Source MAC address` — MAC address of Ethernet card for message source
`EEPROM` (default) | `Specify`

Source MAC address in Ethernet transport protocol. From the list, select:

- `EEPROM` — Allow the block to get the Ethernet card MAC address that is built into the Ethernet card. Use this option if you use separate Ethernet connections to transmit data and to synchronize PTP clocks.
- `Specify` — Explicitly specify the source MAC address. Use this option if both of these conditions are true:
  - You use the same Ethernet connection to transmit data as you use to synchronize PTP clocks.
  - You do not know the built-in MAC address of the Ethernet card.

Dependency

Selecting `Specify` makes the **Specify source MAC address** parameter visible.

`Specify source MAC address` — Explicit MAC address of Ethernet card for message source
`00:00:00:00:00:00` (default) | *`xx:xx:xx:xx:xx:xx`*

Enter the MAC address for the Ethernet card. Use the MAC address that is built into the Ethernet card or an arbitrary MAC address that is unique within the PTP network. Do not use one of the standard PTP multicast MAC addresses.

Dependency

To make this parameter visible, set **Source MAC address** to `Specify`.

`Destination MAC address` — MAC address of Ethernet card for message destination
`Standard PTP Multicast (01:1B:19:00:00:00, 01:80:C2:00:00:0E)` (default) | `Specify`

Destination MAC address in Ethernet transport protocol. Select one of:

- `Standard PTP Multicast (01:1B:19:00:00:00, 01:80:C2:00:00:0E)` — Default multicast MAC address assigned to the PTP protocol. If you select this option, the destination MAC addresses are:
  - `01:1B:19:00:00:00` for non-peer-delay measurement mechanism messages (`Announce`, `Sync`, `Follow_up`, `Delay_Req`, `Delay_Resp`)
  - `01:80:C2:00:00:0E` for peer-delay measurement mechanism messages (`Pdelay_Req`, `Pdelay_Resp`, `Pdelay_Resp_Follow_up`)
- `Specify` — Explicitly specify the destination MAC address.

You do not have to specify a source MAC address. The block uses the unique MAC address of the PTP Ethernet card.

Dependency

Selecting `Specify` makes the **Specify destination MAC address** parameter visible.

`Specify destination MAC address` — Explicit MAC address of Ethernet card for message destination

`00:00:00:00:00:00` (default) | *`xx:xx:xx:xx:xx:xx`*

Specify a MAC address for the message destination. Use this option for **Slave only** nodes. Specify the MAC address of the master node Ethernet card. The master node uses the standard PTP multicast MAC address to transmit messages to all slave nodes.

Dependency

To make this parameter visible, set **Destination MAC address** to `Specify`.

Clock Parameters

`Timescale (epoch)` — Origin point of the PTP timescale

`PTP (1970-01-01)` (default) | `GPS (1980-06-01)` | `NTP (1900-01-01)` | `Specify`

Specify the origin point of the PTP timescale. Select one of:

- `PTP (1970-01-01)` — Precision Time Protocol standard epoch, starting January 1, 1970.
- `GPS (1980-06-01)` — Global Positioning System standard epoch, starting June 1, 1980.
- `NTP (1900-01-01)` — Network Time Protocol standard epoch, starting January 1, 1900.
- `Specify` — Explicitly specify the timescale epoch.

Dependency

Selecting `Specify` makes the **Arbitrary timescale epoch (yyyy mm dd hh)** parameter visible.

`Arbitrary timescale epoch [yyyy mm dd hh]` — Explicit origin point for PTP timescale
`[1970 01 01 00]` (default) | `[yyyy mm dd hh]`

Specify the origin point for the PTP timescale, in year, month, day, and hour.

Dependency

To make this parameter visible, set **Timescale (epoch)** to `Specify`.

`Delay measurement mechanism` — Method of measuring link delays
`Request-response` (default) | `Peer-delay`

Specify the method of measuring link delays. Configure all PTP network nodes to use the same link delay measurement mechanism.

For more information, see IEEE Std Clause 7.5.4.

`Slave only` — Node that cannot be allocated as master PTP clock
`off` (default) | `on`

When you select this check box, you cannot allocate the PTP Ethernet card that this block represents as a master PTP clock.

In **Slave only** mode, the values of the advanced parameters (**Priority 1**, **Clock class**, **Clock accuracy**, and **Priority 2**) are set to their highest values. When the parameters

have these settings, all of the other nodes must have the same configuration. If a node has a different configuration, the Best Master Clock Algorithm (BMCA) cannot choose the node as best master clock. If the BMCA selects a **Slave only** node as best clock, the node remains in the `LISTENING` state.

`Show advanced configuration parameters` — Enable low-level PTP configuration parameters

`off` (default) | `on`

For more information, see IEEE Std 1588-2008.

Dependency

Selecting this check box makes advanced configuration parameters visible: **Domain number**, **Current UTC offset**, **Priority 1**, **Clock class**, **Clock accuracy**, and **Priority 2**.

`Domain number` — Domain number of PTP network

0 (default) | 0–127

Specify the domain number of the PTP network to which the node belongs.

A Simulink Real-Time PTP node can belong to only one PTP domain at a given time. If the node receives a PTP message with a different domain number, it ignores it. For more information, see IEEE Std Clause 7.1.

Dependency

To make this parameter visible, select the **Show advanced configuration parameters** check box.

`Current UTC offset` — Current offset from Coordinated Universal Time

35 (default) | integer

The current UTC offset, in seconds.

If you specify a nonzero value, that value is considered valid. The `UTCOffsetValid` flag is set to `true`. Otherwise, the flag is set to `false`. For more information, see IEEE Std 1588-2008 Clause 7.2.3.

Dependency

To make this parameter visible, select the **Show advanced configuration parameters** check box.

`Priority 1` — Priority of PTP node

128 (default) | 0–255

Specify an integer value encoding the priority of the PTP node in the network. When the value is `0`, the node has the highest priority. When it is `255`, the node has the lowest priority.

To assess the quality of two PTP clocks, the Best Master Clock Algorithm compares the following parameters, in order:

1. **Priority 1**
2. **Clock class**
3. **Clock accuracy**
4. **Priority 2**

For each parameter, the algorithm selects the clock with the smaller value as the best clock. If all four parameters are equal for both clocks, the algorithm compares the MAC addresses of the nodes.

For more information, see IEEE Std 1588-2008 Clause 7.6.2.2.

Dependency

To make this parameter visible, select the **Show advanced configuration parameters** check box.

`Clock class` — Clock class designator

248 (default) | 0–255

Specify a nonreserved integer value. If **Clock class** is less than `128`, the node cannot enter the `SLAVE` state. If **Clock class** is less than `128` and the node is not selected as the best clock, the node enters the `PASSIVE` state.

If you specify a reserved integer value, the block produces an error during model update. For more information, see IEEE Std 1588-2008 Clause 7.6.2.4. For a list of reserved and nonreserved **Clock class** values, see IEEE Std 1588-2008 Table 5.

Dependency

To make this parameter visible, select the **Show advanced configuration parameters** check box.

`Clock accuracy` — Accuracy code for clock

`hex2dec('FE')` (default) | `0–254`

Specify a nonreserved integer value. For more information, see IEEE Std 1588-2008 Clause 7.6.2.5. For a list of reserved and nonreserved **Clock accuracy** values, see IEEE Std 1588-2008 Table 6.

Dependency

To make this parameter visible, select the **Show advanced configuration parameters** check box.

`Priority 2` — Secondary priority of PTP node

`128` (default) | `0–255`

Specify secondary priority of PTP node. When the value is `0`, the node has the highest priority. When it is `255`, the node has the lowest priority.

For more information, see IEEE Std 1588-2008 Clause 7.6.2.3.

Dependency

To make this parameter visible, select the **Show advanced configuration parameters** check box.

Time Intervals

`Announce interval (second)` — Period of master node Announce message

`2` (default) | numeric

The period, in seconds, of an Announce message transmitted by a node in master state.

For more information, see IEEE Std 1588-2008 Clause 9.5.8.

`Sync interval (second)` — Period of master node Sync message
`0.1` (default) | numeric

The period, in seconds, of a Sync message transmitted by a node in master state.

For more information, see IEEE Std 1588-2008 Clause 9.5.9.

`Min delay or pdelay request interval (second)` — Period of slave node request message
`0.1` (default) | numeric

Period of delay request message or of peer-delay request message transmitted by a node in the slave state. When the delay measurement mechanism is `Request-response`, the block transmits delay request messages. When the mechanism is `Peer-delay`, it transmits peer-delay request messages.

For more information, see IEEE Std 1588-2008 Clauses 9.5.11 and 9.5.13.

`Announce receipt timeout (in announce intervals)` — Timeout for Announce message response
3 (default) | integer

Specifies the number of announce intervals a node not in the master state has to wait without receiving an announce message. After the timeout passes, the node enters the master state.

For more information, see IEEE Std 1588-2008 Clause 9.2.6.11.

Model Examples

See Also

Topics

"Synchronize Timestamps Across Data-Gathering Network" on page 21-5

External Websites

standards.ieee.org

Introduced in R2015b

IEEE 1588 Read Parameter

Output Precision Time Protocol status parameter value

Library

Simulink Real-Time Library for Precision Time Protocol

Description

Read the parameter that you select and send its value to the block output. The block output name changes based on the parameter that you select.

Block Outputs

Name	Description
Time	Current number of nanoseconds (double), counting from the beginning of the epoch. When you select the **Time at block start** check box, the value is measured at the beginning of block execution. When you clear the **Time at block start** check box, the value is measured at the end of block execution. When **Parameter to read** is `PTP time (nanosecond)`, output Time is visible.
Date	Current time in time-of-day format. The value is a vector of size `8`, data type `uint16`, containing: year, month (1–12), day of week (0–6), day of month (0–31), hour (0–23), minute (0–59), second (0–59), and millisecond (0–999).

Name	Description
	When **Parameter to read** is `PTP time (time-of-day)`, output Date is visible.
Offset	Last computed offset from master PTP clock node, in nanoseconds (double). When **Parameter to read** is `Offset from Master`, output Offset is visible.
PDelay	Last computed mean path delay, in ns (double). When **Parameter to read** is `Path delay`, output PDelay is visible.

Name	Description
State	Current state of the protocol state machine. Returns one of: • 1 = INITIALIZING — Initializing data set and communication protocol • 2 = FAULTY — Occurrence of serious fault • 3 = DISABLED — Management message disables the node • 4 = LISTENING — Waiting for announce receipt timeout period to expire • 5 = PRE_MASTER — Intermediate state before moving to MASTER state after execution of Best Master Clock Algorithm (BMCA) • 6 = MASTER — Node is the master PTP clock node • 7 = PASSIVE — BCMA designates node as passive • 8 = UNCALIBRATED — Intermediate state before moving to SLAVE state after execution of BMCA • 9 = SLAVE — Node is a slave node For more information, see IEEE Std 1588-2008 Clause 9.2.5. When **Parameter to read** is `Protocol state`, output State is visible.

Block Parameters

Parameter to read

Specify parameter to read and make corresponding output port visible. Select one of:

`PTP time (nanosecond)` — Reveals port Time and parameter **Time at block start**

`PTP time (time-of-day)` — Reveals port Date

`Offset from Master` — Reveals port Offset

`Path delay` — Reveals port PDelay

`Protocol state` — Reveals port State

Time at block start

When you select this check box, the Time output contains the time at the beginning of block execution. When you clear this check box (the default), the Time output contains the time at the end of block execution.

Setting **Parameter to read** to `PTP time (nanosecond)` makes this check box visible.

Sample time (-1 for inherited)

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

See Also

Topics

"Synchronize Timestamps Across Data-Gathering Network" on page 21-5

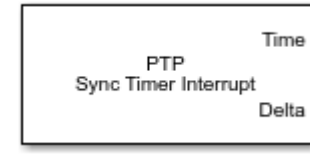
External Websites

standards.ieee.org

Introduced in R2015b

IEEE 1588 Sync Execution

Synchronize model execution to Precision Time Protocol clock
Library: IEEE 1588



Description

When the PTP time is a multiple of the fundamental step size of the model, this block causes a real-time interrupt.

Make measurements across multiple target computers at the same time step by using the IEEE 1588 Sync Execution block. The block uses a control loop to adjust the step size toward the synchronization objective. During this process, the control loop decreases or increases the step size. When the control loop decreases the step size, the CPU can become overloaded. You can decrease the maximum adjustment value by decreasing the **Proportional gain** parameter. The upper bound of the adjustment value is 10% of the model fundamental sample time, regardless of the **Proportional gain** value.

Use this block in every model that requires synchronized execution, whether it is a PTP master or slave model. To use this block, in the Simulink Real-Time options, set the real-time interrupt source to `Timer`. As a best practice, for all models, use the same fundamental sample time. Set the sample time in this block to that fundamental sample time.

If you use the IEEE 1588 Sync Execution block in your model, configuring EtherCAT distributed clocks in master shift mode in the same model produces a build error. To include IEEE 1588 synchronized execution and EtherCAT distributed clocks in the same model, use EtherCAT bus shift mode.

Ports

Output

`Time` — PTP time of real-time interrupt

scalar

PTP time value at which the interrupt occurs, in seconds.

Data Types: `double`

`Delta` — Difference between interrupt time and nearest PTP sample time

scalar

Current difference, in seconds, between the PTP time at the interrupt and the nearest PTP time that is a multiple of the fundamental sample time.

Data Types: `double`

Parameters

`Proportional gain` — Proportional gain of the kernel clock adjustment controller

`0.1` (default) | double

The current value of output port Delta is multiplied by the proportional gain to get the first part of the controller output.

`Low pass filter pole` — Pole of the low-pass filter of the kernel clock adjustment controller

`0.7` (default) | double

The low-pass filter is a discrete-time, first-order transfer function. The low-pass filter tracks the rate difference between the kernel and PTP clocks and provides the second part of the controller output.

`PTP clock synchronization threshold (seconds)` — Threshold value at which the controller begins to adjust the kernel clock

`1e-3` (default) | double

The effect of this value depends on the PTP node state:

- Slave node — The controller starts the kernel adjustment when the slave PTP clock offset from the master clock is less than or equal to this parameter.
- Master node — The controller starts the kernel clock adjustment immediately after it enters the master state, regardless of the value of this parameter.

It is a best practice to start adjusting the kernel clock only when the PTP clock is stable. Keep this value less than or equal to a millisecond.

`Sample time (-1 for inherited)` — Sample time of block
`-1` (default) | numeric

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

Model Examples

"Synchronize Timestamps Across Data-Gathering Network" on page 21-5

See Also

See Also

Transfer Fcn First Order

External Websites

standards.ieee.org

Introduced in R2016a

IEEE 1588 Sync Status

Output synchronization status of Precision Time Protocol

Library

Simulink Real-Time Library for Precision Time Protocol

Description

When the absolute value of the offset from the master PTP clock is less than or equal to the threshold, the output port value is `true`. If the node is selected as the master clock, the output port value becomes true when it enters the MASTER state.

Block Outputs

Name	Description
S	When the absolute value of the offset from the master PTP clock is less than or equal to the specified threshold, returns `true`.

Block Parameters

Offset threshold (second)

Threshold value, in seconds, from which the node is considered well synchronized to the master PTP clock node. The default value is 10 µs. The minimum allowed value is 1 µs.

Sample time (-1 for inherited)

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

See Also

Topics

“Synchronize Timestamps Across Data-Gathering Network” on page 21-5

External Websites

standards.ieee.org

Introduced in R2015b

IEEE 1588 Setup

Configure node for Precision Time Protocol execution

Library

Simulink Real-Time Library for Precision Time Protocol

Description

Sets up the Precision Time Protocol for the specified transport protocol (Ethernet or UDP). Exposes as outputs the state of the protocol, the delay measurement mechanism, and triggers for sending messages.

The PTP internal block descriptions are for informational purposes only. You cannot use these blocks by themselves in a model. The subsystem mask controls the block parameters. Do not edit the parameters directly.

Block Outputs

Name	Description
`State`	Current state of the protocol state machine. Returns one of: • 1 = INITIALIZING — Initializing data set and communication protocol • 2 = FAULTY — Occurrence of serious fault • 3 = DISABLED — Management message disables the node • 4 = LISTENING — Waiting for announce receipt timeout period to expire • 5 = PRE_MASTER — Intermediate state before moving to MASTER state

Name	Description
	after execution of Best Master Clock Algorithm (BMCA) • `6` = MASTER — Node is the master PTP clock node • `7` = PASSIVE — BCMA designates node as passive • `8` = UNCALIBRATED — Intermediate state before moving to SLAVE state after execution of BMCA • `9` = SLAVE — Node is a slave node For more information, see IEEE Std 1588-2008 Clause 9.2.5.
`DM`	Value of **Delay measurement mechanism**. Returns one of: • `1` = `Request-response` • `2` = `Peer-delay`
`ST`	Value of synchronization trigger, `true` every **Sync interval**
`AT`	Value of announce trigger, `true` every **Announce interval**
`DT`	Value of delay request trigger, `true` every **Min delay or pdelay request interval**

Block Parameters

General

Device ID

From the list, select a unique number to identify the Ethernet board. Select the same **Device ID** as the one that you selected for the protocol configuration block.

Transport protocol

The network protocol for communicating messages. Select one of `Real-Time UDP` and `Raw Ethernet`.

IP address

IP address of Ethernet card, or node, represented by the PTP Setup block.

Board time increment value

Value that changes the PTP clock.

PCI bus

Enter the PCI bus number for the Ethernet card.

PCI slot

Enter the PCI slot number for the Ethernet card.

Time Properties

Time source

Source of PTP clock signal. Select one of:

- `Precise System Time` — Synchronization of system time without hardware timestamp
- `Tick Counter` — Synchronization of tick counter read without hardware timestamp
- `Ethernet board` — Clock on PTP Ethernet board

Timescale (epoch)

Specify origin point of the PTP timescale. Select one of:

- `PTP (1970-01-01` — Precision Time Protocol standard epoch, starting January 1, 1970.
- `GPS (1980-06-01)` — Global Positioning System standard epoch, starting June 1, 1980.
- `NTP (1900-01-01)` — Network Time Protocol standard epoch, starting January 1, 1900.
- `Specify` — Selecting this value makes the **Arbitrary timescale epoch (yyyy mm dd hh)** parameter visible.

Arbitrary timescale epoch (yyyy mm dd hh)

Specify origin point for PTP timescale, in year, month, day, and hour.

When **Timescale (epoch)** is `Specify`, **Arbitrary timescale epoch (yyyy mm dd hh)** is visible.

Delay measurement mechanism

Method of measuring link delays. Select one of `Request-response` and `Peer-delay`.

In a PTP network, you must configure all nodes to use the same link delay measurement mechanism.

For more information, see IEEE Std 1588-2008 Clause 7.5.4.

Sample time (-1 for inherited)

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

Slave only

When you select this check box, you cannot allocate the PTP Ethernet card that this block represents as a master PTP clock.

In **Slave only** mode, the values of the advanced parameters (**Priority 1**, **Clock class**, **Clock accuracy**, and **Priority 2**) are set to their highest values. When the parameters have these settings, the Best Master Clock Algorithm (BMCA) cannot choose the node as best master clock unless all of the other nodes have the same configuration. If the BMCA selects a **Slave only** node as best clock, the node remains in the `LISTENING` state.

Time Intervals

Announce interval (second)

The period, in seconds, of an Announce message transmitted by a node in master state.

For more information, see IEEE Std 1588-2008 Clause 9.5.8.

Sync interval (second)

The period, in seconds, of a Sync message transmitted by a node in master state.

For more information, see IEEE Std 1588-2008 Clause 9.5.9.

Min delay or pdelay request interval (second)

Period of delay request message or of peer-delay request message transmitted by a node in the slave state. When the delay measurement mechanism is `Request-`

`response`, the block transmits delay request messages. When the mechanism is `Peer-delay`, it transmits peer-delay request messages.

For more information, see IEEE Std 1588-2008 Clauses 9.5.11 and 9.5.13.

Announce receipt timeout (in announce intervals)

Specifies the number of announce intervals a node not in the master state has to wait without receiving an announce message before the node enters the master state.

For more information, see IEEE Std 1588-2008 Clause 9.2.6.11.

Advanced

Domain number

Specify the domain number of the PTP network to which the node belongs.

A Simulink Real-Time PTP node can belong to only one PTP domain at a given time. If the node receives a PTP message with a different domain number, it ignores it. For more information, see IEEE Std 1588-2008 Clause 7.1.

When you select the **Show advanced configuration parameters** check box, **Domain number** is visible.

Current UTC offset

The current UTC offset, in seconds.

If you specify a nonzero value, that value is considered valid. The `UTCOffsetValid` flag is set to `true`. Otherwise, the flag is set to `false`. For more information, see IEEE Std 1588-2008 Clause 7.2.3.

When you select the **Show advanced configuration parameters** check box, **Current UTC offset** is visible.

Priority 1

Specify an integer value in the range `0`–`255`. When the value is `0`, the node has the highest priority. When it is `255`, the node has the lowest priority.

To assess the quality of two PTP clocks, the Best Master Clock Algorithm compares the following parameters, in order:

1. **Priority 1**
2. **Clock class**

3 **Clock accuracy**

4 **Priority 2**

If a parameter for one PTP clock has a smaller value than that parameter for the other clock, the algorithm selects the clock with the smaller value as the best clock. If all four parameters are equal for both clocks, the algorithm compares the MAC addresses of the nodes.

For more information, see IEEE Std 1588-2008 Clause 7.6.2.2.

When you select the **Show advanced configuration parameters** check box, **Priority 1** is visible.

Clock class

Specify a nonreserved integer value in the range `0`–`255`. If **Clock class** is less than `128`, the node cannot enter the `SLAVE` state. If **Clock class** is less than `128` and the node is not selected as the best clock, the node enters the `PASSIVE` state.

If you specify a reserved integer value, the block produces an error during model update. For more information, see IEEE Std 1588-2008 Clause 7.6.2.4. For a list of reserved and nonreserved **Clock class** values, see IEEE Std 1588-2008 Table 5.

When you select the **Show advanced configuration parameters** check box, **Clock class** is visible.

Clock accuracy

Specify a nonreserved integer value in the range `0`–`254`.

For more information, see IEEE Std 1588-2008 Clause 7.6.2.5. For a list of reserved and nonreserved **Clock accuracy** values, see IEEE Std 1588-2008 Table 6.

When you select the **Show advanced configuration parameters** check box, **Clock accuracy** is visible.

Priority 2

Specify an integer value in the range `0`–`255`. When the value is `0`, the node has the highest priority. When it is `255`, the node has the lowest priority.

For more information, see IEEE Std 1588-2008 Clause 7.6.2.3.

When you select the **Show advanced configuration parameters** check box, **Priority 2** is visible.

See Also

Topics

"Synchronize Timestamps Across Data-Gathering Network" on page 21-5

External Websites

standards.ieee.org

Introduced in R2015b

IEEE 1588 Adjust Time

Run Precision Time Protocol clock correction servo

Library

Simulink Real-Time Library for Precision Time Protocol

Description

When `Trigger` is `true`, IEEE 1588 Adjust Time performs time offset correction at every sample time until the offset drops below a threshold value. It then switches to rate correction. During rate correction, the block adjusts the time increment value of the Ethernet card to track the master PTP clock rate.

The PTP internal block descriptions are for informational purposes only. You cannot use these blocks by themselves in a model. The subsystem mask controls the block parameters. Do not edit the parameters directly.

Block Inputs

Name	Description
`Trigger`	While `Trigger` is `true`, the block runs the time adjustment algorithm at every time step.

Block Parameters

Sample time (-1 for inherited)

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

See Also

Topics

"Synchronize Timestamps Across Data-Gathering Network" on page 21-5

External Websites

standards.ieee.org

Introduced in R2015b

IEEE 1588 Create Message

Pack a Precision Time Protocol message for transmission

Library

Simulink Real-Time Library for Precision Time Protocol

Description

Creates and packs an IEEE 1588 message of a type specified by parameter **Message type**. Sends the message and the message length to the `Data` and `Length` outputs, respectively.

By default, IEEE 1588 Create Message creates a message at every sample time. If you select the **Enable trigger port** check box, IEEE 1588 Create Message creates a message at every sample time only when the trigger is `true`.

The PTP internal block descriptions are for informational purposes only. You cannot use these blocks by themselves in a model. The subsystem mask controls the block parameters. Do not edit the parameters directly.

Block Inputs and Outputs

Inputs

Name	Description
Trigger	While Trigger is `true`, the block creates a message at every sample time. When **Enable trigger port** is `true`, this input is visible.

Outputs

Name	Description
Data	Message data in a `uint8` array.

Name	Description
Length	Message data length (`double`).

Block Parameters

Message type

Select the PTP message type to pack. Select one of `Sync`, `Delay_Req`, `Pdelay_Req`, `Pdelay_Resp`, `Follow_up`, `Delay_Resp`, `Pdelay_Resp_Follow_up`, and `Announce`.

For more information, see IEEE Std 1588-2008 Clause 7.3.3.

Sample time (-1 for inherited)

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

Enable trigger port

Selecting this check box makes input port Trigger visible.

See Also

Topics

“Synchronize Timestamps Across Data-Gathering Network” on page 21-5

External Websites

standards.ieee.org

Introduced in R2015b

IEEE 1588 Process Message

Process a received Precision Time Protocol message

Library

Simulink Real-Time Library for Precision Time Protocol

Description

Unpack a received PTP message and execute the actions that the message data requires. For example, get timestamps and calculate offset.

The PTP internal block descriptions are for informational purposes only. You cannot use these blocks by themselves in a model. The subsystem mask controls the block parameters. Do not edit the parameters directly.

Block Inputs and Outputs

Inputs

Name	Description
`Data`	Message data in a `uint8` array.
`Length`	Message data length (`double`).

Outputs

Name	Description
`Flag`	If `true`, the two-steps flag bit in the message is set, otherwise, the bit is cleared.
`Type`	Message type (`uint8`). Return one of: • `0` = `Sync` • `1` = `Delay_Req` • `2` = `Pdelay_Req`

Name	Description
	• `3 = Pdelay_Resp` • `8 = Follow_up` • `9 = Delay_Resp` • `10 = Pdelay_Resp_Follow_up` • `11 = Announce` For more information, see IEEE Std 1588-2008 Clause 7.3.3.

Block Parameters

Sample time (-1 for inherited)

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

See Also

Topics

"Synchronize Timestamps Across Data-Gathering Network" on page 21-5

External Websites

standards.ieee.org

Introduced in R2015b

IEEE 1588 Sync Error

Output block execution step size and offset delta of real-time interrupt

Library

Simulink Real-Time Library for Precision Time Protocol

Description

The block returns the block execution step size as measured by the PTP clock. It also returns the following information:

- The PTP time when the real-time interrupt triggers.
- The difference between the PTP time at the real-time interrupt and the nearest PTP time that is a multiple of the fundamental sample time.

To use this block, in the Simulink Real-Time options, set the real-time interrupt source to `Timer`.

The PTP internal block descriptions are for informational purposes only. You cannot use these blocks by themselves in a model. The subsystem mask controls the block parameters. Do not edit the parameters directly.

Block Outputs

Name	Description
`Time`	PTP time value when the real-time interrupt occurs, in seconds
`Step`	Actual block execution step size, in seconds, as calculated from PTP clock measurements
`Delta`	Current difference, in seconds, between the PTP time at the interrupt and the nearest PTP time that is a multiple of the fundamental sample time

Block Parameters

Sample time (-1 for inherited)

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

See Also

Topics

"Synchronize Timestamps Across Data-Gathering Network" on page 21-5

External Websites

standards.ieee.org

Introduced in R2016a

Adjust Step Size

Adjust block step size during execution

Library

Simulink Real-Time Library for Precision Time Protocol

Description

The block sets the execution step size of the block. The block execution step size is equal to the fundamental sample time plus the value of `Adj`, scaled to the block sample time. For example, if the block sample time is twice the fundamental sample time, each fundamental execution step gets half of the `Adj` value.

When the block changes the execution step size, the new value remains active until the block changes the value again.

The PTP internal block descriptions are for informational purposes only. You cannot use these blocks by themselves in a model. The subsystem mask controls the block parameters. Do not edit the parameters directly.

Block Inputs

Name	Description
`Adj`	Step size increment

Block Parameters

Sample time (-1 for inherited)

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

See Also

Topics

"Synchronize Timestamps Across Data-Gathering Network" on page 21-5

External Websites

standards.ieee.org

Introduced in R2016a

Current Step Size

Output current block execution step size

Library

Simulink Real-Time Library for Precision Time Protocol

Description

The block returns the current block execution step size. When you select **Enable base rate output port**, output port BR shows the base rate or model fundamental step size.

The PTP internal block descriptions are for informational purposes only. You cannot use these blocks by themselves in a model. The subsystem mask controls the block parameters. Do not edit the parameters directly.

Block Outputs

Name	Description
CS	Current block execution step size, in seconds
BR	Base rate, or model fundamental step size, in seconds When you select **Enable base rate output port**, this port becomes visible.

Block Parameters

Sample time (-1 for inherited)

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

Enable base rate output port

To make the BR output visible, select this check box.

See Also

Topics

"Synchronize Timestamps Across Data-Gathering Network" on page 21-5

External Websites

standards.ieee.org

Introduced in R2016a

Real-Time UDP Configuration

Configure network interface for real-time UDP communication

Library

Simulink Real-Time Library for PTP UDP

Description

The Real-Time UDP Configuration block configures the network for real-time UDP operation.

IP fragmentation is not supported in Simulink Real-Time PTP UDP blocks. The packet payload is limited to 1472 bytes (1500 bytes UDP packet size − 28 bytes combined packet header size).

Block Parameters

Device ID

From the list, select a unique integer to identify the Ethernet board. Use this ID to associate the other UDP blocks with this board.

IP Address

Enter the IP address for the interface.

The addresses `0.0.0.0` and `255.255.255.255` are invalid local IP addresses.

Subnet mask

Enter the subnet mask for the interface.

Gateway

Enter the gateway address for the interface.

Ethernet Driver

Identifies the driver for each chip family supported. Possible values are `Intel 8255X` and `Intel Gigabit`.

PCI bus

Enter the PCI bus number for the Ethernet card.

PCI slot

Enter the PCI slot number for the Ethernet card.

See Also

See Also

`Receive` | `Send`

External Websites

www.iso.org

Introduced in R2014b

Receive

Receive data over UDP network on a dedicated network interface

Library

Simulink Real-Time Library for PTP UDP

Description

The Receive block receives UDP data on the specified local (destination) port. To receive all data sent to this port, set **Source IP address** to `0.0.0.0`, otherwise set **Source IP address** to a valid IP address.

IP fragmentation is not supported in Simulink Real-Time PTP UDP blocks. The packet payload is limited to 1472 bytes (1500 bytes UDP packet size − 28 bytes combined packet header size).

Block Outputs

Name	Description
Data	Vector of `uint8` containing data received
N	Number of new bytes received, and otherwise `0`

The default block behavior is to keep the previous output when there is no new data.

Block Parameters

Device ID

From the list, select a unique number to identify the Ethernet board. Select the same **Device ID** as the one you selected for the Real-Time UDP Configuration block.

Source IP address

Enter a valid IP address as a dotted decimal character vector, for example, `10.10.10.3`. You can also use a MATLAB expression that returns a valid IP address as a character vector. With **Local (destination) port**, this parameter defines the source address.

The default address, `0.0.0.0`, causes the block to accept UDP packets from any accessible computer. If set to a specific IP address, packets arriving from only that IP address are received.

The address `255.255.255.255` is an invalid IP address.

Local (destination) port

Specify the port of the target computer or device from which to receive the UDP packets. With **Source IP address**, this parameter defines the source address.

Output port width

Enter the output port width.

Sample time

Enter the base sample time or a multiple of the base sample time (`-1` means sample time is inherited).

See Also

See Also

`Real-Time UDP Configuration` | `Send`

External Websites

www.iso.org

Introduced in R2011a

Send

Send data over UDP network on a dedicated network interface

Library

Simulink Real-Time Library for PTP UDP

Description

The Send block sends UDP packets from **Local (source) port** to **Destination port**. To broadcast to all devices, set **Destination IP address** to `255.255.255.255`, otherwise set **Destination IP address** to a valid IP address.

IP fragmentation is not supported in Simulink Real-Time PTP UDP blocks. The packet payload is limited to 1472 bytes (1500 bytes UDP packet size − 28 bytes combined packet header size).

Block Inputs

Name	Description
Data	Vector of `uint8` containing data to send
N	Number of bytes to send

Block Parameters

Device ID

From the list, select a unique number to identify the Ethernet board. Select the same **Device ID** as the one you selected for the Real-Time UDP Configuration block.

Destination IP address

Specify the IP address of the target computer or other device to which you want to send the UDP packets. To broadcast the packets to all listening computers or

devices, enter `255.255.255.255`. With **Destination port**, this parameter defines the destination address.

Destination port

Specify the target computer port to which you want to send the UDP packets. With **Destination IP address**, this parameter defines the destination address.

Local (source) port

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 (`-1` means sample time is inherited).

See Also

See Also

`Real-Time UDP Configuration` | `Receive`

External Websites

www.iso.org

Introduced in R2011a

UDP Consume

Consume a UDP packet

Library

Simulink Real-Time Library for PTP UDP

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

IP Group

Enter a number in the range 0 to 7. This value identifies the IP Init block inside the Real-Time UDP Configuration subsystem that is associated with this block.

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 (`-1` means sample time is inherited).

See Also

External Websites

www.iso.org

Introduced in R2011a

UDP Produce

Produce UDP packet by adding a UDP header to the input data

Library

Simulink Real-Time Library for PTP UDP

Description

The UDP Produce block receives a network buffer and adds a header to that buffer. It then outputs that updated buffer.

Block Parameters

IP Group

Enter a number in the range `0` to `7`. This value identifies the IP Init block inside the Real-Time UDP Configuration subsystem that is associated with this block.

IP address to send to (255.255.255.255 for broadcast)

Specify IP address of the target computer to which to send the UDP packets. To broadcast the packets to all listening computers 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 target computer port to which 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 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 (`-1` means sample time is inherited).

See Also

External Websites

www.iso.org

Introduced in R2011a

UDP Rx

Receive UDP packet

Library

Simulink Real-Time Library for PTP UDP

Description

The UDP Rx block outputs a network buffer with a UDP header.

Block Parameters

IP Group

Enter a number in the range `0` to `7`. This value identifies the IP Init block inside the Real-Time UDP Configuration subsystem that is associated with this block.

IP address to receive from (0.0.0.0 for accepting all)

Enter a valid IP address as a dotted decimal character vector. For example, `10.10.10.3`. You can also use a MATLAB expression that returns a valid IP address as a character vector. With **IP port to receive from**, this parameter defines the source address.

The default address, `0.0.0.0`, enables the acceptance of all UDP packets from any accessible computer. If set to a specific IP address, only packets arriving from that IP address are received.

IP port to receive from

Specify the port of the target computer or device from which to receive the video frames. With **IP address to receive from (0.0.0.0 for accepting all)**, this parameter defines the source address.

Sample time

Enter the base sample time or a multiple of the base sample time (`-1` means sample time is inherited).

See Also

External Websites

www.iso.org

Introduced in R2011a

UDP Tx

Transmit UDP packet

Library

Simulink Real-Time Library for PTP UDP

Description

The UDP Tx block receives a network buffer with a UDP header and sends it.

Block Parameters

IP Group

Enter a number in the range 0 to 7. This value identifies the IP Init block inside the Real-Time UDP Configuration subsystem that is associated with this block.

Sample time

Enter the base sample time or a multiple of the base sample time (-1 means sample time is inherited).

See Also

External Websites

www.iso.org

Introduced in R2011a

SAE J1939

SAE J1939 Blocks

The Simulink Real-Time J1939 blocks enable you to send and receive messages over a FIFO-mode CAN network using the SAE J1939 message protocol. See “CAN”.

Before you start, provide a J1939 database in `.dbc` format.

The “J1939 I/O - Using Transport Protocol” example illustrates the transmission and reception of J1939 data over a FIFO mode CAN network. It uses a loopback connection of two CAN ports in a single target computer. The example requires a J1939 database file, `C:\work\J1939_demo.dbc`, which the Simulink Real-Time software does not supply. Provide your own database file, which must contain the requested information outlined in the example.

This example assumes familiarity with the SAE J1939 specifications.

More About

- “J1939 I/O - Using Transport Protocol”
- “CAN”

SAE J1939 Blocks

J1939 Controller Application

J1939 Controller Application

Library

Simulink Real-Time Library for 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. Once conflicts are resolved, the blocks on the CAN network each have unique addresses and IDs.

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 10 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 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.

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

External Websites

www.sae.org/

Introduced in R2009b

J1939 Database (CANdb) Setup

J1939 Database Setup

Library

Simulink Real-Time Library for J1939

Description

The J1939 Database Setup block is where you specify the user-supplied database for the J1939 block set. Use one block per model.

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 folder; 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: You must supply the database file. The Simulink Real-Time 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**

  Specifies that the database format is simplified ID (CANDB++ 2.7 SP7 and before).

See Also

External Websites

www.sae.org/

Introduced in R2009a

J1939 Message Trigger

J1939 Message Triggering

Library

Simulink Real-Time Library for 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.

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.

See Also

External Websites

www.sae.org/

Introduced in R2009a

J1939 Protocol Stack

J1939 Protocol Stack instance

Library

Simulink Real-Time Library for J1939

Description

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.

Block Parameters

General tab:

Protocol Stack ID

Enter the protocol stack ID.

Max CAN Message Receive (Per Sample Time)

Specify maximum number of CAN messages that the block can receive (process) in a single sample time.

Max CAN Message Transmit (Per Sample Time)

Specify maximum number of CAN messages the block can send in a single sample time.

Sample time

Specify the sample rate for the protocol stack.

Transport Protocol tab:

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.

See Also

External Websites

www.sae.org/

Introduced in R2009a

J1939 Receive Message

J1939 Receive Message

Library

Simulink Real-Time Library for J1939

Description

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.

Block Outputs

Signal Output(s)

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.

New Message Received (Optional)

Outputs `1` when a new message is received from the CAN bus; otherwise, outputs `0`.

Block Parameters

PGN

From the list, select the parameter group number. The contents of this list vary depending on the messages that the J1939 database file specifies.

CA ID

Specify the ID of the controller application that this block maps to. The ID must match the **CA ID** value of the corresponding `J1939 Controller Application` block.

Source Address Filter (255:all)

Specify the source node address from which the block is to expect messages. Type `255` to receive messages from any node.

Destination Address Filter

From the list, select the node destination of the expected 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.

See Also

External Websites

www.sae.org/

Introduced in R2009a

J1939 Transmit Message

J1939 Transmit

Library

Simulink Real-Time Library for J1939

Description

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.

Block Inputs

- `Trigger`

  Enables the transmission of the message for that sample (a value of `1` indicates send, a value of `0` indicates do not send).

- `Signal Input(s)`

  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.

Block Parameters

PGN

From the list, select the parameter group number. The contents of this list vary depending on the messages that the J1939 database file specifies.

CA ID

Identifies the controller application that this block maps to. The ID must match the **CA ID** value of the corresponding `J1939 Controller Application` block.

Priority (0-7; highest to lowest

From a range of 0 to 7, specify the priority for the message transmission. `0` is the highest, `7` is the lowest.

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.

See Also

External Websites

www.sae.org/

Introduced in R2009a

Shared Memory Support

This topic describes implementations of reflective (shared) memory by various manufacturers.

Create GE Fanuc Shared Partitions

The Simulink Real-Time software uses a model for reflective (shared) memory that includes Simulink blocks for shared memory driver functions. To define node initialization and shared memory partitions, the driver functions use MATLAB structures. This topic describes the Simulink Real-Time 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 are collectively referenced as PCI-5565 in the documentation. Simulink Real-Time supports these boards with the same set of blocks.

To use the 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 Simulink Real-Time 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 “GE Fanuc Shared Partition Structure” on page 25-6 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. Partially define a structure in a script file, run that script in the MATLAB workspace, and fill in the resulting structure with 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 data type, size, and alignment of the preceding segment define the base addresses of subsequent segments.
  - To populate the partition structure, call the `completepartitionstruct()` command.

    ```
    Partition = completepartitionstruct(Partition,'5565');
    ```

Initialize GE Fanuc Shared 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 GE Fanuc Shared Partitions” on page 25-2) and the PCI-5565 board configuration, including interrupt settings. 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 “GE Fanuc Shared Node Initialization Structure” on page 25-8 for the complete list of fields in a node initialization structure.

The following description refers to the `completenodestruct` command. Type

```
help completenodestruct
```

for more information.

- Create a node initialization structure in one of the following ways. Using the `completenodestruct` command at the MATLAB Command Window, create a default node initialization structure. For example, type

  ```
  node=completenodestruct([],'5565')

  node =

       Interface: [1x1 struct]
      Partitions: [1x1 struct]
  ```

- Fill in the structure fields. 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 `completenodestruct` function. For example:

  ```
  node.Interface.NodeID = '128';
  ```

```
node.Partitions = Partition;
node.completenodestruct(node,'5565');
```

GE Fanuc Shared Partition Structure

You do not need to use all the fields of a partition initialization structure. However, knowing the possible structure fields is 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
`Address`	Specifies the base address (in hexadecimal) of the memory partition within the shared memory space of the node. The default value is `'0x0'`, the first location in shared memory. Align partition addresses on 32-bit word boundaries (for example, `0x0`, `0x4`, `0x8`).
`Type`	Specifies the data type of the memory segment. Specify one of the following types: • `single` (IEEE Single Precision) • `double` (IEEE Double Precision) • `uint8` • `int8` • `uint16` • `int16` • `uint32` • `int32` • `Boolean` (a single byte represents a boolean value) The default value is `'uint32'`. The minimum partition size is 32 bits.

Partition Fields	Description
`Size`	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'`. • scalar — Treats the `Size` entry as the specification of the length of a non-oriented array or vector • `[m,n]` — Treats the `Size` entry as an array dimension. The total number of elements in this segment is `m*n`.
`Alignment`	If another partition precedes this partition, this field 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.
`Internal`	Reserved for internal use.

GE Fanuc Shared Node Initialization Structure

A node initialization structure has the following fields:

```
     Interface: [1x1 struct]
    Partitions: [1x1 struct]
```

where

Node Structure Fields	Description
`Interface`	Specifies how the board is configured. The Interface structure has the following fields, three of which are structures: • `Mode` — Configures board registers (see "Board Mode" on page 25-8) • `Interrupts` — 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 25-9) • `NodeID` — Specifies the node ID for the board (see "Board Node ID" on page 25-11) • `Internal` — Reserved for internal use
`Partitions`	Stores the shared memory segments (see "Create GE Fanuc Shared Partitions" on page 25-2)

Board Mode

The PCI-5565 board has several 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'
```

The mode values interact with 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:

Board Modes	Description
`StatusLEDOff`	Turns the PCI-5565 board status LED on and off. Setting this value to 'off' turns off the LED when the Simulink Real-Time model runs, setting this value to 'on' turns on the LED when the Simulink Real-Time model runs. When the Simulink Real-Time software terminates, the LED status reverses in both cases. The default value is `'off'`.
`MemoryOffset`	Applies a global offset to the network data transfers coming from the PCI-5565 board. The following table lists offset values and the resulting offset. The default value is `'0'`.
`MemorySize`	Specifies the minimum memory size required, in the format `'`*`size`*`MByte'`. 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 returns an error.

This table lists the values for `MemoryOffset`:

Value	Offset Produced
`'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 Simulink Real-Time Simulink models, one as master, and one as a slave of the broadcast node in the master Simulink Real-Time model. In such a configuration, the broadcast node interrupt triggers the model 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 (the master model). This block issues network interrupts at the model sample rate. To enable other nodes of the network (the slave models) to accept broadcast interrupts, configure the slave models to expect the broadcast interrupt.

The following procedure describes how to configure an entire Simulink Real-Time 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 Simulink Real-Time model expects.

1 From the MATLAB Command Window, type

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

This command lists board information for the installed PCI devices that the Simulink Real-Time software knows about.

2 Find the IRQ specified for the PCI-5565 board.

   To specify in the **Real-Time interrupt source** field of the **Simulink Real-Time Options** pane, use this interrupt source number.

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 **Simulink Real-Time 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, 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:

NodeID Value	Description
`'any'`	Allows the PCI-5565 driver to work with a PCI-5565 node regardless of the PCI-5565 board node ID jumper setting

NodeID Value	Description
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' meets requirements in most instances. If you have multiple PCI-5565 boards in your system, to identify the driver for a particular node, specify a particular NodeID value.

Create Curtiss-Wright Shared Partitions

The Simulink Real-Time software uses a model for reflective (shared) memory that includes Simulink blocks for shared memory driver functions. To define node initialization and shared memory partitions, the driver functions use MATLAB structures. This topic describes Simulink Real-Time support of the Systran® SCRAMNet + SC150 board.

To use the Simulink Real-Time 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 Simulink Real-Time 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 “Curtiss-Wright Shared Partition Structure” on page 25-16 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. Partially define a structure in a script file, run that script in the MATLAB workspace, and fill in the resulting structure with 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 data type, size, and alignment of the preceding segment define the base addresses of subsequent segments.
  - To populate the partition structure, call the `completepartitionstruct()` command.

    ```
    Partition = completepartitionstruct(Partition,'scramnet');
    ```

Initialize Curtiss-Wright Shared 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 Curtiss-Wright Shared Partitions" on page 25-13) and the SCRAMNet+ SC150 board configuration, including interrupt settings. 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 "Curtiss-Wright Shared Node Initialization Structure" on page 25-21 for the complete list of fields in a node initialization.

The following description refers to the `completenodestruct` command. Type

```
help completenodestruct
```

for more information.

- Create a node initialization structure in one of the following ways. Using the `completenodestruct` command at the MATLAB Command Window, create a default node initialization structure. For example, type

  ```
  node=completenodestruct([],'scramnet')

  node =

       Interface: [1x1 struct]
      Partitions: [1x1 struct]
  ```

- Fill in the structure fields. 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 `completenodestruct` function. For example:

  ```
  node.Interface.NodeID = '128';
  node.Partitions = Partition;
  node = completenodestruct(node,'scramnet');
  ```

Curtiss-Wright Shared 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 is 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

Partition Fields	Description
Address	Specifies the base address (in hexadecimal) of the memory partition within the node shared memory space. The default value is `'0x0'`, the first location in shared memory. The base address is byte aligned.
Type	Specifies the data type of the memory segment. Specify one of the following types: • `double` • `float` • `uint8` • `int8` • `uint16` • `int16` • `uint32` • `int32` • `boolean` (a single byte represents a boolean value) The minimum partition size is 32 bits.

Partition Fields	Description
	The default value is `'uint32'`.
Size	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'`. • scalar — Treats the `Size` entry as the specification of the length of a non-oriented array or vector • `[m,n]` — Treats the `Size` entry as an array dimension. The total number of elements in this segment is `m*n`.
Alignment	Specifies the byte alignment of the next partition (if one is defined). Enter alignment value in bytes: `1`, `2`, `3`, `4`. The alignment value defines the end of the current segment, and therefore the beginning alignment of the next segment. The default value is `'4'`, forcing a double word boundary alignment. See “Alignment Examples” on page 25-19.
RIE	Specifies whether this partition can receive interrupts (Receive Interrupt Register (RIE)). Specify one of: • `'off'` (default) — Prevents the partition from receiving interrupts. • `'first'` — Allows only the first double word of the memory segment to be marked with the corresponding Auxiliary Control RAM bit. • `'all'` — Allows all memory locations of the memory segment to be marked with the corresponding Auxiliary Control RAM bit. • `'last'` — Allows only the last double word of the memory segment to be marked with the corresponding Auxiliary Control RAM bit.

Partition Fields	Description
TIE	Specifies whether this partition can transmit interrupts (Transmit Enable (TIE)). Specify one of: • `'off'` (default) — Prevents the partition from transmitting interrupts. • `'first'` — Allows only the first double word of the memory segment to be marked with the corresponding Auxiliary Control RAM bit. • `'all'` — Allows all memory locations of the memory segment to be marked with the corresponding Auxiliary Control RAM bit. • `'last'` — Allows only the last double word of the memory segment to be marked with the corresponding Auxiliary Control RAM bit.
ExtTrigger1	If this partition receives a write access, specifies whether this partition can generate a trigger signal to an external connector. Specify one of: • `'off'` (default) — Prevents the partition from generating signals. • `'first'` — Allows only the first double word of the memory segment to be marked with the corresponding Auxiliary Control RAM bit. • `'all'` — Allows all memory locations of the memory segment to be marked with the corresponding Auxiliary Control RAM bit. • `'last'` — Allows only the last double word of the memory segment to be marked with the corresponding Auxiliary Control RAM bit.

Partition Fields	Description
ExtTrigger2	If this partition receives a write access, specifies whether this partition can generate a trigger signal to an external connector. Specify one of: `'off'` (default) — Prevents the partition from generating signals. `'first'` — Allows only the first double word of the memory segment to be marked with the corresponding Auxiliary Control RAM bit. `'all'` — Allows all memory locations of the memory segment to be marked with the corresponding Auxiliary Control RAM bit. `'last'` — Allows only the last double word of the memory segment to be marked with the corresponding Auxiliary Control RAM bit.
HIPRO	Specifies whether the elements in this partition can be transmitted as one network message. Specify one of: `'off'` (default) — Prevents the partition from transmitting the elements as one message `'on'` — Allows the partition to transmit the elements as one message
Internal	Reserved for internal use.

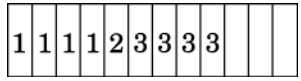
Alignment Examples

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');
```

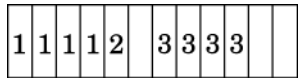
This example shows the shared memory map with alignment value changed from 4 to 1 in the second partition.

```
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');
```



Curtiss-Wright Shared Node Initialization Structure

A node initialization structure has the following fields:

```
     Interface: [1x1 struct]
    Partitions: [1x1 struct]
```

where

Node Structure Fields	Description
Interface	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. • `Mode` — Configures board modes (see "Board Mode" on page 25-21) • `Timeout` — Enables the board to set the timeout value (see "Board Timeout" on page 25-23) • `DataFilter` — Controls the data filtering operation (see "Board Data Filter" on page 25-23) • `VirtualPaging` — Controls the board virtual paging operation (see "Virtual Paging" on page 25-24) • `Interrupts` — Enables the board to generate and receive interrupts from the network (see "Board Interrupts" on page 25-24) • `Internal` — Reserved for internal use
Partitions	Stores the shared memory segments (see "Create Curtiss-Wright Shared Partitions" on page 25-13)

Board Mode

The SCRAMNet+ SC150 board has 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
NetworkCommunications Mode	`'none'`, `'receiveonly'`, `'transmitonly'`, `'transmit receive'`	`'transmit receive'`	CSR3[8..15]
InsertNode	`'off'`, `'on'`	`'on'`	CSR0[0..1]
DisableFiberOptic Loopback	`'off'`, `'on'`	`'on'`	CSR2[6]
EnableWire Loopback	`'off'`, `'on'`	`'off'`	CSR2[7]
DisableHost ToMemory Write	`'off'`, `'on'`	`'off'`	CSR2[8]
WriteOwnSlotEnable	`'off'`, `'on'`	`'off'`	CSR2[9]
Message LengthLimit	`'256'`, `'1024'`	`'256'`	CSR2[11]
Variable Length MessagesOn Network	`'off'`, `'on'`	`'off'`	CSR212]
HIPROEnable	`'off'`, `'on'`	`'off'`	CSR2[13]
Multiple Messages	`'off'`, `'on'`	`'on'`	CSR2[14]
NoNetwork Error Correction	`'off'`, `'on'`	`'on'`	CSR2[15]
Mechanical Switch Override	`'off'`, `'on'`	`'on'`	CSR8[11]
Disable Holdoff	`'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
NumOfNodes InRing	`'0' to '255'`	`'2'`	CSR5
TotableCable LengthInM	`'0' to '`*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'` to `'2047'`	`'0'`	CSR12[5..15]

Board Interrupts

The SCRAMNet+ SC150 board allows you to specify the interrupt sources transmitted and received between the nodes of the network. You can set these bits through the `Interface.Interrupts` field. The Interface Interrupts fields set the corresponding bits in the CSR.

```
>> node.Interface.Interrupts
ans =
                HostInterrupt: 'off'
    InterruptOnMemoryMaskMatch: 'off'
      OverrideReceiveInterrupt: 'off'
               InterruptOnError: 'off'
               NetworkInterrupt: 'off'
     OverrideTransmitInterrupt: 'off'
            InterruptOnOwnSlot: 'off'
      ReceiveInterruptOverride: 'off'
```

These fields have the following values:

Field	Values	Default	CSR
HostInterrupt	`'off'`, `'on'`	`'off'`	CSR0[3]
InterruptOn MemoryMask Match	`'off'`, `'on'`	`'off'`	CSR0[5]

Field	Values	Default	CSR
Override Receive Interrupt	`'off'`, `'on'`	`'off'`	CSR0[6]
InterruptOn Error	`'off'`, `'on'`	`'off'`	CSR0[7]
Network Interrupt	`'off'`, `'on'`	`'off'`	CSR0[8]
Override Transmit Interrupt	`'off'`, `'on'`	`'off'`	CSR0[9]
InterruptOn OwnSlot	`'off'`, `'on'`	`'off'`	CSR2[10]
Receive Interrupt Override	`'off'`, `'on'`	`'off'`	CSR8[10]

Video, XCP

Video Image Processing

Process Video Images with Simulink Real-Time

The Simulink Real-Time 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.

Using blocks from the Simulink Real-Time Video library, on your target computer, you can do the following:

- Acquire real-time video frames from cameras connected to the target computer.
- Display the real-time image on the target computer monitor.
- Process or reduce the real-time image, for instance to choose a region of interest.
- Compress video frames acquired on the target computer.
- Stream video frames acquired on the target computer to the development computer.

Using blocks from the Computer Vision System Toolbox™ or Image Processing Toolbox™, on your development computer, you can do the following:

- Receive images from the target computer.
- Decompress video frames on the development computer.
- Process or reduce images.
- Display images on the development computer.

USB Video Display on Development Computer

The following workflow acquires and displays video frames using a USB Video Class (UVC) compliant webcam. To view the images on the development computer, compress the video frames on the target computer and transmit them to the development computer. You then decompress and display the video frames.

1. Enable USB general support through the target computer BIOS. See "BIOS Settings".
2. Acquire a USB Video Class (UVC) compliant webcam, and then connect it to the target computer USB port.
3. To query the available camera configurations, add the `USB Video Device List` block.
4. Add the image input block `From USB Video Device` to the portion of the model that runs on the target computer. Configure the block as required.
5. If your USB camera does not support on-chip JPEG compression, add the `JPEG Compression` block to the target computer portion. Connect this block to the output of the image input block.
6. Add the `Image Transmit` block to the target computer portion. Connect this block to the output of the compression block. Configure the block to transmit frames to the development computer.
7. Add the `Image Receive` block to the portion of the model that runs on the development computer. Configure the block to receive frames from the target computer.
8. Add the `JPEG Decompression` block to the development computer portion. Connect this block to the output of the `Image Receive` block.
9. Add the To Video Display block from the Computer Vision System Toolbox to the development computer portion.
10. Build and download the target computer portion of the model to the target computer.
11. Run the development computer portion of the model on the development computer.

USB Video Display on Target Computer

The following workflow acquires and displays video frames using a USB Video Class (UVC) compliant webcam. To view the images on the target computer, use a Video Display block.

1. Enable USB general support through the target computer BIOS. See "BIOS Settings".
2. Acquire a USB Video Class (UVC) compliant webcam, and then connect it to the target computer USB port.
3. To query the available camera configurations, add the `USB Video Device List` block.
4. Add the image input block `From USB Video Device` to the portion of the model that runs on the target computer. Configure the block as required.

   Record the setting of the **Image signal** parameter.
5. Add the `Video Display` block. Set its **Image signal** parameter to match the corresponding setting in the From USB Video Device block.
6. Build and download the target computer portion of the model to the target computer.
7. Execute the model on the target computer.

Camera Link Camera Display on Development Computer

The following workflow acquires and displays video frames using a camera that is compliant with the Camera Link standard. To view the images on the development computer, compress the video frames on the target computer and transmit them to the development computer. You then decompress and display the video frames.

1. Install the BitFlow Neon-CLB board in a PCI Express® `x4` or wider slot in the target computer.
2. Install the BitFlow™ Neon-CLB Support Package and the BitFlow SDK. See "Install BitFlow Neon-CLB Support" on page 26-17.
3. Acquire a monochrome digital camera that is compliant with Camera Link. The camera must have a configuration file in the BitFlow SDK that specifies a free-running or triggered mode.

   To configure the BitFlow Neon-CLB board, the `NEON BitFlow Image Input` block requires the BitFlow SDK and a configuration file for your specific camera. If the SDK does not contain the required file, contact your BitFlow representative. You cannot proceed without this file.
4. Attach the camera to the BitFlow Neon-CLB board in the target computer.
5. Add the `NEON BitFlow Image Input` block, available in the Simulink Real-Time Video library, to the portion of the model that runs on the target computer. Configure the block as required.
6. Add the `JPEG Compression` block to the target computer portion. Connect this block to the output of the image input block.
7. Add the `Image Transmit` block to the target computer portion. Connect this block to the output of the compression block. Configure the block to transmit frames to the development computer.
8. Add the `Image Receive` block to the portion of the model that runs on the development computer. Configure the block to receive frames from the target computer.
9. Add the `JPEG Decompression` block to the development computer portion. Connect this block to the output of the `Image Receive` block.
10. Add the To Video Display block from the Computer Vision System Toolbox to the development computer portion.
11. Build and download the target computer portion of the model to the target computer.

12 Run the development computer portion of the model on the development computer.

Camera Link Camera Display on Target Computer

The following workflow acquires and displays video frames using a camera that is compliant with the Camera Link standard. To view the images on the target computer, use a Video Display block.

1. Install the BitFlow Neon-CLB board in a PCI Express x4 or wider slot in the target computer.
2. Install the BitFlow Neon-CLB Support Package and the BitFlow SDK. See "Install BitFlow Neon-CLB Support" on page 26-17.
3. Acquire a monochrome digital camera that is compliant with Camera Link. The camera must have a configuration file in the BitFlow SDK that specifies a free-running or triggered mode.

   To configure the BitFlow Neon-CLB board, the `NEON BitFlow Image Input` block requires the BitFlow SDK and a configuration file for your specific camera. If the SDK does not contain the required file, contact your BitFlow representative. You cannot proceed without this file.
4. Attach the camera to the BitFlow Neon-CLB board in the target computer.
5. Add the `NEON BitFlow Image Input` block, available in the Simulink Real-Time Video library, to the target computer portion of the model. Configure the block as required.
6. Add the `Video Display` block. Set its **Image signal** parameter to `Single multidimensional signal`.
7. Build and download the target computer portion of the model to the target computer.
8. Execute the model on the target computer.

Acquire Images from Camera Link Cameras

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 `www.bitflow.com`).

Be familiar with the following information:

- Video capture procedures
- Camera Link interface standard
- Manufacturer instructions for installing the BitFlow Neon-CLB board on the target computer
- Instructions for installing the BitFlow SDK. See “Install BitFlow Neon-CLB Support” on page 26-17.
- Serial command codes for your particular camera model. See “Serial Camera Configuration” on page 26-15.

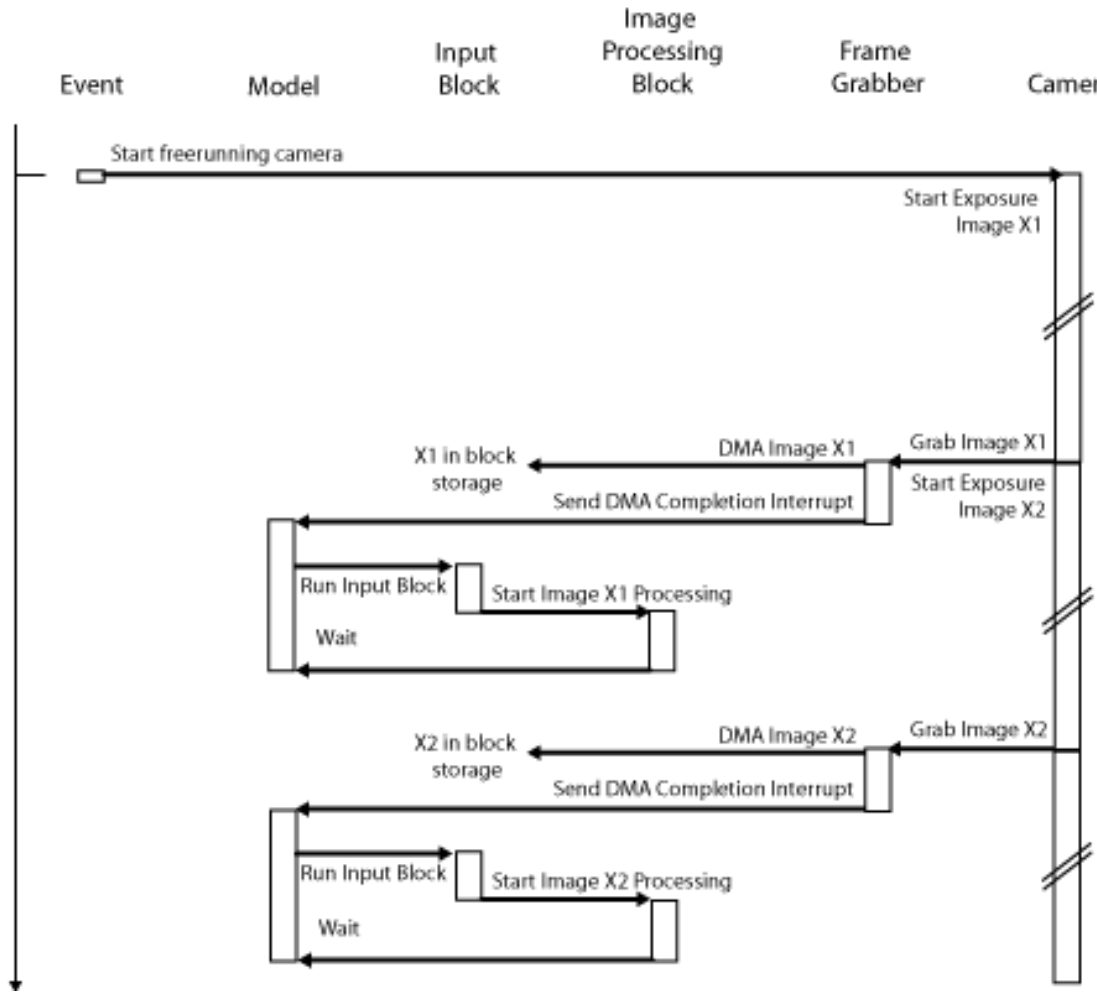
The Simulink Real-Time product contains the following example that illustrates how to acquire images from a camera connected to the BitFlow Neon-CLB board.

`“Image Capture with Camera Link® and Bitflow™ Neon-CLB Frame Grabber”`

Camera Link 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 computer memory using DMA and signals a frame completion interrupt. Upon receiving the interrupt, the model runs and performs image processing.



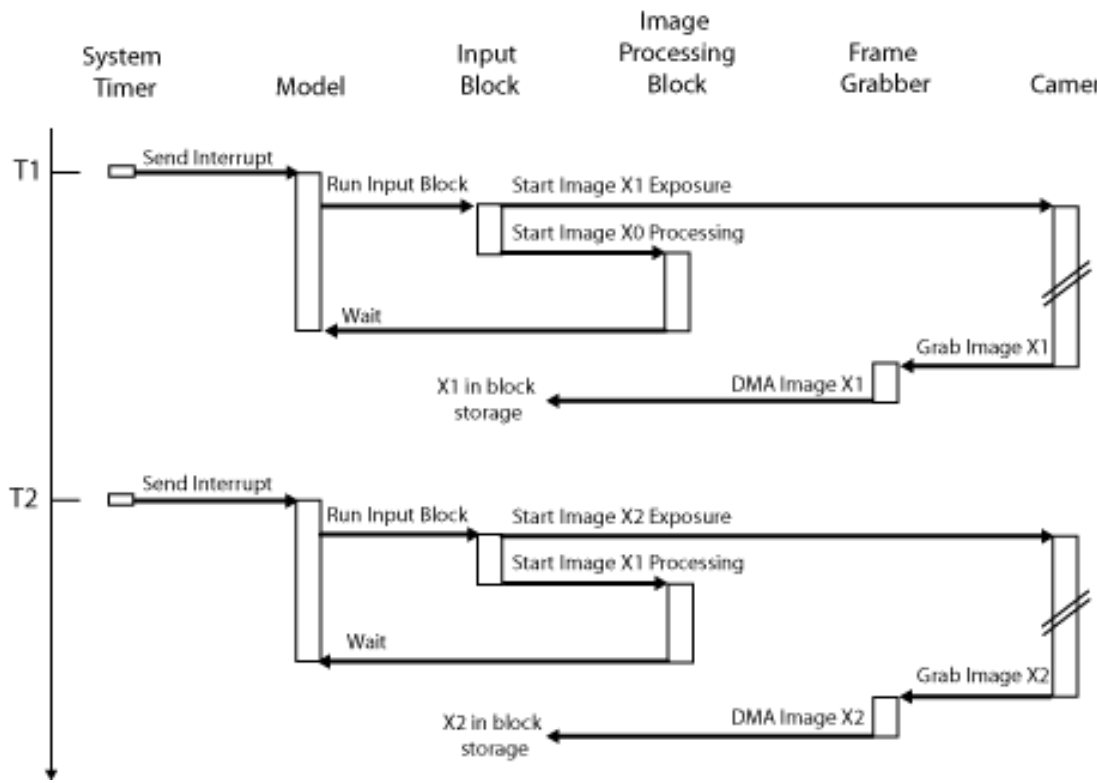
Before the next DMA completion interrupt, the model must complete image processing. Otherwise, the CPU becomes overloaded.

If the model requires heavy image processing, 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. 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. The subsystem runs at T_s * 3, sampling every third image. The model is configured for a large format camera where the Jpeg Compression and Image Transmit blocks take more than T_s to complete.

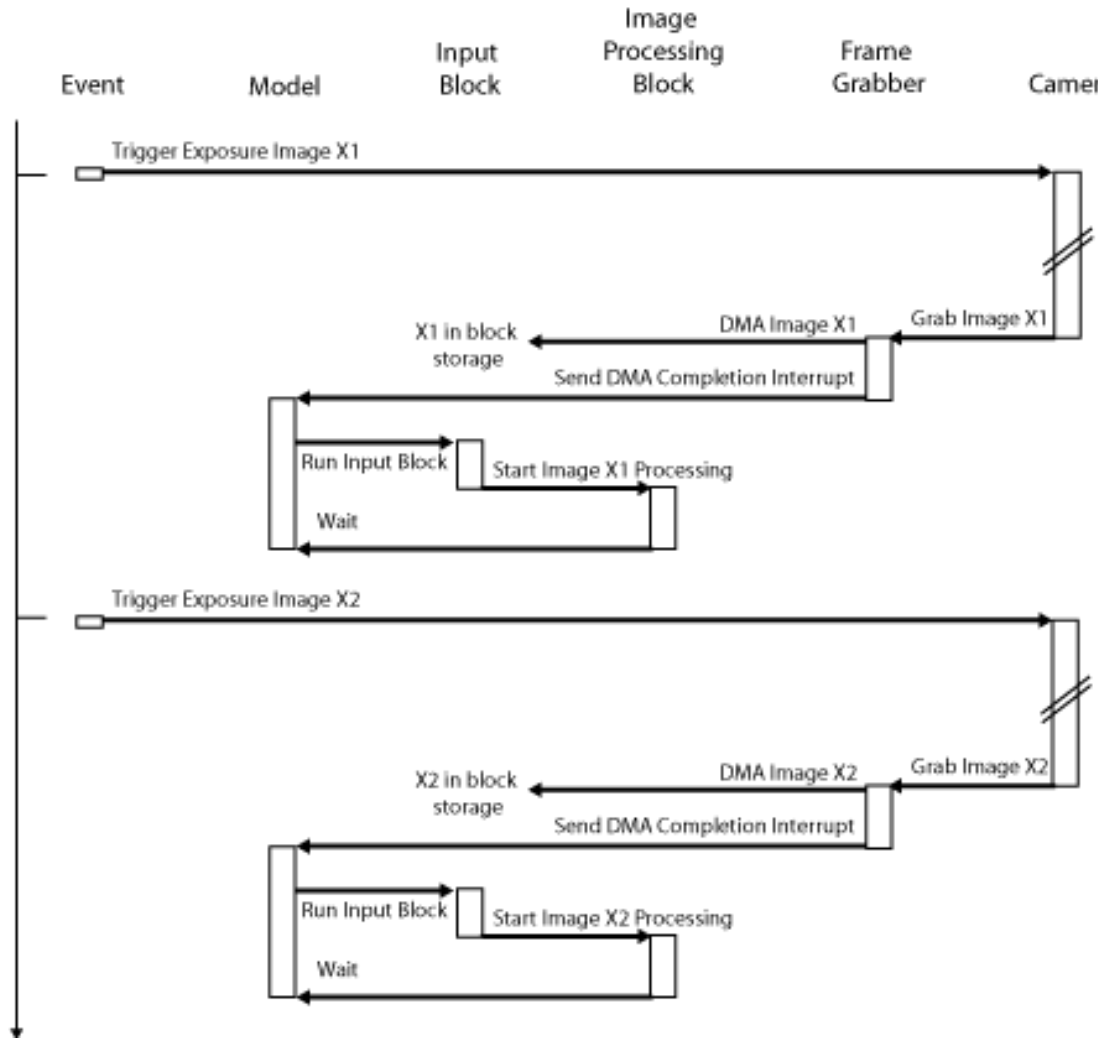
- `Timer Triggered`. The target computer system timer starts the model. When the model runs, it simultaneously starts a camera exposure and begins to process the image captured during the previous time step. When the camera finishes the exposure, the frame grabber copies the image to target computer memory using DMA for processing in the next time step.



- `Differential Trigger`. A rising edge on the differential trigger input triggers the camera. When the camera finishes the exposure, the frame grabber copies the image

to target computer memory using DMA, and signals a frame completion interrupt. Upon receiving the interrupt, the model runs and performs image processing.

Camera Link cameras are designed to respond to one of four camera control lines (`CC1` to `CC4`) coming from the frame grabber. The required control line is in the triggered-mode configuration file for the camera. Connect the external trigger to the frame grabber trigger input.



- `TTL Trigger`. A rising edge on the TTL trigger input triggers the camera. The model runs from the frame completion interrupt.
- `Opto Trigger`. A rising edge on the Opto trigger input triggers the camera. The model runs from the frame completion interrupt.

The camera configuration file puts the frame grabber into triggered mode, but does not put the camera into that mode. To set the camera to triggered mode, send a command to the camera. See “Serial Camera Configuration” on page 26-15.

The camera configuration file determines whether you can use the block mask to switch the trigger mode to another mode, such as `Free Running`. You could need to contact BitFlow for a camera configuration file for the mode that 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.

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 different from any other. For example, suppose that you want to put a camera from the given series into triggered mode and to set the shutter time to 1 ms. Use these settings and serial setup commands.

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`
Sony XCL	38,400 baud 8 bits No parity	`SHUTTER 7\r` `TRG_MODE 1\r`

Cohu 7800	9600 baud
	8 bits
	No parity
		`\002\0377cE2,10`*`chk`*`\003`
		`\002\0377cM2`*`chk`*`\003`
		`\002\0377T0,1`*`chk`*`\003`
		`chk` is a checksum for the given portion of the message.

For your camera, find the required command codes in the manufacturer documentation for that specific camera. If those codes are not in the manual, contact the camera manufacturer.

Some cameras return a success or failure value in response to the serial 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.

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

Some camera manufacturers offer Windows utilities to send configuration commands to their cameras. Using such a utility, you can configure the camera on Windows and save the settings on the camera. You can then connect the camera to the target computer and use it.

Other camera manufacturers require a special serial cable that connects the serial communication line of the camera to a separate serial connection. In this case, you cannot initialize the camera using settings in the **Camera configuration serial command** section. Instead, you connect the DB9 to a serial port on the development computer and use manufacturer software to configure the camera before use.

Install BitFlow Neon-CLB Support

To add support for BitFlow Neon-CLB I/O modules, do the following:

1. From the BitFlow website (www.bitflow.com), install the BitFlow SDK.
2. After installing the BitFlow SDK, test the installation by building and downloading a video example.

Video Blocks

BitFlow Neon-CLB

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 that is compliant with Camera Link with power over Camera Link (PoCL)
- SafePower, which protects against Camera Link power line faults

The BitFlow Neon-CLB board requires a PCI Express x4 or wider slot. The Simulink Real-Time block library supports this board with the NEON BitFlow Image Input block.

Board Characteristics

Board name	BitFlow Neon-CLB
Manufacturer	BitFlow
Bus type	PCI Express
Multiple block instance support	No
Multiple board support	No

See Also

Topics

"Install BitFlow Neon-CLB Support" on page 26-17
"Camera Link Camera Triggering" on page 26-9

"Serial Camera Configuration" on page 26-15

External Websites

www.bitflow.com

Introduced in R2011a

NEON BitFlow Image Input

Support for NEON BitFlow Image Input block

Library

Simulink Real-Time Library for BitFlow Camera Link

Description

Use the NEON BitFlow Image Input block to acquire real-time video frames or still images from a Camera Link camera. Attach the camera to a BitFlow Neon-CLB camera interface board on the target computer. After you acquire the image, you can:

- Display the output on the target computer monitor using a Video Display block.
- Stream captured frames to the development computer display (for example, using the To Video Display block from Computer Vision System Toolbox).
- Analyze the image signals on the development computer.
- Compress or decompress the input signal with the JPEG Compression or JPEG Decompression blocks.

The **Image signal** type for the NEON BitFlow Image Input block output is `Single multidimensional signal`.

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 board supports 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.

If you cannot locate your camera in the list, contact BitFlow to find a camera that is compatible with your requirements.

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, if the camera is not on, the board emits an interrupt.

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 the following:

- `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 target computer system timer starts the model. 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`. A rising edge on the differential trigger input triggers the camera. 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`. A rising edge on the TTL trigger input triggers the camera. The model runs from the frame completion interrupt, as with `Differential Trigger`.
- `Opto Trigger`. A rising edge on the Opto trigger input triggers the camera. The model runs from the frame completion interrupt, as with `Differential Trigger`.

The camera configuration file determines whether you can use the block mask to switch the trigger mode to another mode, such as `Free Running`. You may need to contact BitFlow for a camera configuration file for the mode that 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 are 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.

Camera configuration serial command

Select the serial port configuration and specify a command to send to the camera when the model starts executing.

Baud Rate

Select one of the following: `9600`, `19200`, `38400`, `57600`, `115200`, `230400`, or `460800` baud.

Bits per Byte

Select one of the following: `5`, `6`, `7`, or `8`. Most camera configurations use 8 bits per byte.

Parity

Select one of the following: `None`, `Even`, `Odd`, `Mark`, or `Space`.

Stop Bits

Select either `1` or `2`.

Command String

Enter a command to send to your camera. The exact format of the command depends on the camera.

Display camera response string

Select this check box to display the camera response to the command. Nonprintable control codes are printed as C escape sequences.

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 real-time 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 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

Topics

"Install BitFlow Neon-CLB Support" on page 26-17
"Camera Link Camera Triggering" on page 26-9
"Serial Camera Configuration" on page 26-15

External Websites

www.bitflow.com

From USB Video Device

From USB Video Device block

Library

Simulink Real-Time 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. You attach the webcam to a USB port on the target computer. After you acquire the image, you can:

- Display the output on the target computer monitor using a Video Display block.
- Stream captured frames to the development computer display (for example, using the To Video Display block from Computer Vision System Toolbox).
- Analyze the image signals on the development 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.

The **Image signal** setting determines the **Image signal** setting for blocks receiving this signal, such as the Video Display block.

Note: When you execute a model containing a From USB Video Device block on a single-core target computer, insufficient time is sometimes available to process frames received through the USB port. Under these conditions, the block can drop frames. If the block is dropping frames, select a larger frame interval, lengthen the sample time, or use a multicore target computer.

Block Parameters

Configuration

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

Select the sample time between frame transfers:

- `1/60`
- `1/30`
- `1/25`
- `1/20`
- `1/15`
- `1/10`
- `1/7.5`
- `1/5`

Frame format

Select whether the incoming frames are to be compressed:

- `Uncompressed`

  Do not compress frames.

- `MJPEG`

  Compress frames using Motion JPEG format. Each frame is individually compressed as a JPEG image. Selecting this option disables the **Color format** and **Image signal** parameters.

Color format

Select 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.

Image signal

- `One multidimensional signal`

  One signal where each dimension contains color information. Selecting this option creates one port, Image.
- `Separate color signals`

  Multiple color signals where each signal contains the information for one color. Selecting this option creates the following ports, depending upon the colorspace.

  - RGB: ports R, G, B
  - YCbCr: ports Y, Cb, Cr

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

Topics

"Serial Camera Configuration" on page 26-15

Introduced in R2011a

Image Receive

Image Receive block

Library

Simulink Real-Time Library for Video Utilities

Description

The Image Receive block receives the video from the target computer at the IP address and port pair.

Block Outputs

Name	Description
Data	Vector of `byte` containing video data received by block.
Status	`1` if the block received a full fixed-length frame, and otherwise `0`. Visible when the **Allow variable length packets** check box is cleared (default).
Length	Number of bytes received. Visible when the **Allow variable length packets** check box is set.

Block Parameters

IP address to receive from (0.0.0.0 for accepting all)

Enter a valid IP address as a dotted decimal character vector 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 character vector.

The default address, 0.0.0.0, enables the acceptance of frames from any accessible target computer. If set to a specific IP address, packets arriving from only that IP address are received.

The **IP port to receive from** parameter specifies the port for the source.

IP port to receive from

Specify the port of the target computer from which to receive the video frames. The **IP address to receive from (0.0.0.0 for accepting all)** parameter specifies the IP address for the source.

Use the following local IP port (-1 for automatic port assignment)

Specify the port of the target computer receiving the video frames.

Enter `-1` to automatically assign a port for the target computer.

Allow variable length packets

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. Selecting this block changes the default Status output port to the Length output port.

Sample time

Enter the base sample time or a multiple of the base sample time (`-1` means sample time is inherited).

See Also

See Also

`Image Transmit`

Topics

"Serial Camera Configuration" on page 26-15

Introduced in R2011a

Image Transmit

Image Transmit block

Library

Simulink Real-Time Library for Video Utilities

Block Inputs

Name	Description
Data	Vector of `byte` containing video data for the block to transmit.
Length	Number of bytes to transmit. Visible when the **Allow variable length packets** check box is set.

Description

The Image Transmit block sends the video frame from the target computer to the development computer at the IP address and port pair.

Block Parameters

IP address sent to (255.255.255.255 for broadcast)

Specify the IP address of the target computer to which you send the video frames. To broadcast the video frames to all listening computers, enter `255.255.255.255`. The **Remote IP port to send to** parameter specifies the port for the destination.

Remote IP port to send to

Specify the target computer port to which you send the video frames. The **IP address sent to (255.255.255.255 for broadcast)** parameter specifies the IP address for the destination.

Use the following local IP port (-1 for automatic port assignment)

Specify the target computer port from which to send the video frames.

Enter `-1` to automatically assign a port for the target computer.

Allow variable length packets

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 Data port sends the actual data. The Length input port becomes visible and contains the number of bytes being sent. If the port size is less than Length, the block sends up to the port size.

If this check box is cleared (default), the block sends only fixed-length packets.

Sample time

Enter the base sample time or a multiple of the base sample time (`-1` means sample time is inherited).

See Also

See Also

`Image Receive`

Topics

"Serial Camera Configuration" on page 26-15

Introduced in R2011a

JPEG Compression

JPEG Compression block

Library

Simulink Real-Time Library for Video Utilities

Description

The JPEG Compression block compresses the video frame received by the target computer.

Block Inputs and Outputs

Inputs

Name	Description
Image	Video frame to compress

Outputs

Name	Description
Data	Vector of `byte` containing video data compressed by block.
Length	Number of bytes of compressed data. Visible when the **Show output image length** check box is set.

Block Parameters

Compression quality (-1:default)

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.

Enter `-1` to use the default compression quality for the video frame.

Input colorspace

One of the following:

- `Grayscale`

  Compress the video frame using a grayscale color space scheme.

- `YCbCr 4;4:4`

  Compress the video frame using the YCbCr color space scheme.

- `RGB`

  Compress the video frame using the red, blue, green (RGB) color space scheme.

Image signal

- `One multidimensional signal`

  One signal where each dimension contains color information. Selecting this option creates one port, Image.

- `Separate color signals`

  Multiple color signals where each signal contains the information for one color. Selecting this option creates the following ports, depending upon the colorspace.

  - Grayscale: port Image
  - RGB: ports R, G, B
  - YCbCr: ports Y, Cb, Cr

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

See Also

JPEG Decompression

Introduced in R2011a

JPEG Decompression

JPEG Decompression block

Library

Simulink Real-Time Library for Video Utilities

Description

The JPEG Decompression block decompresses the video frame received by the target computer.

Block Inputs and Outputs

Inputs

Name	Description
Data	Vector of `byte` containing video data for the block to decompress.
Trigger	Trigger for video decompression, active high. Visible when the **Show trigger input** check box is set.

Outputs

Name	Description
Image	Decompressed video frame

Block Parameters

Image width

Specify the width, in bytes, of the video frame to be decompressed.

Image height

Specify the height, in bytes, of the video frame to be decompressed.

Output colorspace

One of the following:

- `Grayscale`

  Compress the video frame using a grayscale color space scheme.
- `YCbCr 4;4:4`

  Compress the video frame using the YCbCr color space scheme.
- `RGB`

  Compress the video frame using the red, blue, green (RGB) color space scheme.

Image signal

- `One multidimensional signal`

  One signal where each dimension contains color information. Selecting this option creates one port, Image.
- `Separate color signals`

  Multiple color signals where each signal contains the information for one color. Selecting this option creates the following ports, depending upon the colorspace.

  - Grayscale: port Image
  - RGB: ports R, G, B
  - YCbCr: ports Y, Cb, Cr

Show trigger input

Select this check box to display an input port, Trigger, for the block.

See Also

See Also

`JPEG Compression`

Introduced in R2011a

USB Video Device List

USB Video Device List block

Library

Simulink Real-Time Library for USB Camera

Description

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 choose the configuration parameters required by your USB webcam, and then name the configuration for future use.

When you add the From USB Video Device block to your model, add the USB Video Device List block. You need the USB Video Device List block to configure the webcam.

Block Parameters

Manufacturer

Select the manufacturer for the installed webcam.

Format

Select an image format that the connected webcam supports, for example, `MJPEG`.

Resolution

Select a supported resolution for the image.

Interval

Select the sample time for the video frame.

Configurations

Use this parameter to store and name a particular configuration. A configuration consists of the settings that you select for a particular webcam.

After you select the options in the other parameters:

1. In the edit field, enter a name for the configuration.
2. Click the **Add** button to add the configuration.

To remove the configuration, click the **Remove** button.

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

See Also

`Image Receive`

Topics

"Serial Camera Configuration" on page 26-15

Introduced in R2011a

Video Display

Video Display block

Library

Simulink Real-Time Library for Video Utilities

Description

When you add the Video Display block to your model, you can display an RGB video signal on the target computer. The signal can come from a USB Video webcam, a Camera Link camera, or even a constant block.

The **Image signal** setting must match the **Image signal** setting for the camera output block.

There can be no more than two Video Display blocks in a model. The combined number of Video Display blocks and target scopes cannot exceed nine.

Block Parameters

Image signal

- `One multidimensional signal`

  One signal where each dimension contains color information. Selecting this option creates one port.

- `Separate color signals`

  Multiple color signals where each signal contains the information for one color. Selecting this option creates three ports.

Image colorspace

Can only be `RGB`.

See Also

See Also

"Target Scope Usage"

Introduced in R2014b

XCP Master Mode

XCP Master Mode

The Universal Measurement and Calibration Protocol (XCP) is a network protocol that you can use to connect calibration systems to electronic control units (ECUs).

A node in the network can run in either master mode or slave mode. Simulink Real-Time supports using XCP in master mode to replace (bypass) a subsystem of the ECU controller. The bypass model applies stimulus to the subsystem output signals and acquires signal response from the ECU controller.

To support XCP master mode, the Simulink Real-Time software provides the XCP sublibrary. You can:

- Parse A2L (ASAP2 database) files.
- Synchronize one or more slave or ECU devices.
- Initialize an XCP slave server running in an ECU.
- Apply stimulus data.
- Acquire real-time measurement data when specific events occur.

To create models to run in master mode:

- Provide an A2L (ASAP2) format file that contains signal and parameter access information for the slave ECUs and for the XCP-specific network elements.
- Provide an XCP Configuration block to load the A2L data into the XCP database.
- Provide one XCP CAN Transport Layer or XCP UDP Transport Layer block for each XCP Configuration block.

  Simulink Real-Time supports XCP implemented by using FIFO mode CAN or real-time UDP as transport protocols.
- Apply stimulus data to the slave device by using the XCP Data Stimulation block.
- Acquire measurement data from the slave device by using the XCP Data Acquisition block.

See Also

`XCP CAN Transport Layer` | `XCP Configuration` | `XCP Data Acquisition` | `XCP Data Stimulation` | `XCP UDP Transport Layer`

More About

- "CAN"
- "FIFO Mode" on page 6-2
- "Real-Time UDP"
- "Third-Party Calibration Support"

XCP Blocks

XCP CAN Transport Layer

Generate and consume XCP messages that are transported by CAN hardware

Library

Simulink Real-Time Library for XCP

Block Inputs and Outputs

Inputs

Name	Description
`CAN Msg`	Vector of CAN MESSAGE structures being consumed
`N`	Number of messages in the vector

Outputs

Name	Description
`CAN Msg`	Vector of CAN MESSAGE structures being generated
`N`	Number of messages in the vector

Description

The XCP CAN Transport Layer block handles CAN messages that your model transmits or receives with Simulink Real-Time CAN library blocks.

Connect the input side of the block to a block that receives CAN messages. Connect the output side of the block to a block that transmits the XCP messages over CAN. Set up the transmitting block so that a CAN message is sent only when an XCP message is available. Otherwise, the block sends `0` byte data when XCP messages are not available, causing undefined behavior.

See Also

See Also

`XCP Data Acquisition` | `XCP Configuration` | `XCP Data Stimulation`

External Websites

www.asam.net

Introduced in R2014a

XCP Configuration

Configure XCP slave connection

Library

XCP Communication

Description

The XCP Configuration block uses the parameters specified in the A2L file and the ASAP2 database to establish XCP slave connection.

Specify the A2L file to use in your XCP Configuration before you acquire or stimulate data. Use one XCP Configuration to configure one slave for data acquisition or stimulation. If you add Data Acquisition and Data Stimulation blocks, your model checks to see if there is a corresponding XCP Configuration block and will prompt you to add one.

Other Supported Features

The XCP communication blocks support the use of Simulink Accelerator and Rapid Accelerator mode. Using this feature, you can speed up the execution of Simulink models. For more information on this feature, see the Simulink documentation.

The XCP communication blocks also support code generation with limited deployment capabilities. Code generation requires the Microsoft C++ compiler.

Parameters

Config name

Specify a unique name for your XCP session.

A2L File

Click **Browse** to select an A2L file for your XCP session.

Enable seed/key security

Select this option if your slave requires a secure key to establish connection. You need to select a file that contains the seed/key definition to enable the security.

File (*.DLL)

This field is enabled if you select **Enable seed/key security**. Click **Browse** to select the file that contains seed and key security algorithm used to unlock an XCP slave module.

Output connection status

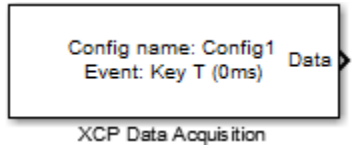
Select this option to display the status of the connection to the slave module. Selecting this option adds a new output port.

See Also

Introduced in R2013a

XCP Data Acquisition

Acquire selected measurements from configured slave



Library

XCP Communication

Description

The XCP Data Acquisition block acquires data from the configured slave based on the selected measurements. The block uses the XCP CAN transport layer to obtain raw data for the selected measurements at the specified simulation time step. Configure your XCP connection and use the XCP Data Acquisition block to select your event and measurements for the configured slave. The block displays the selected measurements as output ports.

Note A model with XCP Data Acquisition blocks does not disconnect from the XCP slave when the simulation ends. The model continues to acquire measurements until the data transmission from the XCP slave is terminated.

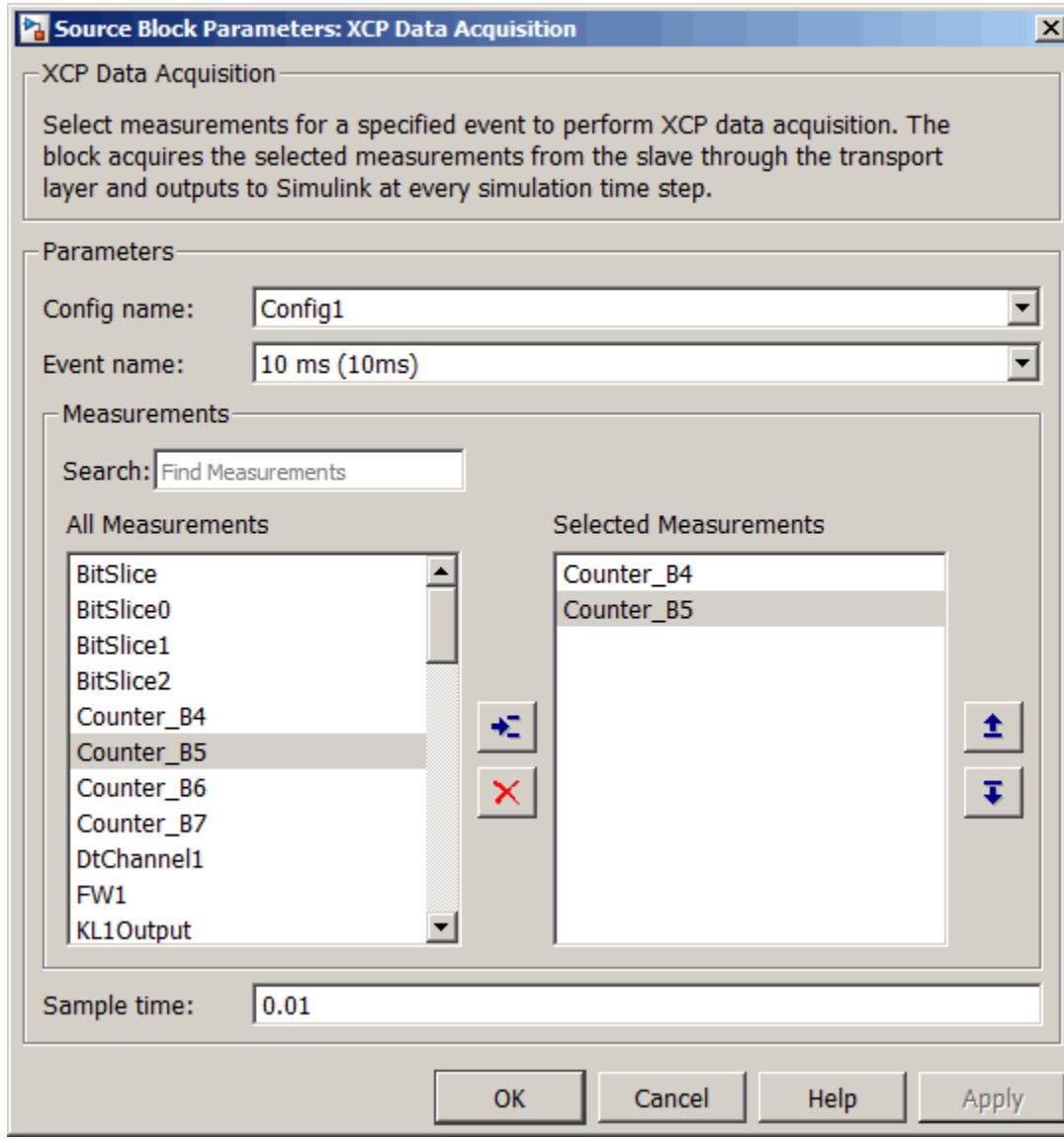
Other Supported Features

The XCP communication blocks support the use of Simulink Accelerator and Rapid Accelerator mode. Using this feature, you can speed up the execution of Simulink models. For more information on this feature, see the Simulink documentation.

The XCP communication blocks also support code generation with limited deployment capabilities. Code generation requires the Microsoft C++ compiler.

Dialog Box

Use the Block Parameters dialog box to select your data acquisition parameters.



Parameters

Config name

Select the name of XCP configuration you want to use. The list displays all available names specified in the available XCP Configuration blocks in the model. Selecting a configuration displays events and measurements available in this configuration's A2L file.

Note: You can acquire measurements for only one event using an XCP Data Acquisition block. Use one block each for each event whose measurements you want to acquire.

Event name

Select an event from the available list of events. The XCP Configuration block uses the specified A2L file to populate the events list.

Measurements

Search

Type the name of the measurement you want to use. The All Measurements lists displays a list of all matching terms. Click the x

to clear your search.

All Measurements

This list displays all measurements available for the selected event. Select the measurement you want to use and click the add button,  to add it to the selected measurements. Hold the `Ctrl` key on your keyboard to select multiple measurements.

Selected Measurements

This list displays selected measurements. To remove a measurement from this list, select the measurement and click the remove button,  .

Toggle buttons

Use the toggle buttons  to reorder the selected measurements.

Sample time

Specify the sampling time of the block during simulation, which is the simulation time as described by the Simulink documentation. This value defines the frequency at which the XCP Data Acquisition block runs during simulation. If the block is inside a triggered subsystem or to inherit sample time, you can specify –1 as your sample time. You can also specify a MATLAB variable for sample time. The default value is 0.01 (in seconds).

See Also

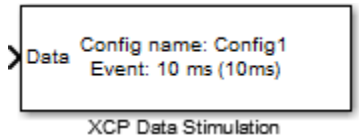
See Also

`XCP Configuration`

Introduced in R2013a

XCP Data Stimulation

Perform data stimulation on selected measurements



Library

XCP Communication

Description

The XCP Data Stimulation block sends data to the selected slave for the selected event measurements. The block uses the XCP CAN transport layer to output raw data for the selected measurements at the specified stimulation time step. Configure your XCP session and use the XCP Data Stimulation block to select your event and measurements on the configured slave. The block displays the selected measurements as input ports.

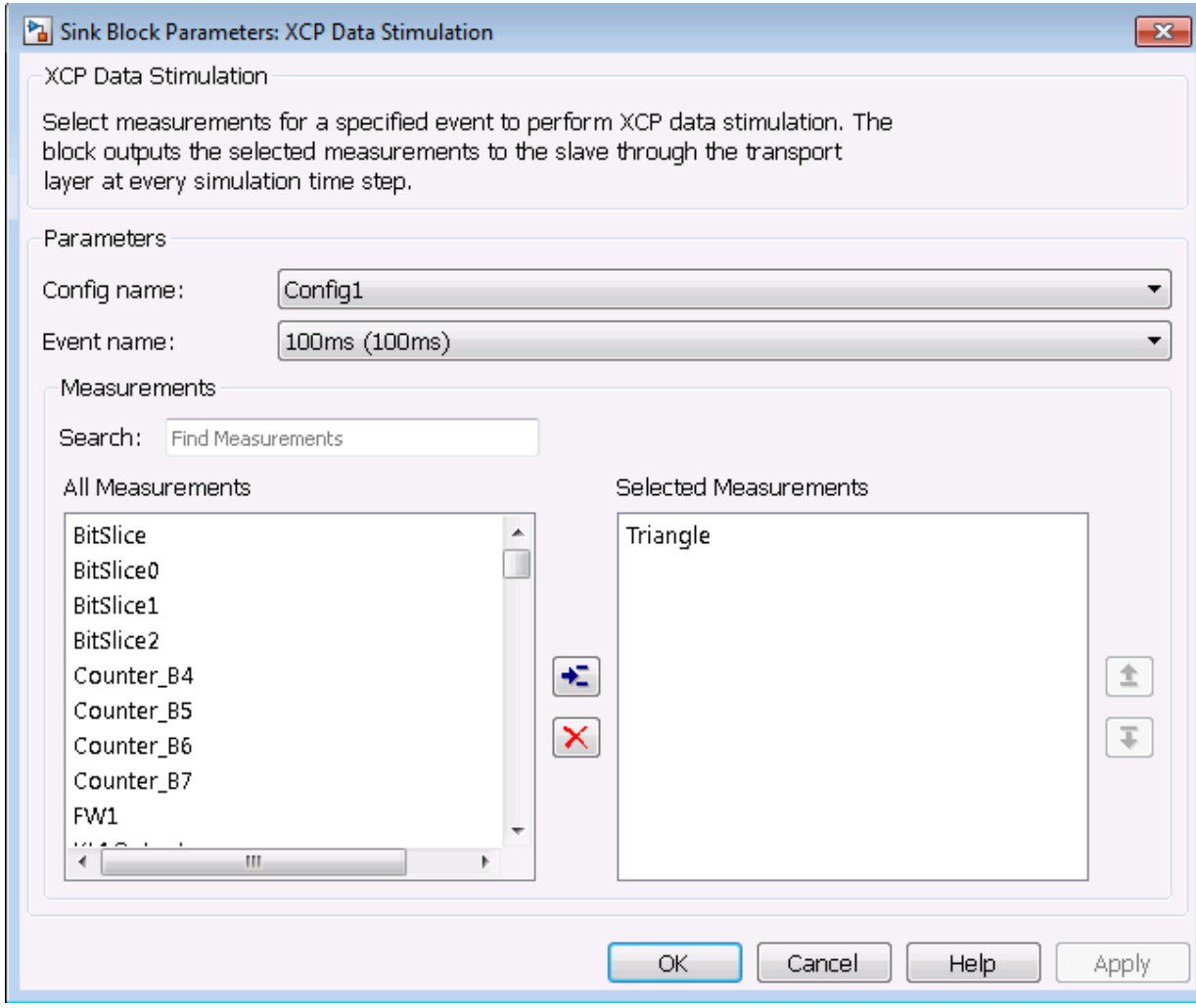
Other Supported Features

The XCP communication blocks support the use of Simulink Accelerator and Rapid Accelerator mode. Using this feature, you can speed up the execution of Simulink models. For more information on this feature, see the Simulink documentation.

The XCP communication blocks also support code generation with limited deployment capabilities. Code generation requires the Microsoft C++ compiler.

Dialog Box

Use the Block Parameters dialog box to select your data stimulation parameters.



Parameters

Config name

Select the name of XCP configuration you want to use. The list displays all available names specified in the available XCP Configuration blocks in the model. Selecting

a configuration displays events and measurements available in this configuration's A2L file.

Note: You can stimulate measurements for only one event using an XCP Data Stimulation block. Use one block each for each event whose measurements you want to stimulate.

Event name

Select an event from the available list of events. The XCP Configuration block uses the specified A2L file to populate the events list.

Measurements

Search

Type the name of the measurement you want to use. The All Measurements lists displays a list of all matching terms. Click the x

to clear your search.

All Measurements

This list displays all measurements available for the selected event. Select the measurement you want to use and click the add button,  to move it to the selected measurements. Hold the `Ctrl` key on your keyboard to select multiple measurements.

Selected Measurements

This list displays selected measurements. To remove a measurement from this list, select the measurement and click the remove button,  .

Toggle buttons

Use the toggle buttons  to reorder the selected measurements.

See Also

Introduced in R2013a

XCP UDP Transport Layer

Send and receive XCP messages over real-time UDP

Library

Simulink Real-Time Library for XCP

Description

The XCP UDP Transport Layer block uses the specified Ethernet device to send and receive XCP messages. Select the Ethernet device using the PCI bus and slot numbers.

Block Parameters

Master IP Address

Enter the IP address for the UDP master node.

Simulink Real-Time supports modeling the UDP master node only.

Port

UDP port that the transport layer uses.

Subnet mask

Enter the subnet mask for the interface.

Gateway

Enter the gateway address for the interface.

Ethernet Driver

Identifies the Ethernet driver for each chip family supported. Possible values are `Intel 8255X` and `Intel Gigabit`.

PCI bus

Enter the PCI bus number for the Ethernet card.

PCI slot

Enter the PCI slot number for the Ethernet card.

Disable CTR error detection

Select this check box to turn off bus controller error detection.

Sample time

Enter the base sample time or a multiple of the base sample time (`-1` means sample time is inherited).

See Also

See Also

`XCP Data Acquisition` | `XCP Configuration` | `XCP Data Stimulation`

External Websites

www.asam.net

Introduced in R2014a

Adlink

Adlink PCI-6208

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 Simulink Real-Time 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

See Also

External Websites

www.adlinktech.com

Adlink PCI-6208A Analog Output

Adlink PCI-6208A Analog Output block

Library

Simulink Real-Time Library for Adlink

Scaling Output to Input

I/O Module 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 arbitrary order. Number the channels beginning with 1 even if this 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:

```
Input = 10*sin(omega*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 manufacturer documentation, which cautions that 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 replicated over the channel vector. If you specify a value of 1, the corresponding channel is reset to the value specified in 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 current 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 replicated as initial value over the channel vector. 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 [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

Enter the base sample time or a multiple of the base sample time (`-1` means sample time is inherited).

See Also

External Websites

www.adlinktech.com

Adlink PCI-6208A Digital Input

Adlink PCI-6208A Digital Input

Library

Simulink Real-Time Library for Adlink

Scaling Input to Output

I/O Module 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 arbitrary order. Number the channels beginning with 1 even if this 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

Enter the base sample time or a multiple of the base sample time (`-1` means sample time is inherited).

PCI slot

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

External Websites

www.adlinktech.com

Adlink PCI-6208A Digital Output

Adlink PCI-6208A Digital Output

Library

Simulink Real-Time Library for Adlink

Scaling Input to Output

I/O Module Output	Block Input Data Type	Scaling
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 arbitrary order. Number the channels beginning with 1 even if this 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 replicated over the channel vector. If you specify a value of 1, the corresponding channel is reset to the value specified in 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 replicated as initial value over the channel vector. 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

Enter the base sample time or a multiple of the base sample time (`-1` means sample time is inherited).

PCI slot

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

External Websites

www.adlinktech.com

BVM, Contec, Curtiss-Wright

BVM

This topic describes the BVM I/O board supported by the Simulink Real-Time product (www.bvmltd.co.uk).

BVM PMCDIO64

BVM PMCDIO64 Digital Input

BVM PMCDIO64 Digital Output

BVM PMCDIO64

Support for 64–bit digital I/O

Board

BVM PMCDIO64

General Description

The PMCDIO64 is a 64-bit digital I/O board. The board 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 Simulink Real-Time block library supports this board with these driver blocks:

- BVM PMCDIO64 Digital Input
- BVM PMCDIO64 Digital Output

Board Characteristics

Board name	PMCDIO64
Manufacturer	BVM
Bus type	PCI
Access method	Memory mapped
Multiple block instance support	Yes
Multiple board support	Yes

See Also

External Websites

www.bvmltd.co.uk

BVM PMCDIO64 Digital Input

PMCDIO64 Digital Input block

Library

Simulink Real-Time Library for BVM

Scaling Input to Output

I/O Module Input	Block Output Data Type	Scaling
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 input voltage is low.

If the format is `Single 32 bit port`, then the output from the block is a 32-bit integer, where the 32 bits on the I/O module 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 manufacturer documentation.

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 if this manufacturer 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 (`-1` means sample time is inherited).

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

External Websites

www.bvmltd.co.uk

BVM PMCDIO64 Digital Output

PMCDIO64 Digital Output block

Library

Simulink Real-Time Library for BVM

Scaling Input to Output

I/O Module Output	Block Input Data Type	Scaling
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 output is set to low voltage if the input is < 0.5 and high voltage if the input is ≥ 0.5.

If the format is `Single 32 bit port`, then the input to the block is a 32-bit integer where the 32 bits on the I/O module 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 manufacturer documentation.

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 if this manufacturer labels them 0 to 31. 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, the 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 model download 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 scaler 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 (`-1` means sample time is inherited).

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

External Websites

www.bvmltd.co.uk

Contec

This topic describes the Contec I/O boards supported by the Simulink Real-Time product (`www.contec.com`).

Contec AD12-16(PCI)

Contec AD12-16(PCI) I/O board

Board

Contec AD12-16(PCI)

General Description

The Contec AD12-16(PCI) provides 16 single-ended or 8 differential analog input (A/D) channels (12 bit), 4 digital input lines, and 4 digital output lines.

The Simulink Real-Time 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 Simulink Real-Time 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

See Also

External Websites

www.contec.com

Contec AD12-16(PCI) Analog Input (A/D)

Contec AD12-16(PCI) Analog Input (A/D) block

Library

Simulink Real-Time Library for Contec

Scaling Input to Output

I/O Module Input	Block Output Data Type	Scaling
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 arbitrary order. Number them beginning with `1` even if this 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 I/O module converts all the channels between 1 and `n`, whether or not they occur in your channel vector. It is more efficient to specify a contiguous range of channels. (Permuting the order of such a range has little impact on efficiency however.)

Range vector

This board allows the range of each channel to be selected independently. Enter a scalar, in which case the same value is replicated over the channel vector, 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 is replicated over the channel vector.

Sample time

Enter the base sample time or a multiple of the base sample time (`-1` means sample time is inherited).

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

External Websites

www.contec.com

Contec AD12-16(PCI) Digital Input

Contec AD12-16(PCI) Digital Input block

Library

Simulink Real-Time Library for Contec

Scaling Input to Output

I/O Module Input	Block Output Data Type	Scaling
TTL	Double	TTL low = 0.0 TTL high = 1.0

Block Parameters

Channel vector

Enter a vector of numbers to specify the input channels. For example to use the first and third digital input channels enter

```
[1, 3]
```

The channel numbers can occur in arbitrary order. Number the channels beginning with 1 even if this 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 (`-1` means sample time is inherited).

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

External Websites

www.contec.com

Contec AD12-16(PCI) Digital Output

Contec AD12-16(PCI) Digital Output block

Library

Simulink Real-Time Library for Contec

Scaling Input to Output

I/O Module Output	Block Input Data Type	Scaling
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 arbitrary order. Number the channels beginning with 1 even if this 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 replicated over the channel vector. If you specify a value of `1`, the corresponding channel is reset to the value specified in the initial value vector. If you specify a value of `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 replicated as the initial value over the channel vector. 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 (`-1` means sample time is inherited).

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

External Websites

www.contec.com

Contec AD12-16(PCI)E

Contec AD12-16(PCI)E I/O board

Board

Contec AD12-16(PCI)E

General Description

The Contec AD12-16(PCI)E provides:

- 16 single-ended or 8 differential analog input (A/D) channels (12 bit)
- 1 analog output channel (12 bit)
- 4 digital input lines
- 4 digital output lines
- 1 i8254-compatible counter

The Simulink Real-Time 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 Simulink Real-Time 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

See Also

External Websites

www.contec.com

Contec AD12-16(PCI)E Analog Input (A/D)

Contec AD12-16(PCI)E Analog Input block

Library

Simulink Real-Time Library for Contec

Scaling Input to Output

I/O Module Input	Block Output Data Type	Scaling
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 arbitrary order. Number them beginning with 1 even if this 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 replicated over the channel vector.

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 value is replicated over the channel vector.

Sample time

Enter the base sample time or a multiple of the base sample time (`-1` means sample time is inherited).

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

External Websites

www.contec.com

Contec AD12-16(PCI)E Analog Output (D/A)

Contec AD12-16(PCI)E Analog Output block

Library

Simulink Real-Time Library for Contec

Scaling Input to Output

I/O Module Output	Block Input Data Type	Scaling
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 (`-1` means sample time is inherited).

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

External Websites

www.contec.com

Contec AD12-16U(PCI)E

Contec AD12-16U(PCI)E I/O board

Board

Contec AD12-16U(PCI)E

General Description

The Contec AD12-16U(PCI)E provides:

- 16 single-ended or 8 differential analog input (A/D) channels (12 bit)
- 1 analog output channel (12 bit)
- 4 digital input lines
- 4 digital output lines
- 1 i8254-compatible counter

The Simulink Real-Time 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 Simulink Real-Time 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

See Also

External Websites

www.contec.com

Contec AD12-16U(PCI)E Analog Input (A/D)

Contec AD12-16U(PCI)E Analog Input block

Library

Simulink Real-Time Library for Contec

Scaling Input to Output

I/O Module Input	Block Output Data Type	Scaling
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 arbitrary order. Number them beginning with 1 even if this 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 value is replicated over the channel vector.

Sample time

Enter the base sample time or a multiple of the base sample time (`-1` means sample time is inherited).

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

External Websites

www.contec.com

Contec AD12-16U(PCI)E Analog Output (D/A)

Contec AD12-16U(PCI)E Analog Output block

Library

Simulink Real-Time Library for Contec

Scaling Input to Output

I/O Module Output	Block Input Data Type	Scaling
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 (`-1` means sample time is inherited).

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

External Websites

www.contec.com

Contec AD12-64(PCI)

Contec AD12-64(PCI) I/O board

Board

Contec AD12-64(PCI)

General Description

The Contec AD12-64(PCI) provides 64 single-ended or 32 differential analog input (A/D) channels (12 bit), 4 digital input lines, and 4 digital output lines.

The Simulink Real-Time 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 Simulink Real-Time software does not support the Counter/Timer functionality of this board.

Board Characteristics

Board name	AD12-64(PCI)
Manufacturer	Contec
Bus type	PCI
Access method	I/O mapped
Multiple block instance support	No
Multiple board support	Yes

See Also

External Websites

www.contec.com

Contec AD12-64(PCI) Analog Input (A/D)

Contec AD12-64(PCI) Analog Input block

Library

Simulink Real-Time Library for Contec

Scaling Input to Output

I/O Module Input	Block Output Data Type	Scaling
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 arbitrary order. Number them beginning with `1` even if this 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 I/O module will convert all the channels between `1` and `n`, whether or not they occur in your channel vector. It is more efficient to specify a contiguous range of channels. (Permuting the order of such a range has little impact on efficiency however.)

Range vector

This board allows the range of each channel to be selected independently. If you enter a scalar, the value is replicated over the channel vector. 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 value is replicated over the channel vector.

Sample time

Enter the base sample time or a multiple of the base sample time (`-1` means sample time is inherited).

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

External Websites

www.contec.com

Contec AD12-64(PCI) Digital Input

Contec AD12-64(PCI) Digital Input block

Library

Simulink Real-Time Library for Contec

Scaling Input to Output

I/O Module Input	Block Output Data Type	Scaling
TTL	Double	TTL low = 0.0 TTL high = 1.0

Block Parameters

Channel vector

Enter a vector of numbers to specify the input channels. For example, to use the first and third digital input channels enter

```
[1, 3]
```

The channel numbers can occur in arbitrary order. Number the channels beginning with 1 even if this 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 (`-1` means sample time is inherited).

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

External Websites

www.contec.com

Contec AD12-64(PCI) Digital Output

Contec AD12-64(PCI) Digital Output block

Library

Simulink Real-Time Library for Contec

Scaling Input to Output

I/O Module Output	Block Input Data Type	Scaling
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 arbitrary order. Number the channels beginning with 1 even if this 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 replicated over the channel vector. If you specify a value of `1`, the corresponding channel is reset to the value specified in the initial value vector. If you specify a value of `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 replicated as the initial value over the channel vector. 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 (`-1` means sample time is inherited).

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

External Websites

www.contec.com

Contec AD16-16(PCI)E

Contec AD16-16(PCI)E I/O board

Board

Contec AD16-16(PCI)E

General Description

The Contec AD16-16(PCI)E provides:

- 16 single-ended or 8 differential analog input (A/D) channels (16 bit)
- 1 analog output channel (16 bit)
- 4 digital input lines
- 4 digital output lines
- 1 i8254-compatible counter

The Simulink Real-Time 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 Simulink Real-Time 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

See Also

External Websites

www.contec.com

Contec AD16-16(PCI)E Analog Input (A/D)

Contec AD16-16(PCI)E Analog Input block

Library

Simulink Real-Time Library for Contec

Scaling Input to Output

I/O Module Input	Block Output Data Type	Scaling
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 arbitrary order. Number them beginning with 1 even if this 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 value is replicated over the channel vector.

Sample time

Enter the base sample time or a multiple of the base sample time (`-1` means sample time is inherited).

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

External Websites

www.contec.com

Contec AD16-16(PCI)E Analog Output (D/A)

Contec AD16-16(PCI)E Analog Output block

Library

Simulink Real-Time Library for Contec

Scaling Input to Output

I/O Module Output	Block Input Data Type	Scaling
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 (`-1` means sample time is inherited).

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

External Websites

www.contec.com

Contec ADI12-16(PCI)

Contec ADI12-16(PCI) I/O board

Board

Contec ADI12-16(PCI)

General Description

The Contec ADI12-16(PCI) provides 16 single-ended or 8 differential analog input (A/D) channels (12 bit), 4 digital input lines, and 4 digital output lines.

The Simulink Real-Time block library supports this board with one 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

See Also

External Websites

www.contec.com

Contec ADI12-16(PCI) Analog Input (A/D)

Contec ADI12-16(PCI) Analog Input block

Library

Simulink Real-Time Library for Contec

Scaling Input to Output

I/O Module Output	Block Input Data Type	Scaling
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 arbitrary order. Number them beginning with 1 even if this 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 replicated over the channel vector.

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 value is replicated over the channel vector.

Sample time

Enter the base sample time or a multiple of the base sample time (`-1` means sample time is inherited).

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

External Websites

www.contec.com

Contec CNT24-4D(PCI)

Contec CNT24-4D(PCI) counter board

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 Simulink Real-Time block library supports this board with one driver block:

- Contec CNT24-4D(PCI) Incremental Encoder

The Simulink Real-Time 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

See Also

External Websites

www.contec.com

Contec CNT24-4D(PCI) Incremental Encoder

Contec CNT24-4D(PCI) Incremental Encoder block

Library

Simulink Real-Time Library for Contec

Block Parameters

You can have only one driver block instance for each physical board. 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.

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** does nothing.

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 (`-1` means sample time is inherited).

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

External Websites

www.contec.com

Contec CNT32-8M(PCI)

Contec CNT32-8M(PCI) board

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 Simulink Real-Time 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

See Also

External Websites

www.contec.com

Contec CNT32-8M(PCI) Incremental Encoder

Contec CNT32-8M(PCI) Incremental Encoder block

Library

Simulink Real-Time Library for Contec

Note

Each channel of this block has an external input and output signal. You can configure these channels to trigger events (such as starting the encoder on the rising or falling edge of an input signal). You can also configure them for general-purpose I/O, where the signals behave like digital I/O signals. The block can have the following ports:

- Output port labeled with the channel number
- If you configure the block for digital input, an output port labeled `din`.
- If you configure the block for digital output, an input port labeled `dout`.

Scaling

Encoder (Input to Output)

I/O Module Input	Block Input Data Type	Scaling
TTL or differential (pulse input)	Double	Counts

Digital Input (Input to Output)

I/O Module Input	Block Input Data Type	Scaling
TTL	Double	TTL low = 0.0 TTL high = 1.0

Digital Output (Output to Input)

I/O Module Output	Block Output Data Type	Scaling
TTL	Double	TTL low = < 0.5 TTL high = > 0.5

Block Parameters: Main Tab

The **Main** tab contains parameters that define the 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 if this 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. To disable digital filtering, select `No filtering`.

Pulse input

From the list, select one of the following pulse inputs:

- `Differential line receiver input`
- `TTL Level input`

  You can connect the CNT32-8M(PCI) board to either differential or TTL input signals. See the Contec CNT32-8M(PCI) 8-Ch 32-Bit Up/Down High-Speed Counter Board for PCI documentation for connection details.

Direction

From the list, select one of the following count directions:

- `Up in the clockwise direction/Down in the counterclockwise direction`
- `Down in the clockwise direction/Up in the counterclockwise direction`

  The board can count either up or down in the clockwise direction.

Initial value

Enter a number from 0 to the largest 32-bit number. This parameter specifies the initial value of the counter before the model begins.

Load initial value

Select this check box to load the value from **Initial value** into the counter, overwriting the previous count value. If you do not select this check box, the driver does not load the initial value, preserving the previous count value.

Sample time

Enter the base sample time or a multiple of the base sample time (`-1` means sample time is inherited).

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

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.
- `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.

Match 0 value

Enter the value that you want the counter value to be compared to for the **Zero-clear match 0** or **Preset match 0** conditions.

Match 1 value

Enter the value that you want the counter value to be compared to for the **Zero-clear match 1** or **Preset match 1** conditions.

Block Parameters: Signals Tab

The **Signals** tab contains parameters that

- Control the optional use of the external output signal
- Control the encoder index signal

- Enable the external signals for general purpose

Output signal match 0

Select this check box to strobe the output signal when the counter value has the same value as the **Match 0 parameter**.

Output signal match 1

Select this check box to strobe the output signal when the counter value has the same value as the **Match 1 parameter**.

Output signal abnormal input error

Select this check box to strobe the output signal when the phase-A and phase-B signals change at the same time.

Output signal digital filter error

Select this check box to strobe the output signal when a pulse is faster than the digital filter time setting.

Output signal disconnection alarm error

Select this check box to strobe the output signal when both the positive and negative differential inputs are high.

One-shot duration

From the list, select the pulse width of the output signal for the one-shot duration, in microseconds or milliseconds.

Index enable

From the list, select the following to enable or disable the phase-Z/CLR input signal:

- `Disable` — Disables phase-Z input (select this option if phase-Z does not exist).
- `Enable Once` — Enables only the next phase-Z input.
- `Enable All` — Enables all of the 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.

See Also

External Websites

www.contec.com

Contec DA12-16(PCI)

Contec DA12-16(PCI) I/O block

Board

Contec DA12-16(PCI)

General Description

The Contec DA12-16(PCI) provides 16 analog output (D/A) channels (12 bit).

The Simulink Real-Time block library supports this board with this driver block:

- Contec DA12-16(PCI) Analog Output (D/A)

The Simulink Real-Time software does not support the timer, external trigger, or interrupt functionality of this board.

Board Characteristics

Board name	DA12-16(PCI)
Manufacturer	Contec
Bus type	PCI
Access method	I/O mapped
Multiple block instance support	No
Multiple board support	Yes

See Also

External Websites

www.contec.com

Contec DA12-16(PCI) Analog Output (D/A)

Contec DA12-16(PCI) Analog Output block

Library

Simulink Real-Time Library for Contec

Scaling Input to Output

I/O Module Output	Block Input Data Type	Scaling
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 arbitrary order. Number the channels beginning with 1 even if this 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 is replicated over the analog output channel vector, 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
-10 to +10	-10
-5 to +5	-5

Input Range (V)	Range Code
-2.5 to 2.5	-2

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 replicated over the channel vector. If you specify a value of `1`, the corresponding channel is reset to the value specified in the initial value vector. If you specify a value of `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 replicated as the initial value over the channel vector. 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 (`-1` means sample time is inherited).

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

External Websites

www.contec.com

Contec DA12-4(PCI)

Contec DA12-4(PCI) I/O board

Board

Contec DA12-4(PCI)

General Description

The Contec DA12-4(PCI) provides 4 analog output (D/A) channels (12 bit).

The Simulink Real-Time block library supports this board with this driver block:

- Contec DA12-4(PCI) Analog Output (D/A)

The Simulink Real-Time 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

See Also

External Websites

www.contec.com

Contec DA12-4(PCI) Analog Output (D/A)

Contec DA12-4(PCI) Analog Output block

Library

Simulink Real-Time Library for Contec

Scaling Input to Output

I/O Module Output	Block Input Data Type	Scaling
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 arbitrary order. Number the channels beginning with 1 even if this 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 replicated over the channel vector. If you specify a value of `1`, the corresponding channel is reset to the value specified in the initial value vector. If you specify a value of `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 replicated as the initial value over the channel vector. 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 (`-1` means sample time is inherited).

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

External Websites

www.contec.com

Contec PI-64L(PCI)H

Contec PI-64L(PCI)H I/O board

Board

Contec PI-64L(PCI)H

General Description

The Contec PI-64L(PCI)H is a digital input board that provides 64 digital input channels. The supporting block can control up to 32 bits. You can configure the 32 bits 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 Simulink Real-Time block library supports this board with one driver block:

- Contec PI-64L(PCI)H Digital Input

Board Characteristics

Board name	PI-64L(PCI)H
Manufacturer	Contec
Bus type	PCI
Multiple block instance support	No
Multiple board support	Yes

See Also

External Websites

www.contec.com

Contec PI-64L(PCI)H Digital Input

Contec PI-64L(PCI)H Digital Input block

Library

Simulink Real-Time Library for Contec

Scaling Input to Output

I/O Module Input	Block Output Data Type	Scaling
TTL	Double (**Format**: `32 1-bit Channels`)	Double: TTL low = 0.0 TTL high = 1.0
	uint32 (**Format**: `Single 32-bit Port`)	uint32: TTL low corresponding bit is clear TTL high corresponding bit is set

Block Parameters

Format

From the list, select either

- `32 1-bit Channels` — Indicates that the channel vector specifies the configuration of the block. Each output is a double with value of 0 or 1. The value is 0 when the 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.

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 arbitrary 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 (`-1` means sample time is inherited).

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

External Websites

www.contec.com

Contec PIO-32/32F(PCI)

Contec PIO-32/32F(PCI) I/O board

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 Simulink Real-Time block library supports this board with these driver blocks:

- Contec PIO-32/32T(PCI) Digital Input
- Contec PIO-32/32T(PCI) Digital Output

The Simulink Real-Time 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

See Also

External Websites

www.contec.com

Contec PIO-32/32F(PCI) Digital Input

Contec PIO-32/32F(PCI) Digital Input block

Library

Simulink Real-Time Library for Contec

Scaling Input to Output

I/O Module Input	Block Output Data Type	Scaling
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 arbitrary 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 (`-1` means sample time is inherited).

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

External Websites

www.contec.com

Contec PIO-32/32F(PCI) Digital Output

Contec PIO-32/32F(PCI) Digital Output block

Library

Simulink Real-Time Library for Contec

Scaling Input to Output

I/O Module Output	Block Input Data Type	Scaling
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 arbitrary 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 replicated over the channel vector. If you specify a value of `1`, the corresponding channel is reset to the value specified in the initial value vector.

If you specify a value of `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 replicated as the initial value over the channel vector. 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 (`-1` means sample time is inherited).

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

External Websites

www.contec.com

Contec PIO-32/32L(PCI)H

Contec PIO-32/32L(PCI)H I/O board

Board

Contec PIO-32/32L(PCI)H

General Description

The Contec PIO-32/32L(PCI)H provides 32 digital input channels and 32 output channels. Each supporting block can control up to 32 bits. You can configure the 32 bits 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 Simulink Real-Time 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 Simulink Real-Time 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

See Also

External Websites

www.contec.com

Contec PIO-32/32L(PCI)H Digital Input

Contec PIO-32/32L(PCI)H Digital Input block

Library

Simulink Real-Time Library for Contec

Scaling Input to Output

I/O Module Input	Block Output Data Type	Scaling
TTL	Double (**Format**: `32 1-bit Channels`) uint32 (**Format**: `Single 32-bit Port`)	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 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 manufacturer 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.

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 arbitrary 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 (`-1` means sample time is inherited).

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

External Websites

www.contec.com

Contec PIO-32/32L(PCI)H Digital Output

Contec PIO-32/32L(PCI)H Digital Output block

Library

Simulink Real-Time Library for Contec

Scaling Input to Output

I/O Module Output	Block Input Data Type	Scaling
TTL	Double (**Format**: `32 1-bit Channels`)	Double: < 0.5 = TTL low
	uint32 (**Format**: `Single 32-bit Port`)	>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 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 manufacturer 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.

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 arbitrary 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 replicated as the initial value over the channel vector. 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 replicated over the channel vector. If you specify 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 replicated over the channel vector. 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 (`-1` means sample time is inherited).

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

External Websites

www.contec.com

Contec PIO-32/32T(PCI)

Contec PIO-32/32T(PCI) I/O board

Board

Contec PIO-32/32T(PCI)

General Description

The Contec PIO-32/32T(PCI) provides 32 digital input channels and 32 output channels.

The Simulink Real-Time block library supports this board with these driver blocks:

- Contec PIO-32/32T(PCI) Digital Input
- Contec PIO-32/32T(PCI) Digital Output

The Simulink Real-Time 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

See Also

External Websites

www.contec.com

Contec PIO-32/32T(PCI) Digital Input

Contec PIO-32/32T(PCI) Digital Input block

Library

Simulink Real-Time Library for Contec

Scaling Input to Output

I/O Module Input	Block Output Data Type	Scaling
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 arbitrary 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 (`-1` means sample time is inherited).

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

External Websites

www.contec.com

Contec PIO-32/32T(PCI) Digital Output

Contec PIO-32/32T(PCI) Digital Output block

Library

Simulink Real-Time Library for Contec

Scaling Input to Output

I/O Module Output	Block Input Data Type	Scaling
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 arbitrary 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 replicated over the channel vector. If you specify a value of `1`, the corresponding channel is reset to the value specified in the initial value vector.

If you specify a value of `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 replicated as the initial value over the channel vector. 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 (`-1` means sample time is inherited).

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

External Websites

www.contec.com

Contec PO-64L(PCI)H

Contec PO-64L(PCI)H board

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. You can configure the 32 bits 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 Simulink Real-Time block library supports this board with one 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

See Also

External Websites

www.contec.com

Contec PO-64L(PCI)H Digital Output

Contec PO-64L(PCI)H Digital Output block

Library

Simulink Real-Time Library for Contec

Scaling Input to Output

I/O Module Output	Block Input Data Type	Scaling
TTL	Double (**Format**: `32 1-bit Channels`) uint32 (**Format**: `Single 32-bit Port`)	Double: < 0.5 = TTL low >0.5 = TTL high uint32: Bit clear = TTL low Bit set = TTL high

Block Parameters

Format

From the list, select either

- `32 1-bit Channels` — Indicates that the channel vector specifies the configuration of the block. Each output is a double with value of 0 or 1. The value is 0 when the 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 model 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 manufacturer 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.

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 arbitrary 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 replicated as the initial value over the channel vector. 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 replicated over the channel vector. If you specify 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 replicated over the channel vector. 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 (`-1` means sample time is inherited).

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

External Websites

www.contec.com

Curtiss-Wright Electronic Systems

Curtiss-Wright Electronic Systems Drivers

Systran SCRAMNet+ SC150 PCI

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 Simulink Real-Time 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 endianess block, see Byte Reversal/Change Endianess.

Board Characteristics

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

See Also

External Websites

www.cwcelectronicsystems.com

Systran SC150 init

SystranSC150 init block

Library

Simulink Real-Time Library for Systran

Note

Before you begin to configure these block parameters, be sure that you have a predefined node initialization structure. See “Initialize Curtiss-Wright Shared Nodes” on page 25-15.

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 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

External Websites

www.cwcelectronicsystems.com

Systran SC150 read

Systran SC150 read block

Library

Simulink Real-Time 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 Curtiss-Wright Shared Partitions” on page 25-13.

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 (`-1` means sample time is inherited)

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

External Websites

www.cwcelectronicsystems.com

Systran SC150 rearm

Systran SC150 rearm block

Library

Simulink Real-Time 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 Curtiss-Wright Shared Partitions” on page 25-13.

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 (`-1` means sample time is inherited)

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

External Websites

www.cwcelectronicsystems.com

Systran SC150 write

Systran SC150 write block

Library

Simulink Real-Time 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 Curtiss-Wright Shared Partitions” on page 25-13.

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 (`-1` means sample time is inherited)

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

External Websites

www.cwcelectronicsystems.com

GE Intelligent Systems

GE Intelligent Platforms Blocks

Simulink Real-Time supports boards manufactured by Abaco, formerly GE Intelligent Platforms and GE Fanuc, which acquired the intellectual property of SBS Technologies, VMIC, and Condor Engineering. See the manufacturer documentation (`www.abaco.com`).

Condor (GE-IP) RCEI-530

Support for ARINC 429 RCEI-530 and CEI-530 boards

Board

Condor (GE-IP) RCEI-530, Condor (GE-IP) CEI-530

General Description

The commercial and aircraft transport industry uses the ARINC 429 protocol. The ARINC-429 driver library allows real-time 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 Simulink Real-Time software supports the ARINC 429 protocol with the RCEI-530 and CEI-530 boards. This board type interfaces a target computer to an ARINC 429 data bus and supports the PCI bus.

The Simulink Real-Time block library provides the following driver blocks to support the RCEI-530 and CEI-530 boards:

- Condor (GE-IP) RCEI-530 Initialize
- Condor (GE-IP) RCEI-530 Send
- Condor (GE-IP) RCEI-530 Receive

Use the following utility blocks to format the data sent to and received from the RCEI-530 Send and Receive blocks:

- Condor Encode ARINC 429 Words for Send
- Condor Decode ARINC 429 Words from Receive

Board Characteristics

Board name	RCEI-530, CEI-530

Manufacturer	GE Fanuc (formerly Condor Engineering)
Bus type	PCI
Access method	Memory mapped
Multiple block instance support	Yes
Multiple board support	Yes

See Also

External Websites

www.aviation-ia.com
www.abaco.com

Condor (GE-IP) RCEI-530 Initialize

Condor RCEI-530 Initialize block, Condor CEI-530 Initialize block

Library

Simulink Real-Time Library for Arinc-429

Note

Your model must include a Condor (GE-IP) RCEI-530 Initialize block for every physical board in the model. To identify the physical board to which your Send and Receive blocks refer, configure them with the **Board ID** value from this block.

Block Parameters

Board ID

From the list, select from 1 to 16 a unique ID for the CEI board. Use the same ID to identify the CEI board in the Initialize, Send, and Receive blocks in your model. This ID is drawn from a pool shared by the RCEI-530, CEI-530, RCEI-830A, and CEI-830 series boards.

Wrap each send channel to the corresponding receive channel

Select this check box to enable the loopback feature. If you select this block, each word sent over the output channel `n` is 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. If you select time tags from a Decode ARINC 429 Words from Receive block, the time tags are provided in units of timer ticks.

Sample time

Enter the base sample time or a multiple of the base sample time (`-1` means sample time is inherited).

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

External Websites

www.aviation-ia.com
www.abaco.com

Condor (GE-IP) RCEI-530 Send

Condor RCEI-530 Send block, Condor CEI-530 Send block

Library

Simulink Real-Time 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. You can provide multiple Send blocks for each channel.

Block Parameters

Board ID

From the list, select from 1 to 16 a unique ID for the CEI board. Use the same ID to identify the CEI board in the Initialize, Send, and Receive blocks in your model. This ID is drawn from a pool shared by the RCEI-530, CEI-530, RCEI-830A, and CEI-830 series boards.

Channel

From the list, select a channel ID. This number varies depending on the particular board that you are using. Refer to 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`

Corresponding Send and Receive blocks must have the same baud rate setting for this channel.

Parity

From the list, select:

- `odd`
- `none`

Corresponding Send and Receive blocks must have the same parity setting for this channel.

Sample time

Enter the base sample time or a multiple of the base sample time (`-1` means sample time is inherited).

See Also

External Websites

www.aviation-ia.com
www.abaco.com

Condor (GE-IP) RCEI-530 Receive

Condor RCEI-530 Receive block, Condor CEI-530 Receive block

Library

Simulink Real-Time 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.

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 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 that do not interpret the data. However, before feeding this port into a block such as a real-time Scope block that does interpret the data, you must first pass it through an ARINC Decode block.

Block Parameters

Board ID

From the list, select from 1 to 16 a unique ID for the CEI board. Use the same ID to identify the CEI board in the Initialize, Send, and Receive blocks in your model. This ID is drawn from a pool shared by the RCEI-530, CEI-530, RCEI-830A, and CEI-830 series boards.

Channel

From the list, select a channel ID. This number varies depending on the particular board that you are using. Refer to 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.

Max number of words to return

Enter the maximum number of words to extract from the receive buffer on the I/O module. This number is the maximum number of words for the selected channel at each sample time.

If you select *n*, the output port of the block has a signal width of *n+1*. The first signal element contains 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 receive buffer on the I/O module. This number is the minimum number of words for the selected channel at each sample time.

If the receive buffer does not contain this minimum number of words for the selected channel during the current sample time, the block does not extract words from the buffer. The word count associated with the output port is then `0`.

Baud rate

From the list, select:

- `12.5 Kbits/sec`
- `100 Kbits/sec`

Corresponding Send and Receive blocks must have the same baud rate setting for this channel.

Parity

From the list, select:

- `odd`
- `none`

Corresponding Send and Receive blocks must have the same parity setting for this channel.

Sample time

Enter the base sample time or a multiple of the base sample time (`-1` means sample time is inherited).

See Also

External Websites

www.aviation-ia.com
www.abaco.com

Condor (GE-IP) RCEI-830A

Support for ARINC 429 RCEI-830A and CEI-830 boards

Board

Condor (GE-IP) RCEI-830A, Condor (GE-IP) CEI-830

General Description

The commercial and aircraft transport industry uses the ARINC 429 protocol. The ARINC-429 driver library allows real-time 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 Simulink Real-Time software supports the ARINC 429 protocol with the RCEI-830A and CEI-830 boards. This board type interfaces a target computer to an ARINC 429 data bus and supports the PCI bus.

The Simulink Real-Time block library provides the following driver blocks to support the RCEI-830A and CEI-830 boards:

- Condor (GE-IP) RCEI-830A Initialize
- Condor (GE-IP) RCEI-830A Send
- Condor (GE-IP) RCEI-830A Receive

Use the following utility blocks to format the data sent to and received from the CEI-830 Send and Receive blocks:

- Condor Encode ARINC 429 Words for Send
- Condor Decode ARINC 429 Words from Receive

Board Characteristics

Board name	RCEI-830A, CEI-830

Manufacturer	GE Fanuc (formerly Condor Engineering)
Bus type	PCI
Access method	Memory mapped
Multiple block instance support	Yes
Multiple board support	Yes

See Also

External Websites

www.aviation-ia.com
www.abaco.com

Condor (GE-IP) RCEI-830A Initialize

Condor RCEI-830A Initialize block, Condor CEI-830 Initialize block

Library

Simulink Real-Time Library for Arinc-429

Note

Your model must include a Condor (GE-IP) RCEI-830A Initialize block for every physical board in the model. To identify the physical board to which your Send and Receive blocks refer, configure them with the **Board ID** value from this block.

Block Parameters

Board ID

From the list, select from 1 to 16 a unique ID for the CEI board. Use the same ID to identify the CEI board in the Initialize, Send, and Receive blocks in your model. This ID is drawn from a pool shared by the RCEI-530, CEI-530, RCEI-830A, and CEI-830 series boards.

Wrap each send channel to the corresponding receive channel

Select this check box to enable the loopback feature. If you select this block, each word sent over the output channel `n` is 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. If you select time tags from a Decode ARINC 429 Words from Receive block, the time tags are provided in units of timer ticks.

Sample time

Enter the base sample time or a multiple of the base sample time (`-1` means sample time is inherited).

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

External Websites

www.aviation-ia.com
www.abaco.com

Condor (GE-IP) RCEI-830A Send

Condor RCEI-830A Send block,Condor CEI-830 Send block

Library

Simulink Real-Time 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. You can provide multiple Send blocks for each channel.

Block Parameters

Board ID

From the list, select from 1 to 16 a unique ID for the CEI board. Use the same ID to identify the CEI board in the Initialize, Send, and Receive blocks in your model. This ID is drawn from a pool shared by the RCEI-530, CEI-530, RCEI-830A, and CEI-830 series boards.

Channel

From the list, select a channel ID. This number varies depending on the particular board that you are using. Refer to 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`

Corresponding Send and Receive blocks must have the same baud rate setting for this channel.

Parity

From the list, select:

- `odd`
- `none`

Corresponding Send and Receive blocks must have the same parity setting for this channel.

Sample time

Enter the base sample time or a multiple of the base sample time (`-1` means sample time is inherited).

See Also

External Websites

www.aviation-ia.com
www.abaco.com

Condor (GE-IP) RCEI-830A Receive

Condor RCEI-830A Receive block, Condor CEI-830 Receive block

Library

Simulink Real-Time 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.

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 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 that do not interpret the data. However, before feeding this port into a block such as a real-time Scope block that does interpret the data, you must first pass it through an ARINC Decode block.

Block Parameters

Board ID

From the list, select from 1 to 16 a unique ID for the CEI board. Use the same ID to identify the CEI board in the Initialize, Send, and Receive blocks in your model. This ID is drawn from a pool shared by the RCEI-530, CEI-530, RCEI-830A, and CEI-830 series boards.

Channel

From the list, select a channel ID. This number varies depending on the particular board that you are using. Refer to 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.

Max number of words to return

Enter the maximum number of words to extract from the receive buffer on the I/O module. This number is the maximum number of words for the selected channel at each sample time.

If you select *n*, the output port of the block has a signal width of *n+1*. The first signal element contains 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 receive buffer on the I/O module. This number is the minimum number of words for the selected channel at each sample time.

If the receive buffer does not contain this minimum number of words for the selected channel during the current sample time, the block does not extract words from the buffer. The word count associated with the output port is then `0`.

Baud rate

From the list, select:

- `12.5 Kbits/sec`
- `100 Kbits/sec`

Corresponding Send and Receive blocks must have the same baud rate setting for this channel.

Parity

From the list, select:

- `odd`
- `none`

Corresponding Send and Receive blocks must have the same parity setting for this channel.

Sample time

Enter the base sample time or a multiple of the base sample time (`-1` means sample time is inherited).

See Also

External Websites

www.aviation-ia.com
www.abaco.com

Condor Encode ARINC 429 Words for Send

Encode ARINC 429 Words for Send block

Library

Simulink Real-Time 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 (GE-IP) RCEI-530 Send`and `Condor (GE-IP) RCEI-830A Send` blocks — The ARINC Send block accepts data as a double.
- 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 without further processing.

Type	Interpretation
1	BNR (two's complement binary notation) — Scale the input by dividing it by the scale vector in the **Resolution (BCD) or Scale (BNR) vector** parameter. This input value is restricted to the range `[-Scale, Scale]`, resulting in a scaled value in the range `[-1,1]`. The driver performs the following on the scaled value: **1** Multiplies the scaled value by 2^18. **2** Truncates the value to a 19-bit fixed-point integer, then masks it to the number of high-order bits specified by the **Bits to send** parameter. **3** Shifts the result up 10 bits and inserts them into the 32-bit integer along with the **SSM**, **SDI**, and **Label** parameter values. To place discrete bits into unused positions, construct the 32-bit word and use it with RAW mode instead of the BNR mode.
2	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 **SSM**, **SDI**, and **Label** parameter values.
3	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 **SSM**, **SDI**, and **Label** parameters.

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 the resolution value. However, you must still include an associated value in the resolution vector.
`BNR`	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`.

Type	Effect
	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.
`BCD`	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. 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.
`Discretes`	The block ignores the resolution value. However, you must still include an associated value in the resolution vector.

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.

Type	Effect
`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 does not perform SDI processing on the corresponding output word.

SSM vector

Enter a vector or scalar value as the SSM vector. This vector 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 does not perform SSM processing on the corresponding output word. The meaning of a given SSM value differs depending on the data type of the ARINC word.

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 does not perform SSM processing on the corresponding output word.

See Also

External Websites

www.aviation-ia.com
www.abaco.com

Condor Decode ARINC 429 Words from Receive

Decode ARINC 429 Words from Receive block

Library

Simulink Real-Time Library for Arinc-429

Note

The input port of a Decode block is a signal of type double. The Decode block input port width automatically adapts to that of the source block.

Because the Decode block interprets the data on this port in a nonstandard way, you send data to this port from the `Condor (GE-IP) RCEI-530 Receive` and `Condor (GE-IP) RCEI-830A Receive` blocks. These blocks output their data into standard double output.

The output port of a Decode block is in standard double format.

Block Parameters

Label

Enter a three digit octal number. If the label of an input word does not match this label, the block completely ignores the word and does not apply the **Sync mask** and **Sync value** parameters.

Data type vector

Enter a vector consisting of values between 0 and 3. These values specify the data type. The length of this vector determines how many ARINC words the block will attempt to decode and output each sample time.

The elements of the vector determine how the input double value is converted to a corresponding double output value, as follows

Type	Interpretation
0	Raw — Convert the entire (unsigned) 32–bit input word to double.

Type	Interpretation
1	BNR — For each word, convert bits 10–28 from signed binary format to a double output in the range `[-Scale, Scale]` using the **Resolution (BCD) or Scale (BNR) vector** 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	Discretes — For each word, extract bits 10–28 and return them as a double.

The elements of the data type vector determine how the input ARINC value is converted to a corresponding double output. The following describes how this block performs the conversion. For the purposes of this description, `n` denotes the length of the data type vector.

- The output width is one of the following, depending on your time tag selection
  - **Provide time tags** selected — `2n + 1` is the output width. The width consists of a count element, followed by `n` data elements, followed by `n` time tag elements.
  - **Provide time tags** cleared — `n + 1` is the output width. 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. If at least one message on the data element is valid, the count element has a nonzero value. It is 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 can 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 the resolution value. However, you must still include an associated value in the resolution vector.
`BNR`	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 `[-1,1]`. The block then multiplies that value by the scale value to recover the original value. 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.
`BCD`	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. 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.
`Discretes`	The block ignores the resolution value. However, you must still include an associated value in the resolution vector.

Sync mask

Enter a value, in hexadecimal, to specify which bits 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 series 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`, sync logic is not used. The next word begins a new message.

Sync value(s)

This parameter specifies the sync logic for the block. Enter one hex value to specify `oneSync`, two hex values separated by a space for `twoSync` logic. For example, the sync value

```
0x100
```

selects `oneSync` logic. The sync value

```
0x100 0x200
```

selects `twoSync` logic. You can enter an 32-bit value.

The sync value takes into account the value of the sync mask, as follows:

- Assume the following:
  - **Sync mask** = `0x300`
  - **Sync value(s)** = `0x100`

  When looking for the beginning of a new message, the block ANDs each input word with the **Sync mask** value `0x300`. It 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(s)** = `0x100  0x200`

  When looking for the beginning of a new message, the block ANDs each input word with the **Sync mask** value `0x300`. It compares the result to `0x100` and `0x300`. When the block finds a match, it ANDs the next input word with `0x300`. If the result 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 synchronization logic until it is time to begin the assembly of a new message.

Provide time tags

Select this check box to enable an output port of width `2n + 1`, with time tag data in the last `n` elements.

See Also

External Websites

www.aviation-ia.com
www.abaco.com

Condor 1553 Create BC Message List

Condor Create BC Message List block for bus controller

Library

Simulink Real-Time Library for MIL-STD-1553

Description

This block allocates space for a BC message list. It sets the messages in the list to no-op. Therefore, the execution of a message on a given time step requires that a 1553 Encode BC Message block execute on that time step. Because the block clears the list, you can control when a message is sent by placing 1553 Encode BC Message blocks in enabled subsystems.

Block Parameters

Number of message buffers

Enter the number of message buffers for the list. This block allocates an empty list for these buffers in Simulink Real-Time memory.

Sample time

Enter the base sample time or a multiple of the base sample time (`-1` means sample time is inherited).

See Also

See Also

`Condor 1553 Encode BC Message`

Topics

"Bus Controller Operation" on page 40-7

External Websites

www.milstd1553.com
www.abaco.com

Condor 1553 Decode BC Message

Condor 1553 Decode BC Message block for bus controller

Library

Simulink Real-Time 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 your model contains only one Decode BC Message block, connect the signal to a terminator or ground.

S — The `S` output is a vector of six elements from the received message, bit packed with information:

```
control, command1, status1, command2, status2, overall status
```

These elements derive from the `API_BC_MBUF` structure, which is contained in the Simulink Real-Time 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 vector 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 (`-1` means sample time is inherited).

See Also

External Websites

www.milstd1553.com
www.abaco.com

Condor 1553 Decode BC Status

Condor 1553 Decode BC Status block for bus controller

Library

Simulink Real-Time Library for MIL-STD-1553

Description

This block extracts individual status bits from the status output of the Decode BC Message block.

Block Parameters

Status to read

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.

Sample time

Enter the base sample time or a multiple of the base sample time (-1 means sample time is inherited).

See Also

External Websites

www.milstd1553.com
www.abaco.com

Condor 1553 Encode BC Message

Condor 1553 Encode BC Message block for bus controller

Library

Simulink Real-Time Library for MIL-STD-1553

Description

This block fills each message for the time step during which the message is sent.

The Create BC Message List block allocates an empty list of message buffers in Simulink Real-Time memory. Use the Encode BC Message block to fill these messages.

Block Parameters

Message number

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.

Remote Terminal address 1 (Receive if RT-RT)

From the list, choose a Remote Terminal address 1.

Sub address 1 (Receive if RT-RT)

From the list, choose a Remote Terminal sub address 1. Setting this parameter to `Send (0) mode command` or `Send (31) mode command` disables the **Message word count** parameter.

Remote Terminal address 2 (Transmit if RT-RT)

From the list, choose a Remote Terminal address 2. This is not used if the **Direction** is `R (BC->RT)` or `T (RT->BC)`.

Sub address 2 (Transmit if RT-RT)

From the list, choose a Remote Terminal sub address 2. This is not used if the **Direction** is `R (BC->RT)` or `T (RT->BC)`.

Mode command

If the **Sub address 1 is 0 or 31**, from the list, choose a mode command.

Message word count

Specify the number of `uint16` words to send or receive.

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 (`-1` means sample time is inherited).

See Also

See Also

`Condor 1553 Create BC Message List` | `Condor 1553 Decode BC Message`

External Websites

www.milstd1553.com
www.abaco.com

Condor 1553 Select BM Message

Condor Select 1553 BM Message block for bus monitor

Library

Simulink Real-Time Library for MIL-STD-1553

Description

This block picks a message with specified properties out of a Bus Monitor list from the PCI-1553 Bus Monitor block.

Block Input and Outputs

This block has the following inputs and outputs:

`L` — The passed through message list

`T` — Time in microseconds at which this message was received

`S` — Status (seven component vector with contents)

- `1 message number`
- `2 cmd1`
- `3 resp1`
- `4 status 1`
- `5 cmd 2`
- `6 resp 2`
- `7 status 2`

Elements `cmd1` and `cmd2` contain address information in bit fields in the 16-bit integer.

```
cmdN=<RRRRR>T<SSSSS><CCCCC>
```

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 sub address. C is the count.

Elements 5, 6, and 7 are only nonzero for the RT->RT messages.

D — 32 uint16 vector data. cmd 1 has the length of real data in the low order 5 bits.

If the block does not find a matching message in the list, the block clears the status and data words to 0.

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 sub address
- RT->RT — By both Remote Terminal and sub addresses

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 sub address 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).

Sub address 2

Specify the sub address 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 (`-1` means sample time is inherited).

See Also

See Also

`Condor PCI-1553 Bus Monitor`

External Websites

www.milstd1553.com
www.abaco.com

Condor PCI-1553

Support for MIL-STD-1553 PCI boards

Board

Condor PCI-1553

General Description

The MIL-STD-1553 driver library allows real-time applications to connect to a MIL-STD-1553 bus network to send and receive messages of up to 32 16-bit words. You must be familiar with the MIL-STD-1553 standard.

The Simulink Real-Time block library supports the MIL-STD-1553 protocol with PCI-1553 boards. These boards connect a target computer to a MIL-STD-1553 data bus via the PCI bus.

The PCI-1553 board is available with one or two channels as either a single function or a multifunction configuration. The multifunction version supports simultaneous use of the Bus Controller, Bus Monitor, and Remote Terminal functions. If more than one function is initialized, the single function version emits an error. PCI-1553 blocks support up to two channels for this board.

To support these boards, the block library provides the following driver blocks:

- Condor PCI-1553 Bus Controller Send
- Condor PCI-1553 Bus Monitor
- Condor PCI-1553 Initialize
- Condor PCI-1553 RT Initialize
- Condor PCI-1553 RT Receive
- Condor PCI-1553 RT Send

To format the Bus Controller messages, use the following utility blocks:

- 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-1553
Manufacturer	GE Fanuc (formerly Condor Engineering)
Bus type	PCI
Multiple board support	Yes

See Also

External Websites

www.milstd1553.com
www.abaco.com

Condor PCI-1553 Bus Controller Send

Condor PCI-1553 BC Send block for bus controller

Library

Simulink Real-Time 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.

For time step `N`, the Send block transmits either the response to time step `N-1` or the response to time step `N`, as specified by the **Response mode** parameter.

Block Parameters

Channel

From the list of available channels on the board, choose `1` or `2`. This channel is initialized for this command stream.

Response mode

Choose from:

- **Read response to previous message then send new message** — Before sending new messages, read the command buffers from the board.
- **Send new message then wait and read response** — Transmit new message, and then wait for response before continuing.

  Use the **Maximum wait time (microseconds)** parameter to set the amount of time the block must 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. If the **Response mode** parameter is set to **Send new message then wait and read response**, use this parameter.

A reasonable maximum wait time is 1000 microseconds (1 millisecond). This value must 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 (`-1` means sample time is inherited).

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

External Websites

www.milstd1553.com

www.abaco.com

Condor PCI-1553 Bus Monitor

Condor PCI-1553 Bus Monitor (BM) block

Library

Simulink Real-Time Library for MIL-STD-1553

Block Parameters

Channel

From the list of available channels on the board, choose `1` or `2`. This channel is initialized for this command stream.

Maximum number of messages to receive

Enter the maximum number of messages this block must read from the board.

Sample time

Enter the base sample time or a multiple of the base sample time (`-1` means sample time is inherited).

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

External Websites

www.milstd1553.com

www.abaco.com

Condor PCI-1553 Initialize

Condor PCI-1553 Initialize block

Library

Simulink Real-Time Library for MIL-STD-1553

Block Parameters

Channel

From the list of available channels on the board, choose `1` or `2`. This channel is initialized for this command stream.

Coupling mode

From the list, select from `Direct Coupling` and `Transformer Coupling`. This selection defines the mode for coupling a terminal device to the bus.

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**, **Enable retries**, **No response timeout (microseconds)**, and **Late response timeout (microseconds)**.

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 series 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. Selecting this check box enables the parameters **Monitor bus A**, **Monitor bus B**, and **Number of monitor buffers to allocate**.

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 a number at least as large as the number of messages that you expect to see between calls to the Bus Monitor block. If more messages arrive than there are buffers, some messages are 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**.

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 (`-1` means sample time is inherited).

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

External Websites

www.milstd1553.com
www.abaco.com

Condor PCI-1553 RT Initialize

Condor PCI-1553 RT Initialize block

Library

Simulink Real-Time 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 the:

- Main Condor PCI-1553 Initialize block for this board
- Another Condor PCI-1553 RT Initialize block that is connected to the main Condor PCI-1553 Initialize block

This connection carries the channel number and PCI slot information.

Block Parameters

Remote Terminal

From the list, choose a Remote Terminal from 0 to 31. The remote terminal number configured in RT Send and RT Receive must match that specified by the corresponding RT Initialize block.

Connect to bus A

Select this check box to have this Remote Terminal listen on bus A. If a message can come in on either bus, choose both A and B.

Connect to bus B

Select this check box to have this Remote Terminal listen on bus B. If a message can come in on either bus, choose both A and B.

Initial status

Enter the value to which the board should initialize the status before it executes commands. The board maintains a status word that it sends to the Bus Controller after it executes a command.

Initial BIT word

Enter the initial value of the 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 value is the initial value of the `Inhibit Terminal Flag` bit. This value can be changed from the Bus Controller with the `Inhibit Terminal` and `Override Inhibit Terminal` mode codes. The terminal bit is one of the bits in the status word that is maintained by the I/O module.

Transmit sub addresses

Enter a vector of sub addresses to which this Remote Terminal must respond. These sub addresses are the sub addresses 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 sub address for the Remote Terminal, include the sub address 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 vector is a vector of bit masks. Each mask bit corresponds to a single message length. Construct the transmit message length 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 **Legal receive message lengths**. Failure to do so prevents the sub address 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.

Bit Position	Mask Value (hex)	Word Length (words)
0	0x00000001	32

Bit Position	Mask Value (hex)	Word Length (words)
1	0x00000002	1
2	0x00000004	2
. . .		
30	0x40000000	30
31	0x80000000	31

- 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. The board sends an error to the Bus Controller by returning a status with the Message Error bit set.

Receive sub addresses

Enter a vector of the sub addresses 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 sub address for the Remote Terminal, include the sub address 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 vector is a vector of bit masks. Each bit corresponds to a single message length. Construct the receive message lengths 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 vector corresponds to the corresponding element of the **Legal transmit message lengths**. Failure to do so prevents the sub address 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.

Bit Position	Mask Value (hex)	Word Length (words)
0	0x00000001	32
1	0x00000002	1
2	0x00000004	2
. . .		
30	0x40000000	30
31	0x80000000	31

- 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. The board sends 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 (`-1` means sample time is inherited).

See Also

External Websites

www.milstd1553.com
www.abaco.com

Condor PCI-1553 RT Receive

Condor PCI-1553 RT Receive block for remote terminal

Library

Simulink Real-Time Library for MIL-STD-1553

Note

Your model can 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-sub address-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 of available channels on the board, choose `1` or `2`. This channel is initialized for this command stream.

Remote Terminal

From the list, choose a Remote Terminal from `0` to `31`. The remote terminal number configured in RT Send and RT Receive must match that specified by the corresponding RT Initialize block.

Sub address

From the list, choose a sub address from `0` 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 has the same vector width.

If this Remote Terminal or sub address can be sent messages with different lengths, specify the longest message length for this parameter. You must determine how much length is significant from content.

Enable timestamp output

Select this check box to cause output `T` to appear.

Output T returns the time, in microseconds, at which the most recent message was received. The time is measured from the beginning of execution.

Sample time

Enter the base sample time or a multiple of the base sample time (`-1` means sample time is inherited).

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

External Websites

www.milstd1553.com
www.abaco.com

Condor PCI-1553 RT Send

Condor PCI-1553 RT Send block for remote terminal

Library

Simulink Real-Time Library for MIL-STD-1553

Description

A Remote Terminal block sends data only if it receives a transmit command from the Bus Controller. The Remote Terminal block prepares the board for the next transmit command. It gets its channel, Remote Terminal number, and sub address from the Initialize and RT Initialize blocks.

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-bit or 32-bit signed or unsigned data. The block sends only the bottom 16 bits of each element.

Your model can 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-sub address-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 of available channels on the board, choose `1` or `2`. This channel is initialized for this command stream.

Remote Terminal

From the list, choose a Remote Terminal from `0` to `31`. The remote terminal number configured in RT Send and RT Receive must match that specified by the corresponding RT Initialize block.

Sub address

From the list, choose a sub address from `0` 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 (`-1` means sample time is inherited).

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

External Websites

www.milstd1553.com
www.abaco.com

Condor Q104-1553

Support for MIL-STD-1553 PC104 boards

Board

Condor Q104-1553

General Description

The MIL-STD-1553 driver library allows real-time applications to connect to a MIL-STD-1553 bus network to send and receive messages of up to 32 16-bit words. You must be familiar with the MIL-STD-1553 standard.

The Simulink Real-Time block library supports the MIL-STD-1553 protocol with Q104-1553 boards. These boards connect a target computer to a MIL-STD-1553 data bus via the PC104 bus.

The Q104-1553 board is available with one, two, three, or four 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. If more than one function is initialized, the single function version emits an error.

To support these boards, the block library provides the following driver blocks:

- 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

To format the Bus Controller messages, use the following utility blocks:

- 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

See Also

External Websites

www.milstd1553.com
www.abaco.com

Condor Q104-1553 Bus Controller Send

Condor Q104-1553 BC Send block for bus controller

Library

Simulink Real-Time 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.

For time step `N`, the Send block transmits either the response to time step `N-1` or the response to time step `N`, as specified by the **Response mode** parameter.

Block Parameters

Channel

From the list of available channels on the board, choose `1`, `2`, `3`, or `4`. This channel is initialized for this command stream.

Response mode

Choose from:

- **Read response to previous message then send new message** — Before sending new messages, read the command buffers from the board.
- **Send new message then wait and read response** — Transmit new message, and then wait for response before continuing.

  Use the **Maximum wait time (microseconds)** parameter to set the amount of time the block must 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. If the **Response mode** parameter is set to **Send new message then wait and read response**, use this parameter.

A reasonable maximum wait time is 1000 microseconds (1 millisecond). This value must 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 (`-1` means sample time is inherited).

Base address

Enter the base address of the board. This entry must correspond to the DIP switch setting on the board. For example, if the DIP switch setting is 300 (hexadecimal), enter `0x300`.

See Also

External Websites

www.milstd1553.com
www.abaco.com

Condor Q104-1553 Bus Monitor

Condor Q104-1553 Bus Monitor (BM) block

Library

Simulink Real-Time Library for MIL-STD-1553

Block Parameters

Channel

From the list of available channels on the board, choose `1`, `2`, `3`, or `4`. This channel is initialized for this command stream.

Maximum number of messages to receive

Enter the maximum number of messages this block must read from the board.

Sample time

Enter the base sample time or a multiple of the base sample time (`-1` means sample time is inherited).

Base address

Enter the base address of the board. This entry must correspond to the DIP switch setting on the board. For example, if the DIP switch setting is 300 (hexadecimal), enter `0x300`.

See Also

External Websites

www.milstd1553.com
www.abaco.com

Condor Q104-1553 Initialize

Condor Q104-1553 Initialize block

Library

Simulink Real-Time Library for MIL-STD-1553

Block Parameters

Channel

From the list of available channels on the board, choose 1, 2, 3, or 4. This channel is initialized for this command stream.

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**, **Enable retries**, **No response timeout (microseconds)**, and **Late response timeout (microseconds)**.

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 series 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. Selecting this check box enables the parameters **Monitor bus A**, **Monitor bus B**, and **Number of monitor buffers to allocate**.

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 a number at least as large as the number of messages that you expect to see between calls to the Bus Monitor block. If more messages arrive than there are buffers, some messages are 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**.

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 (`-1` means sample time is inherited).

Base address

Enter the base address of the board. This entry must correspond to the DIP switch setting on the board. For example, if the DIP switch setting is 300 (hexadecimal), enter `0x300`.

See Also

External Websites

www.milstd1553.com
www.abaco.com

Condor Q104-1553 RT Initialize

Condor Q104-1553 RT Initialize block

Library

Simulink Real-Time 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 the:

- Main Condor Q104-1553 Initialize block for this board
- Another Condor Q104-1553 RT Initialize block that is connected to the main Condor Q104-1553 Initialize block

This connection carries the channel number and PC104 base address.

Block Parameters

Remote Terminal

From the list, choose a Remote Terminal from 0 to 31. The remote terminal number configured in RT Send and RT Receive must match that specified by the corresponding RT Initialize block.

Connect to bus A

Select this check box to have this Remote Terminal listen on bus A. If a message can come in on either bus, choose both A and B.

Connect to bus B

Select this check box to have this Remote Terminal listen on bus B. If a message can come in on either bus, choose both A and B.

Initial status

Enter the value to which the board should initialize the status before it executes commands. The board maintains a status word that it sends to the Bus Controller after it executes a command.

Initial BIT word

Enter the initial value of the 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 value is the initial value of the `Inhibit Terminal Flag` bit. This value can be changed from the Bus Controller with the `Inhibit Terminal` and `Override Inhibit Terminal` mode codes. The terminal bit is one of the bits in the status word that is maintained by the I/O module.

Transmit sub addresses

Enter a vector of sub addresses to which this Remote Terminal must respond. These sub addresses are the sub addresses 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 sub address for the Remote Terminal, include the sub address 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 vector is a vector of bit masks. Each mask bit corresponds to a single message length. Construct the transmit message length 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 **Legal receive message lengths**. Failure to do so prevents the sub address 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.

Bit Position	Mask Value (hex)	Word Length (words)
0	0x00000001	32

Bit Position	Mask Value (hex)	Word Length (words)
1	0x00000002	1
2	0x00000004	2
. . .		
30	0x40000000	30
31	0x80000000	31

- 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. The board sends an error to the Bus Controller by returning a status with the Message Error bit set.

Receive sub addresses

Enter a vector of the sub addresses 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 sub address for the Remote Terminal, include the sub address 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 vector is a vector of bit masks. Each bit corresponds to a single message length. Construct the receive message lengths 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 vector corresponds to the corresponding element of the **Legal transmit message lengths**. Failure to do so prevents the sub address 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.

Bit Position	Mask Value (hex)	Word Length (words)
0	0x00000001	32
1	0x00000002	1
2	0x00000004	2
. . .		
30	0x40000000	30
31	0x80000000	31

- 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. The board sends 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 (`-1` means sample time is inherited).

See Also

External Websites

www.milstd1553.com
www.abaco.com

Condor Q104-1553 RT Receive

Condor Q104-1553 RT Receive block for remote terminal

Library

Simulink Real-Time Library for MIL-STD-1553

Note

Your model can 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-sub address-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 of available channels on the board, choose `1`, `2`, `3`, or `4`. This channel is initialized for this command stream.

Remote Terminal

From the list, choose a Remote Terminal from `0` to `31`. The remote terminal number configured in RT Send and RT Receive must match that specified by the corresponding RT Initialize block.

Sub address

From the list, choose a sub address from `0` 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 has the same vector width.

If this Remote Terminal or sub address can be sent messages with different lengths, specify the longest message length for this parameter. You must determine how much length is significant from content.

Enable timestamp output

Select this check box to cause output `T` to appear.

Output T returns the time, in microseconds, at which the most recent message was received. The time is measured from the beginning of execution.

Sample time

Enter the base sample time or a multiple of the base sample time (`-1` means sample time is inherited).

Base address

Enter the base address of the board. This entry must correspond to the DIP switch setting on the board. For example, if the DIP switch setting is 300 (hexadecimal), enter `0x300`.

See Also

External Websites

www.milstd1553.com
www.abaco.com

Condor Q104-1553 RT Send

Condor Q104-1553 RT Send block for remote terminal

Library

Simulink Real-Time Library for MIL-STD-1553

Description

A Remote Terminal block sends data only if it receives a transmit command from the Bus Controller. The Remote Terminal block prepares the board for the next transmit command. It gets its channel, Remote Terminal number, and sub address from the Initialize and RT Initialize blocks.

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-bit or 32-bit signed or unsigned data. The block sends only the bottom 16 bits of each element.

Your model can 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-sub address-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 of available channels on the board, choose 1, 2, 3, or 4. This channel is initialized for this command stream.

Remote Terminal

From the list, choose a Remote Terminal from 0 to 31. The remote terminal number configured in RT Send and RT Receive must match that specified by the corresponding RT Initialize block.

Sub address

From the list, choose a sub address from 0 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 (-1 means sample time is inherited).

Base address

Enter the base address of the board. This entry must correspond to the DIP switch setting on the board. For example, if the DIP switch setting is 300 (hexadecimal), enter 0x300.

See Also

External Websites

www.milstd1553.com
www.abaco.com

Condor QPCI-1553

Support for MIL-STD-1553 QPCI boards

Board

Condor QPCI-1553

General Description

The MIL-STD-1553 driver library allows real-time applications to connect to a MIL-STD-1553 bus network to send and receive messages of up to 32 16-bit words. You must be familiar with the MIL-STD-1553 standard.

The Simulink Real-Time block library supports the MIL-STD-1553 protocol with QPCI-1553 boards. These boards connect a target computer to a MIL-STD-1553 data bus via the high-density PCI (QPCI) bus.

The QPCI-1553 board is available with one, two, three, or four 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. If more than one function is initialized, the single function version emits an error. This board also supports a loopback mode for testing. QPCI-1553 blocks support up to four channels for this board.

To support these boards, the block library provides the following driver blocks:

- 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

To format the Bus Controller messages, use the following utility blocks:

- 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	QPCI-1553
Manufacturer	GE Fanuc (formerly Condor Engineering)
Bus type	QPCI
Multiple board support	Yes

See Also

External Websites

www.milstd1553.com
www.abaco.com

Condor QPCI-1553 Bus Controller Send

Condor QPCI-1553 BC Send block for bus controller

Library

Simulink Real-Time 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.

For time step `N`, the Send block transmits either the response to time step `N-1` or the response to time step `N`, as specified by the **Response mode** parameter.

Block Parameters

Channel

From the list of available channels on the board, choose `1`, `2`, `3`, or `4`. This channel is initialized for this command stream.

Response mode

Choose from:

- **Read response to previous message then send new message** — Before sending new messages, read the command buffers from the board.
- **Send new message then wait and read response** — Transmit new message, and then wait for response before continuing.

  Use the **Maximum wait time (microseconds)** parameter to set the amount of time the block must 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. If the **Response mode** parameter is set to **Send new message then wait and read response**, use this parameter.

A reasonable maximum wait time is 1000 microseconds (1 millisecond). This value must 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 (`-1` means sample time is inherited).

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

External Websites

www.milstd1553.com

www.abaco.com

Condor QPCI-1553 Bus Monitor

Condor QPCI-1553 Bus Monitor (BM) block

Library

Simulink Real-Time Library for MIL-STD-1553

Block Parameters

Channel

From the list of available channels on the board, choose `1`, `2`, `3`, or `4`. This channel is initialized for this command stream.

Maximum number of messages to receive

Enter the maximum number of messages this block must read from the board.

Sample time

Enter the base sample time or a multiple of the base sample time (`-1` means sample time is inherited).

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

External Websites

www.milstd1553.com

www.abaco.com

Condor QPCI-1553 Initialize

Condor QPCI-1553 Initialize block

Library

Simulink Real-Time Library for MIL-STD-1553

Block Parameters

Channel

From the list of available channels on the board, choose `1`, `2`, `3`, or `4`. This channel is initialized for this command stream.

Loopback enabled

Select this check box to route signals to the test bus on the board. This bus provides a loopback connection between the four channels on the board that does not require external wiring.

Coupling mode

From the list, select from `Direct Coupling` and `Transformer Coupling`. This selection defines the mode for coupling a terminal device to the bus.

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**, **Enable retries**, **No response timeout (microseconds)**, and **Late response timeout (microseconds)**.

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 series 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. Selecting this check box enables the parameters **Monitor bus A**, **Monitor bus B**, and **Number of monitor buffers to allocate**.

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 a number at least as large as the number of messages that you expect to see between calls to the Bus Monitor block. If more messages arrive than there are buffers, some messages are 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**.

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 (`-1` means sample time is inherited).

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

External Websites

www.milstd1553.com
www.abaco.com

Condor QPCI-1553 RT Initialize

Condor QPCI-1553 RT Initialize block

Library

Simulink Real-Time 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 the:

- 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 0 to 31. The remote terminal number configured in RT Send and RT Receive must match that specified by the corresponding RT Initialize block.

Connect to bus A

Select this check box to have this Remote Terminal listen on bus A. If a message can come in on either bus, choose both A and B.

Connect to bus B

Select this check box to have this Remote Terminal listen on bus B. If a message can come in on either bus, choose both A and B.

Initial status

Enter the value to which the board should initialize the status before it executes commands. The board maintains a status word that it sends to the Bus Controller after it executes a command.

Initial BIT word

Enter the initial value of the 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 value is the initial value of the `Inhibit Terminal Flag` bit. This value can be changed from the Bus Controller with the `Inhibit Terminal` and `Override Inhibit Terminal` mode codes. The terminal bit is one of the bits in the status word that is maintained by the I/O module.

Transmit sub addresses

Enter a vector of sub addresses to which this Remote Terminal must respond. These sub addresses are the sub addresses 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 sub address for the Remote Terminal, include the sub address 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 vector is a vector of bit masks. Each mask bit corresponds to a single message length. Construct the transmit message length 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 **Legal receive message lengths**. Failure to do so prevents the sub address 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.

Bit Position	Mask Value (hex)	Word Length (words)
0	0x00000001	32

Bit Position	Mask Value (hex)	Word Length (words)
1	0x00000002	1
2	0x00000004	2
. . .		
30	0x40000000	30
31	0x80000000	31

- 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. The board sends an error to the Bus Controller by returning a status with the Message Error bit set.

Receive sub addresses

Enter a vector of the sub addresses 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 sub address for the Remote Terminal, include the sub address 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 vector is a vector of bit masks. Each bit corresponds to a single message length. Construct the receive message lengths 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 vector corresponds to the corresponding element of the **Legal transmit message lengths**. Failure to do so prevents the sub address 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.

Bit Position	Mask Value (hex)	Word Length (words)
0	0x00000001	32
1	0x00000002	1
2	0x00000004	2
. . .		
30	0x40000000	30
31	0x80000000	31

- 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. The board sends 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 (`-1` means sample time is inherited).

See Also

External Websites

www.milstd1553.com
www.abaco.com

Condor QPCI-1553 RT Receive

Condor QPCI-1553 RT Receive block for remote terminal

Library

Simulink Real-Time Library for MIL-STD-1553

Note

Your model can 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-sub address-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 of available channels on the board, choose `1`, `2`, `3`, or `4`. This channel is initialized for this command stream.

Remote Terminal

From the list, choose a Remote Terminal from `0` to `31`. The remote terminal number configured in RT Send and RT Receive must match that specified by the corresponding RT Initialize block.

Sub address

From the list, choose a sub address from `0` 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 has the same vector width.

If this Remote Terminal or sub address can be sent messages with different lengths, specify the longest message length for this parameter. You must determine how much length is significant from content.

Enable timestamp output

Select this check box to cause output `T` to appear.

Output T returns the time, in microseconds, at which the most recent message was received. The time is measured from the beginning of execution.

Sample time

Enter the base sample time or a multiple of the base sample time (`-1` means sample time is inherited).

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

External Websites

www.milstd1553.com
www.abaco.com

Condor QPCI-1553 RT Send

Condor QPCI-1553 RT Send block for remote terminal

Library

Simulink Real-Time Library for MIL-STD-1553

Description

A Remote Terminal sends data only if it receives a transmit command from the Bus Controller. The Remote Terminal block prepares the board for the next transmit command. It gets its channel, Remote Terminal number, and sub address from the Initialize and RT Initialize blocks.

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-bit or 32-bit signed or unsigned data. The block sends only the bottom 16 bits of each element.

Your model can 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-sub address-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 of available channels on the board, choose 1, 2, 3, or 4. This channel is initialized for this command stream.

Remote Terminal

From the list, choose a Remote Terminal from 0 to 31. The remote terminal number configured in RT Send and RT Receive must match that specified by the corresponding RT Initialize block.

Sub address

From the list, choose a sub address from 0 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 (`-1` means sample time is inherited).

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

External Websites

www.milstd1553.com
www.abaco.com

Condor QPCX-1553

Support for MIL-STD-1553 QPCX boards

Board

Condor QPCX-1553

General Description

The MIL-STD-1553 driver library allows real-time applications to connect to a MIL-STD-1553 bus network to send and receive messages of up to 32 16-bit words. You must be familiar with the MIL-STD-1553 standard.

The Simulink Real-Time block library supports the MIL-STD-1553 protocol with QPCX-1553 boards. These boards connect a target computer to a MIL-STD-1553 data bus via the high-density PCX (QPCX) bus.

The QPCX-1553 board is available with one, two, three, or four 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. If more than one function is initialized, the single function version emits an error. This board also supports a loopback mode for testing. QPCX-1553 blocks support up to four channels for this board.

To support these boards, the block library provides the following driver blocks:

- Condor QPCX-1553 Bus Controller Send
- Condor QPCX-1553 Bus Monitor
- Condor QPCX-1553 Initialize
- Condor QPCX-1553 RT Initialize
- Condor QPCX-1553 RT Receive
- Condor QPCX-1553 RT Send

To format the Bus Controller messages, use the following utility blocks:

- 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	QPCX-1553
Manufacturer	GE Fanuc (formerly Condor Engineering)
Bus type	QPCX
Multiple board support	Yes

See Also

External Websites

www.milstd1553.com
www.abaco.com

Condor QPCX-1553 Bus Controller Send

Condor QPCX-1553 BC Send block for bus controller

Library

Simulink Real-Time 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.

For time step `N`, the Send block transmits either the response to time step `N-1` or the response to time step `N`, as specified by the **Response mode** parameter.

Block Parameters

Channel

From the list of available channels on the board, choose `1`, `2`, `3`, or `4`. This channel is initialized for this command stream.

Response mode

Choose from:

- **Read response to previous message then send new message** — Before sending new messages, read the command buffers from the board.
- **Send new message then wait and read response** — Transmit new message, and then wait for response before continuing.

  Use the **Maximum wait time (microseconds)** parameter to set the amount of time the block must 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. If the **Response mode** parameter is set to **Send new message then wait and read response**, use this parameter.

A reasonable maximum wait time is 1000 microseconds (1 millisecond). This value must 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 (`-1` means sample time is inherited).

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

External Websites

www.milstd1553.com

www.abaco.com

Condor QPCX-1553 Bus Monitor

Condor QPCX-1553 Bus Monitor (BM) block

Library

Simulink Real-Time Library for MIL-STD-1553

Block Parameters

Channel

From the list of available channels on the board, choose `1`, `2`, `3`, or `4`. This channel is initialized for this command stream.

Maximum number of messages to receive

Enter the maximum number of messages this block must read from the board.

Sample time

Enter the base sample time or a multiple of the base sample time (`-1` means sample time is inherited).

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

External Websites

www.milstd1553.com

www.abaco.com

Condor QPCX-1553 Initialize

Condor QPCX-1553 Initialize block

Library

Simulink Real-Time Library for MIL-STD-1553

Block Parameters

Channel

From the list of available channels on the board, choose `1`, `2`, `3`, or `4`. This channel is initialized for this command stream.

Loopback enabled

Select this check box to route signals to the test bus on the board. This bus provides a loopback connection between the four channels on the board that does not require external wiring.

Coupling mode

From the list, select from `Direct Coupling` and `Transformer Coupling`. This selection defines the mode for coupling a terminal device to the bus.

Output voltage

Enter the required output voltage, in volts. The output voltage range depends on the coupling mode.

- `Direct Coupling` — `0–6.5` V
- `Transformer Coupling` — `0–19.8` V

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**, **Enable retries**, **No response timeout (microseconds)**, and **Late response timeout (microseconds)**.

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 series 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. Selecting this check box enables the parameters **Monitor bus A**, **Monitor bus B**, and **Number of monitor buffers to allocate**.

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 a number at least as large as the number of messages that you expect to see between calls to the Bus Monitor block. If more messages arrive than there are buffers, some messages are 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**.

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 (`-1` means sample time is inherited).

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

External Websites

www.milstd1553.com
www.abaco.com

Condor QPCX-1553 RT Initialize

Condor QPCX-1553 RT Initialize block

Library

Simulink Real-Time 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 the:

- Main Condor QPCX-1553 Initialize block for this board
- Another Condor QPCX-1553 RT Initialize block that is connected to the main Condor QPCX-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 0 to 31. The remote terminal number configured in RT Send and RT Receive must match that specified by the corresponding RT Initialize block.

Connect to bus A

Select this check box to have this Remote Terminal listen on bus A. If a message can come in on either bus, choose both A and B.

Connect to bus B

Select this check box to have this Remote Terminal listen on bus B. If a message can come in on either bus, choose both A and B.

Initial status

Enter the value to which the board should initialize the status before it executes commands. The board maintains a status word that it sends to the Bus Controller after it executes a command.

Initial BIT word

Enter the initial value of the 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 value is the initial value of the `Inhibit Terminal Flag` bit. This value can be changed from the Bus Controller with the `Inhibit Terminal` and `Override Inhibit Terminal` mode codes. The terminal bit is one of the bits in the status word that is maintained by the I/O module.

Transmit sub addresses

Enter a vector of sub addresses to which this Remote Terminal must respond. These sub addresses are the sub addresses 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 sub address for the Remote Terminal, include the sub address 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 vector is a vector of bit masks. Each mask bit corresponds to a single message length. Construct the transmit message length 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 **Legal receive message lengths**. Failure to do so prevents the sub address 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.

Bit Position	Mask Value (hex)	Word Length (words)
0	0x00000001	32

Bit Position	Mask Value (hex)	Word Length (words)
1	0x00000002	1
2	0x00000004	2
. . .		
30	0x40000000	30
31	0x80000000	31

- 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. The board sends an error to the Bus Controller by returning a status with the Message Error bit set.

Receive sub addresses

Enter a vector of the sub addresses 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 sub address for the Remote Terminal, include the sub address 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 vector is a vector of bit masks. Each bit corresponds to a single message length. Construct the receive message lengths 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 vector corresponds to the corresponding element of the **Legal transmit message lengths**. Failure to do so prevents the sub address 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.

Bit Position	Mask Value (hex)	Word Length (words)
0	0x00000001	32
1	0x00000002	1
2	0x00000004	2
. . .		
30	0x40000000	30
31	0x80000000	31

- 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. The board sends 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 (`-1` means sample time is inherited).

See Also

External Websites

www.milstd1553.com
www.abaco.com

Condor QPCX-1553 RT Receive

Condor QPCX-1553 RT Receive block for remote terminal

Library

Simulink Real-Time Library for MIL-STD-1553

Note

Your model can 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-sub address-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 of available channels on the board, choose `1`, `2`, `3`, or `4`. This channel is initialized for this command stream.

Remote Terminal

From the list, choose a Remote Terminal from `0` to `31`. The remote terminal number configured in RT Send and RT Receive must match that specified by the corresponding RT Initialize block.

Sub address

From the list, choose a sub address from `0` 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 has the same vector width.

If this Remote Terminal or sub address can be sent messages with different lengths, specify the longest message length for this parameter. You must determine how much length is significant from content.

Enable timestamp output

Select this check box to cause output `T` to appear.

Output T returns the time, in microseconds, at which the most recent message was received. The time is measured from the beginning of execution.

Sample time

Enter the base sample time or a multiple of the base sample time (`-1` means sample time is inherited).

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

External Websites

www.milstd1553.com
www.abaco.com

Condor QPCX-1553 RT Send

Condor QPCX-1553 RT Send block for remote terminal

Library

Simulink Real-Time Library for MIL-STD-1553

Description

A Remote Terminal sends data only if it receives a transmit command from the Bus Controller. The Remote Terminal block prepares the board for the next transmit command. It gets its channel, Remote Terminal number, and sub address from the Initialize and RT Initialize blocks.

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-bit or 32-bit signed or unsigned data. The block sends only the bottom 16 bits of each element.

Your model can 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-sub address-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 of available channels on the board, choose 1, 2, 3, or 4. This channel is initialized for this command stream.

Remote Terminal

From the list, choose a Remote Terminal from 0 to 31. The remote terminal number configured in RT Send and RT Receive must match that specified by the corresponding RT Initialize block.

Sub address

From the list, choose a sub address from 0 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 (`-1` means sample time is inherited).

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

External Websites

www.milstd1553.com
www.abaco.com

GE Fanuc PCI-5565

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 Simulink Real-Time software uses these boards as part of the shared memory network that you can use to exchange data between computer nodes.

The Simulink Real-Time 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 endianess block, see `Byte Reversal/Change Endianess`.

Board Characteristics

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

See Also

External Websites

www.abaco.com

Introduced in R2009a

GE Fanuc 5565 broadcast

GE Fanuc PCI-5565 broadcast interrupt block

Library

Simulink Real-Time Library for GE Fanuc (VMIC)

Note

The 5565 broadcast block generates a network interrupt that other boards in the shared memory network can detect.

The 5565 broadcast block has two inputs:

- En — This input port is a Boolean input that can enable or disable the transmission of a network interrupt. For continuous interrupts. connect this input port to a Constant block with a Boolean value of true (1). If you want to enable interrupts only part of the time, you can use another kind of input, such as a pulse generator.
- Data — This input port allows you to transmit a data value (`uint32`) with the interrupt.

Note: The Simulink Real-Time 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 the nodes in the shared memory network. You cannot broadcast the interrupt to another 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 25-9).

```
node.Interface.Interrupts.PendingIntX
```

Sampletime

Enter the base sample time or a multiple of the base sample time (`-1` means sample time is inherited)

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

External Websites

www.abaco.com

Introduced in R2009a

GE Fanuc 5565 init

GE Fanuc PCI-5565 initialization block

Library

Simulink Real-Time Library for GE Fanuc (VMIC)

Note

Before you begin to configure these block parameters, be sure that you have a predefined node initialization structure. See “Initialize GE Fanuc Shared Nodes” on page 25-4.

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 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

External Websites

www.abaco.com

Introduced in R2009a

GE Fanuc 5565 read

GE Fanuc PCI-5565 read block

Library

Simulink Real-Time Library for GE Fanuc (VMIC)

Note

Before you begin to configure this block, be sure that you have a predefined shared memory partition structure. The 5565 read block requires this structure to specify how Simulink signal values are mapped in the shared memory. It also uses this structure to determine the total size and address of the shared memory. See “Create GE Fanuc Shared Partitions” on page 25-2.

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 (`-1` means sample time is inherited)

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

Error Status Port

Select this check box to monitor the status of the PCI-5565 board LISR register modes.

See Also

External Websites

www.abaco.com

Introduced in R2009a

GE Fanuc 5565 write

GE Fanuc PCI-5565 write block

Library

Simulink Real-Time Library for GE Fanuc (VMIC)

Note

Before you begin to configure this block, be sure that you have a predefined shared memory partition structure. The 5565 write block requires this structure to specify how Simulink signal values are mapped in the shared memory. It also uses this structure to determine the total size and address of the shared memory. See “Create GE Fanuc Shared Partitions” on page 25-2.

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 (`-1` means sample time is inherited)

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

Error Status Port

Select this check box to monitor the status of the PCI-5565 board LISR register modes.

See Also

External Websites

www.abaco.com

General Standards, HUMUSOFT

General Standards

This topic describes General Standards I/O boards supported by the Simulink Real-Time product (`www.generalstandards.com`).

PMC-ADADIO Basics

The PMC-ADADIO board is an analog I/O PCI mezzanine card (PMC) device that can be used, for example, for data acquisition and process monitoring.

The Simulink Real-Time block library supports this board with A/D and D/A driver blocks. The following Simulink Real-Time 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 Simulink Real-Time 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 differs slightly from other boards. For example, most of the A/D and D/A blocks have Boolean enable ports (labeled `E`) 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 consistency in value 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. 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. During this time, your model can perform other operations. For example, you can insert a typical D/A block configuration between the AD Start and AD Read blocks. A typical configuration for analog output operation connects the DA Write and DA Update blocks.

Adding A/D Blocks for Analog Input

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.

1. In the MATLAB Command Window, type

   ```
   slrtlib
   ```

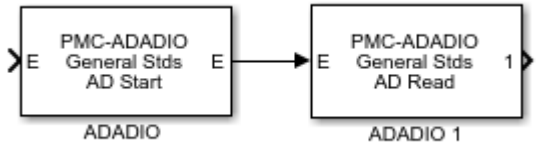
   The Simulink Real-Time driver block library opens.
2. Double-click the A/D group block.

   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 an AD Start 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.
6. Drag 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:

- A signal has not been connected to the AD Start block input enable port, `E`, so the port has a default value of `true'. Therefore, the output enable port of the AD Start block and input enable port of the AD Read block also have a value of `true'.
- Drag a Ground block from the Simulink Source library to your model. To prevent build errors until you add another block, connect that block to the unconnected input port of AD Start.
- Drag a Terminator block from the Simulink Sink library to your model. To prevent build errors until you add another block, connect that block to the unconnected output ports of AD Read.
- 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.

  The Start block must execute before every call to the Read block. If the Read block is executed without the Start block, the system stops responding because it is waiting for data to be available.

8 From the File menu, select **Save As**. Browse to a writable folder 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 for Input Enable” on page 35-6.

Adding Enable Signal Blocks for Input Enable

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.

This procedure assumes that you have a model named AdadioADDA (see "Adding A/D Blocks for Analog Input" on page 35-4). 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:

   ```
   slrtlib
   ```

   The Simulink Real-Time 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 a Create Enable Signal block from the General Standards window to the new model.
6. Double-click the Create Enable Signal block and clear the Show input port for thresholding signal of type double.

   The Create Enable Signal block has an output enable port, `E`, that can provide an input enable signal for the other PMC-ADADIO blocks. The Boolean value of this output enable port comes from an input port `S` and an optional 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 a Sine Wave block to the new model.

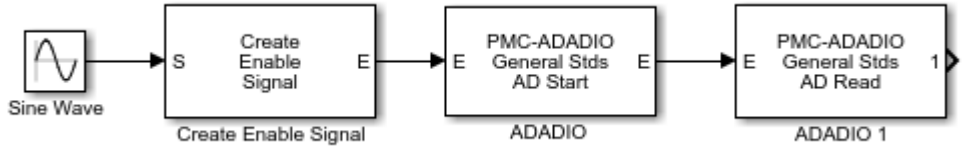
10 Connect the output port of the Sine Wave block to the input port S of the Create Enable Signal block.

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.

Adding D/A Blocks for Analog Output

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.

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:

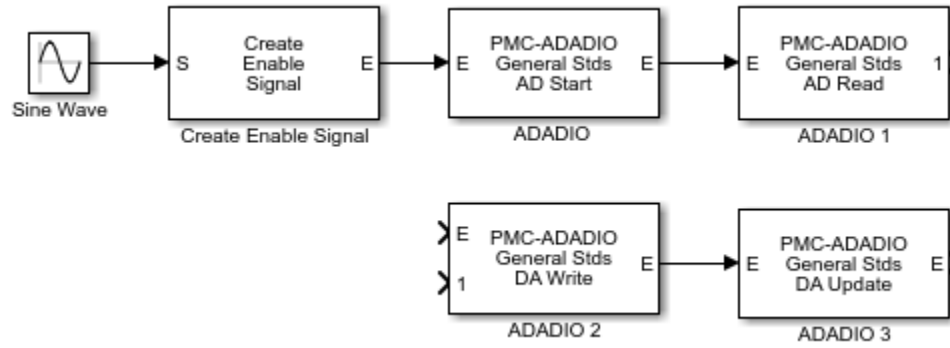
   ```
   slrtlib
   ```

   The Simulink Real-Time 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 a DA Write block from the General Standards window to the new model.
6. Drag a DA Update block from the General Standards window to the new model.
7. Connect the DA Write block to the DA Update block.

The model looks like this figure.



Note the following:

- A signal has not been connected to the DA Write block input enable port, E, so the port has a default value of 'true'. Therefore, 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 AD Read blocks, the DA Write and DA Update blocks do not need to be directly connected to each other. You can also interleave other blocks.

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 or blocks must have an input enable port and an output enable port. See “Interleaving Analog Input and Output Blocks” on page 35-10.

Interleaving Analog Input and Output Blocks

After the AD Start block executes, the acquisition I/O module becomes busy with the operation. If you configure the AD Read block to execute immediately, it sits idle waiting for the module to finish the acquisition. Rather than allowing idle cycles, you can insert other blocks between the AD Start and AD Read block

- 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:

  a AD Start enable out to DA Write enable in

  b DA Write enable out to DA Update enable in

  c DA Update enable out to AD Read enable in

  This execution order results in the time sequence AD Start ⇒ DA Write⇒ DA Update ⇒ AD Read. By the time AD Read executes, the I/O module has either finished or is much closer to finishing. If you use interleaving, less time is wasted. The data input to DA Write comes from another part of the model. Typically, it is 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 AD Read blocks, an Enable Create Signal block, a DA Write, and a DA Update block:

1. If the model `AdadioADDA` is not already open, in the MATLAB Command Window, type:

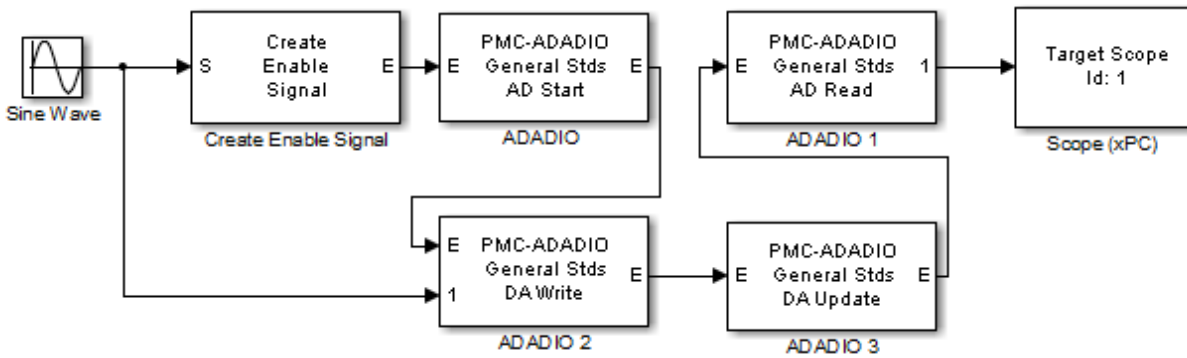
   ```
   AdadioADDA
   ```

   The model opens.
2. In the MATLAB window, type

   ```
   slrtlib
   ```

   The Simulink Real-Time library opens.

3. To insert the intermediate DA blocks between the AD Start and AD Read blocks, first disconnect the AD Start block from the AD Read block.
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 Simulink Real-Time library, double-click the `Displays and Logging` group block
9. Drag the real-time Scope block to the `AdadioADDA` model.
10. Connect the output E port of the AD Read block to the real-time Scope block.



11. From the File menu, select **Save**.

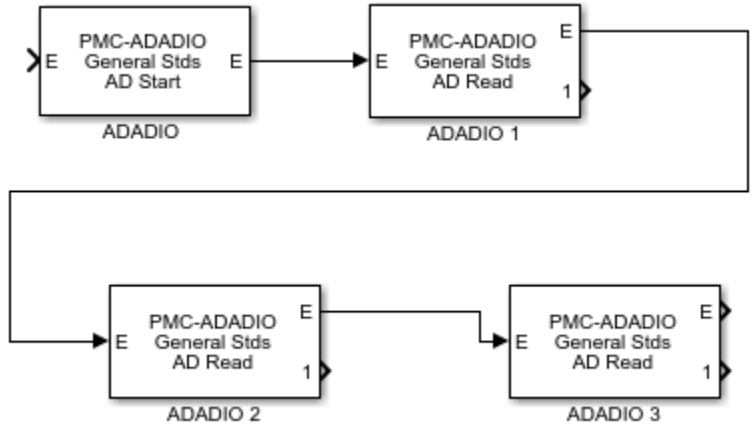
You can build and run the model and download it to your target computer like other Simulink Real-Time models.

Using Multiple PMC-ADADIO Boards

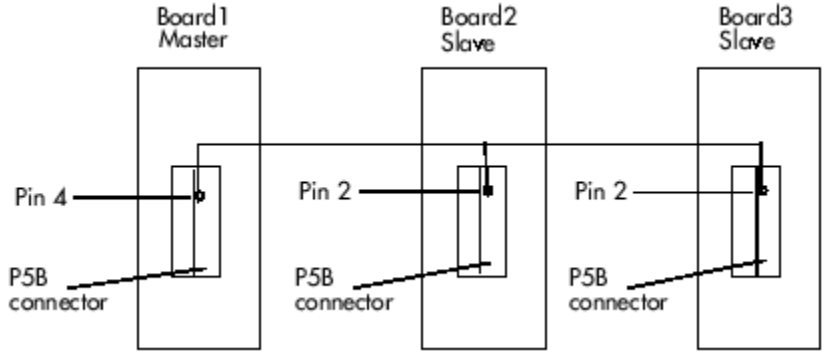
The previous topics describe how to set up a model for up to eight channels performing simultaneous analog to digital conversion. You can increase the number of channels for this conversion by using two or more PMC-ADADIO boards, configured in a master/slave configuration.

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 slave boards, the model must contain an associated A/D Read block.
- Connect the enable output port (E) of the master board A/D Start block to the enable input port (E) of each slave A/D Read blocks. You can connect these ports in one of the following ways. 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.



- Connect pin 4 of the P5B connector of the master board to pin 2 of the P5B connector of each slave board. These connections tie the INPUT TRIGGER READY signal of the master board to the INPUT TRIGGER signal of each slave board.



Note: The pulse of the master board INPUT TRIGGER READY output occurs 250 ns after the software trigger in the A/D Start block. Because the slave boards wait for this pulse, they trigger 250 ns after the master board. For most situations, this delay is negligible.

Overview of Audio Systems

The Simulink Real-Time product supports audio systems 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 Simulink Real-Time blocks support these blocks:

`General Standards 24DSI12 Analog Input`

Your audio system 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 manufacturer 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. 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 manufacturer 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.

General Standards Blocks

This topic describes General Standards I/O blocks supported by the Simulink Real-Time product (`www.generalstandards.com`).

General Standards PMC-ADADIO

Support for PMC-ADADIO multifunction ADC/DAC board

Board

General Standards PMC-ADADIO

General Description

The PMC-ADADIO is a high-speed 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. To determine the voltage range of your board, consult the manufacturer documentation.

The Simulink Real-Time 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

You cannot use the PMC-ADADIO Digital Input and Digital Output driver blocks simultaneously on the same board. Trying to do so results in an error.

Board Characteristics

Board name	PMC-ADADIO

Manufacturer	General Standards
Bus type	PCI
Access method	Memory mapped or I/O mapped
Multiple block instance support	No
Multiple board support	Yes

See Also

External Websites

www.generalstandards.com

General Standards Create Enable Signal

Create Enable Signal block

Library

Simulink Real-Time Library for General Standards

Description

Use the Create Enable Signal block to determine when and in what order a sequence of blocks executes. 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.

Scaling Input to Output

The values passed to the enable output port of this block are Boolean values.

Block Parameters

Show input port for thresholding signal of type double

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.

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.

Enable threshold

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.

See Also

External Websites

www.generalstandards.com

General Standards PMC-ADADIO Analog Input (A/D) Start

PMC-ADADIO Analog Input Start block

Library

Simulink Real-Time Library for General Standards

Scaling Input to Output

The values passed to the enable input port and enable output port of this block are Boolean values.

Block Parameters

Enable input port

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.

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 (`-1` means sample time is inherited).

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

External Websites

www.generalstandards.com

General Standards PMC-ADADIO Analog Input (A/D) Read

PMC-ADADIO Analog Input Read block

Library

Simulink Real-Time Library for General Standards

Scaling Input to Output

I/O Module Input	Block Output Data Type	Scaling
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 acquires data from the channels in numerical order. It is not possible to access channels in a different order or with gaps in the channel sequence. For example, if you enter 3, data is acquired from channels 1, 2, and 3. Select a number between 1 and 8 even if this manufacturer 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 manufacturer documentation to determine the voltage range of your

board. The Simulink Real-Time 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 manufacturer documentation 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 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 manufacturer 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 I/O module . 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 (`-1` means sample time is inherited).

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

External Websites

www.generalstandards.com

General Standards PMC-ADADIO Analog Output Write Block

PMC-ADADIO Analog Output Write block

Library

Simulink Real-Time Library for General Standards

Note

Data from 1 to 4 input data streams is written to the 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 set to true. The output registers are held and not written through to the output converters until the analog output update block is executed.

Scaling Input to Output

I/O Module Output	Block Input Data Type	Scaling
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 if this 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 replicated over the channel vector. If you specify a value of 1, the corresponding channel is reset to the value specified in the initial value vector. If you specify a value of 0, the channel remains at the last value attained while the model was running. For example, if **Channel vector** is `[ 2 3 ]` and the **Reset vector** is `[1]`, the action taken will be the same as if **Reset vector** was set to `[1 1]`. Both channels will be reset to their initial values when model execution is stopped.

Initial value vector

The initial value vector contains the initial voltage values for the output channels. Enter a scalar or a vector that is the same length as the channel vector. If you specify a scalar value, that value is replicated as initial value over the channel vector. 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 manufacturer documentation to determine the voltage range of your board. The Simulink Real-Time 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.

Note: The calibration cycle takes several seconds to run. Refer to the manufacturer 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 I/O module. If the enable input signal is true, then the block input data is written to the D/A ports. If the enable input signal is false, then data is not written to the D/A ports 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 (`-1` means sample time is inherited).

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

External Websites

www.generalstandards.com

General Standards PMC-ADADIO Analog Output (D/A) Update

PMC-ADADIO Analog Output Update block

Library

Simulink Real-Time Library for General Standards

Scaling Input to Output

The values passed to the enable input port and enable output port of this block are Boolean values.

Block Parameters

Enable input port

Select this check box to display an input enable port that controls writing data to the D/A ports on the I/O module. If the enable input signal is true, then the block input data is written to the D/A ports. If the enable input signal is false, then data is not written to the D/A ports 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 (`-1` means sample time is inherited).

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

External Websites

www.generalstandards.com

General Standards PMC-ADADIO Digital Input

PMC-ADADIO Digital Input block

Library

Simulink Real-Time Library for General Standards

Note

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

I/O Module Input	Block Output Data Type	Scaling
TTL	Double	TTL low = 0.0 TTL high = 1.0

The values passed to the enable input port and enable output port of this block are Boolean values.

Block Parameters

Channel vector

Enter numbers between 1 and 8 to select the digital input lines to be read. This driver allows the selection of individual digital input lines in arbitrary 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 if this 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 I/O module. 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 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 (`-1` means sample time is inherited).

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

External Websites

www.generalstandards.com

General Standards PMC-ADADIO Digital Output

PMC-ADADIO Digital Output block

Library

Simulink Real-Time Library for General Standards

Note

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

I/O Module Output	Block Input Data Type	Scaling
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 arbitrary 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 if this 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 replicated over the channel vector. If you specify a value of 1, the corresponding channel is reset to the value specified in 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 replicated as initial value over the channel vector. 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 I/O module. If the enable input signal is true, then the block input data is written to the digital output ports. If the enable input signal is false, then data is not written to the digital output ports 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 (`-1` means sample time is inherited).

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

External Websites

www.generalstandards.com

General Standards PMC-16AO-12

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 Simulink Real-Time block.

The Simulink Real-Time block library supports this board with these driver blocks:

- General Standards PMC-16AO-12 Analog Output

Board Characteristics

Board name	PMC-16AO12
Manufacturer	General Standards
Bus type	PCI
Access method	Memory mapped
Multiple block instance support	No
Multiple board support	Yes

See Also

External Websites

www.generalstandards.com

General Standards PMC-16AO-12 Analog Output

PMC-16AO-12 Analog Output block

Library

Simulink Real-Time Library for General Standards

Scaling Input to Output

The analog output range of this board depends on the output range that was chosen at the time of purchase. This range can be `+-2.5`, `+-5.0`, or `+-10.0`.

I/O Module Output	Block Input Data Type	Scaling
Volts	Double	1

Block Parameters

Channel vector

Enter numbers between 1 and 12. This is a vector parameter that specifies the output channels. For example, to produce output on channels 2 and 3, enter

```
[2 3]
```

The 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 replicated over the channel vector. If you specify a value of 1, the corresponding channel is reset to the value specified in the initial value vector. If you specify a value of 0, the channel remains at the last value attained while the model was running. For example, if **Channel vector** is `[ 2 3 ]` and the

Reset vector is `[1]`, the action taken will be the same as if **Reset vector** was set to `[1 1]`. Both channels will be reset to their initial values when model execution is stopped.

Initial value vector

The initial value vector contains the initial voltage values for the output channels. Enter a scalar or a vector that is the same length as the channel vector. If you specify a scalar value, that value is replicated as initial value over the channel vector. 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 the 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.

Note: The calibration cycle takes several seconds to run. Refer to the manufacturer documentation for further information on autocalibration.

Ground sense

Choose this check box to enable onboard compensation for ground potential differences. Refer to the manufacturer documentation for further information on ground sense.

Sampletime

Enter the base sample time or a multiple of the base sample time (`-1` means sample time is inherited)

PCI slot

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

External Websites

www.generalstandards.com

General Standards PMC66-16AO16

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:

- 8, 12, or 16 output channels
- Optional 10 kHz or 100 kHz analog filters on the output ports

> **Note:** Because the analog output of this board contains a relatively high frequency component, purchase the PMC66-16AO16 board with one of the filter options. If your board does not have a filter, its output has fast rise times 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 Simulink Real-Time block library supports this board with this driver block:

- General Standards 16AO16 Analog Output

Use the General Standards 16AO16 Analog Output block with the `General Standards 24DSI12 Analog Input` for your Simulink Real-Time audio system model.

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

See Also

External Websites

www.generalstandards.com

General Standards 16AO16 Analog Output

General Standards 16AO16 Analog Output

Library

Simulink Real-Time Library for General Standards

Scaling Input to Output

I/O Module Output	Block Input Data Type	Scaling
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.

Input Range (V)	Range Code
-10 to +10	+-10
-5 to +5	+-5
-2.5 to +2.5	+-2.5
-1.5 to +1.5	+-1.5

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 manufacturer documentation for more information about this function.

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. 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 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 manufacturer 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 manufacturer 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 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

External Websites

www.generalstandards.com

General Standards PMC-24DSI12

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 4, 8, or 12 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 system requires more channels:

  1. Designate one board as the master board.
  2. Add slave boards PMC-24DSI12 to the target computer.
  3. Connect 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:

  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:

   ```
   tg = slrt;
   getPCIInfo(tg, 'all')
   .
   ```

   This function returns a list of the installed PCI devices on your target computer.
2. Find the name `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 **Simulink Real-Time Options** node, from the **Real-time interrupt source** list, select the same value as the IRQ value that `SimulinkRealTime.target.getPCIInfo` returns (see step 2).
7. To specify that the General Standards 24DSI12 board generates the interrupt, from the **I/O board generating the interrupt** list, select `General Standards 24DSI12`.
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 Simulink Real-Time block library supports this board with this driver block:

- General Standards 16AO16 Analog Output

For frame audio output, use the General Standards PMC-24DSI12 Analog Input block with the `General Standards 16AO16 Analog Output` block for your Simulink Real-Time audio system model.

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.

See Also

External Websites

www.generalstandards.com

General Standards 24DSI12 Analog Input

General Standards 24DSI12 Analog Input

Library

Simulink Real-Time Library for General Standards

Scaling Output to Input

I/O Module Input	Block Output Data Type	Scaling
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* × *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 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 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 [-2^23 2^23-1]`

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`

This parameter takes effect only if the **Sample rate** parameter exceeds 100 kHz. See the manufacturer 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 manufacturer 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.

`TTL`

See the manufacturer 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 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

External Websites

www.generalstandards.com

HUMUSOFT Blocks

Humusoft AD622

HUMUSOFT AD622 I/O board

Board

HUMUSOFT AD622

General Description

The AD622 provides:

- 8 single analog input (A/D) channels (14-bit)
- 8 analog output (D/A) channels (14-bit)
- 8 digital inputs
- 8 digital outputs

The Simulink Real-Time 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

See Also

External Websites

www.humusoft.cz

Humusoft AD622 Analog Input (A/D)

HUMUSOFT AD622 Analog Input block

Library

Simulink Real-Time Library for HUMUSOFT

Scaling Input to Output

Hardware Input	Block Output Data Type	Scaling
Volts	Double	1

Block Parameters

Channel Vector

Enter numbers between 1 and 8. This driver allows the selection of individual channels in arbitrary order. The number of elements defines the number of channels used.

For example, to use the first, second, and fifth channels, enter

```
[1,2,5]
```

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

Sample time

Enter the base sample time or a multiple of the base sample time (`-1` means sample time is inherited).

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

External Websites

www.humusoft.cz

Humusoft AD622 Analog Output (D/A)

HUMUSOFT AD622 Analog Output block

Library

Simulink Real-Time Library for HUMUSOFT

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 channels in arbitrary order. The number of elements defines the number of channels used.

For example, to use the first, second, and fifth channels, enter

```
[1,2,5]
```

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

Sample time

Enter the base sample time or a multiple of the base sample time (`-1` means sample time is inherited).

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

External Websites

www.humusoft.cz

Humusoft AD622 Digital Input

HUMUSOFT AD622 Digital Input block

Library

Simulink Real-Time Library for HUMUSOFT

Scaling Input to Output

Hardware Input	Block Output Data Type	Scaling
TTL	Double	< 0.8 = TTL low ≥ 0.8 = TTL high

Channel Vector

Enter numbers between 1 and 8. This driver allows the selection of individual channels in arbitrary order. The number of elements defines the number of channels used.

For example, to use the first, second, and fifth channels, enter

```
[1,2,5]
```

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

Sample time

Enter the base sample time or a multiple of the base sample time (`-1` means sample time is inherited).

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

External Websites

www.humusoft.cz

Humusoft AD622 Digital Output

HUMUSOFT AD622 Digital Output block

Library

Simulink Real-Time 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 channels in arbitrary order. The number of elements defines the number of channels used.

For example, to use the first, second, and fifth channels, enter

```
[1,2,5]
```

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

Sample time

Enter the base sample time or a multiple of the base sample time (`-1` means sample time is inherited).

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

External Websites

www.humusoft.cz

Humusoft MF624

HUMUSOFT MF624 I/O board

Board

HUMUSOFT MF624

General Description

The MF624 provides:

- 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
- 4 quadrature encoder inputs with single-ended or differential input

The Simulink Real-Time 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

See Also

External Websites

www.humusoft.cz

Humusoft MF624 Analog Input (A/D)

HUMUSOFT MF624 Analog Input block

Library

Simulink Real-Time Library for HUMUSOFT

Scaling Input to Output

Hardware Input	Block Output Data Type	Scaling
Volts	Double	1

Block Parameters

Channel Vector

Enter numbers between 1 and 8. This driver allows the selection of individual channels in arbitrary order. The number of elements defines the number of channels used.

For example, to use the first, second, and fifth channels, enter

```
[1,2,5]
```

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

Sample time

Enter the base sample time or a multiple of the base sample time (`-1` means sample time is inherited).

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

External Websites

www.humusoft.cz

Humusoft MF624 Analog Output (D/A)

HUMUSOFT MF624 Analog Output block

Library

Simulink Real-Time Library for HUMUSOFT

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 channels in arbitrary order. The number of elements defines the number of channels used.

For example, to use the first, second, and fifth channels, enter

```
[1,2,5]
```

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

Sample time

Enter the base sample time or a multiple of the base sample time (`-1` means sample time is inherited).

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

External Websites

www.humusoft.cz

Humusoft MF624 Digital Input

HUMUSOFT MF624 Digital Input block

Library

Simulink Real-Time Library for HUMUSOFT

Scaling Input to Output

Hardware Input	Block Output Data Type	Scaling
TTL	Double	< 0.8 = TTL low ≥ 0.8 = TTL high

Channel Vector

Enter numbers between 1 and 8. This driver allows the selection of individual channels in arbitrary order. The number of elements defines the number of channels used.

For example, to use the first, second, and fifth channels, enter

```
[1,2,5]
```

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

Sample time

Enter the base sample time or a multiple of the base sample time (`-1` means sample time is inherited).

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

External Websites

www.humusoft.cz

Humusoft MF624 Digital Output

HUMUSOFT MF624 Digital Output block

Library

Simulink Real-Time 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 channels in arbitrary order. The number of elements defines the number of channels used.

For example, to use the first, second, and fifth channels, enter

```
[1,2,5]
```

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

Sample time

Enter the base sample time or a multiple of the base sample time (`-1` means sample time is inherited).

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

External Websites

www.humusoft.cz

Humusoft MF624 Counter Input

HUMUSOFT MF624 Counter Input block

Library

Simulink Real-Time 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 4. This driver allows the selection of individual channels in arbitrary order. The number of elements defines the number of channels used.

For example, to use the first, second, and fourth , enter

```
[1,2,4]
```

Number the lines beginning with 1, even if this 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 (`-1` means sample time is inherited).

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

External Websites

www.humusoft.cz

Humusoft MF624 Encoder Input

HUMUSOFT MF624 Encoder Input block

Library

Simulink Real-Time 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 4. This driver allows the selection of individual channels in arbitrary order. The number of elements defines the number of channels used.

For example, to use the first, second, and fourth , enter

```
[1,2,4]
```

Number the lines beginning with 1, even if this 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 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 (`-1` means sample time is inherited).

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

External Websites

www.humusoft.cz

Humusoft MF624 PWM Output

HUMUSOFT MF624 PWM Output block

Library

Simulink Real-Time 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 4. This driver allows the selection of individual channels in arbitrary order. The number of elements defines the number of channels used.

For example, to use the first, second, and fourth , enter

```
[1,2,4]
```

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

Sample time

Enter the base sample time or a multiple of the base sample time (`-1` means sample time is inherited).

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

External Websites

www.humusoft.cz

Humusoft MF634

HUMUSOFT MF634 I/O board

Board

HUMUSOFT MF634

General Description

The MF634 provides:

- 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
- 4 quadrature encoder inputs with single-ended or differential input

The Simulink Real-Time block library supports this board with these driver blocks:

- Humusoft MF634 Analog Input (A/D)
- Humusoft MF634 Analog Output (D/A)
- Humusoft MF634 Digital Input
- Humusoft MF634 Digital Output
- Humusoft MF634 Counter Input
- Humusoft MF634 Encoder Input
- Humusoft MF634 PWM Output

Board Characteristics

Board name	MF634

Manufacturer	HUMUSOFT
Bus type	PCI
Access method	Memory mapped
Multiple block instance support	Digital Input: Yes, Others: No
Multiple board support	Yes

See Also

External Websites

www.humusoft.cz

Humusoft MF634 Analog Input (A/D)

HUMUSOFT MF634 Analog Input block

Library

Simulink Real-Time Library for HUMUSOFT

Scaling Input to Output

Hardware Input	Block Output Data Type	Scaling
Volts	Double	1

Block Parameters

Channel Vector

Enter numbers between 1 and 8. This driver allows the selection of individual channels in arbitrary order. The number of elements defines the number of channels used.

For example, to use the first, second, and fifth channels, enter

```
[1,2,5]
```

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

Sample time

Enter the base sample time or a multiple of the base sample time (`-1` means sample time is inherited).

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

External Websites

www.humusoft.cz

Humusoft MF634 Analog Output (D/A)

HUMUSOFT MF634 Analog Output block

Library

Simulink Real-Time Library for HUMUSOFT

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 channels in arbitrary order. The number of elements defines the number of channels used.

For example, to use the first, second, and fifth channels, enter

```
[1,2,5]
```

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

Sample time

Enter the base sample time or a multiple of the base sample time (`-1` means sample time is inherited).

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

External Websites

www.humusoft.cz

Humusoft MF634 Digital Input

HUMUSOFT MF634 Digital Input block

Library

Simulink Real-Time Library for HUMUSOFT

Scaling Input to Output

Hardware Input	Block Output Data Type	Scaling
TTL	Double	< 0.8 = TTL low ≥ 0.8 = TTL high

Channel Vector

Enter numbers between 1 and 8. This driver allows the selection of individual channels in arbitrary order. The number of elements defines the number of channels used.

For example, to use the first, second, and fifth channels, enter

```
[1,2,5]
```

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

Sample time

Enter the base sample time or a multiple of the base sample time (`-1` means sample time is inherited).

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

External Websites

www.humusoft.cz

Humusoft MF634 Digital Output

HUMUSOFT MF634 Digital Output block

Library

Simulink Real-Time 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 channels in arbitrary order. The number of elements defines the number of channels used.

For example, to use the first, second, and fifth channels, enter

```
[1,2,5]
```

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

Sample time

Enter the base sample time or a multiple of the base sample time (`-1` means sample time is inherited).

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

External Websites

www.humusoft.cz

Humusoft MF634 Counter Input

HUMUSOFT MF634 Counter Input block

Library

Simulink Real-Time 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 4. This driver allows the selection of individual channels in arbitrary order. The number of elements defines the number of channels used.

For example, to use the first, second, and fourth , enter

```
[1,2,4]
```

Number the lines beginning with 1, even if this 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 (`-1` means sample time is inherited).

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

External Websites

www.humusoft.cz

Humusoft MF634 Encoder Input

HUMUSOFT MF634 Encoder Input block

Library

Simulink Real-Time 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 4. This driver allows the selection of individual channels in arbitrary order. The number of elements defines the number of channels used.

For example, to use the first, second, and fourth , enter

```
[1,2,4]
```

Number the lines beginning with 1, even if this 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 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 (`-1` means sample time is inherited).

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

External Websites

www.humusoft.cz

Humusoft MF634 PWM Output

HUMUSOFT MF634 PWM Output block

Library

Simulink Real-Time 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 4. This driver allows the selection of individual channels in arbitrary order. The number of elements defines the number of channels used.

For example, to use the first, second, and fourth , enter

```
[1,2,4]
```

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

Sample time

Enter the base sample time or a multiple of the base sample time (`-1` means sample time is inherited).

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

External Websites

www.humusoft.cz

MathWorks, Measurement Computing, MIL-STD-1553, MPL

MathWorks

xPC TargetBox Support

This topic describes the xPC TargetBox boards and blocks supported by Simulink Real-Time. xPC TargetBox is an industrial target computer that is optimized for executing real-time code generated with Simulink Real-Time, Simulink Coder, and a C/C++ compiler. MathWorks® supplies and supports the blocks and documentation for users who already own this system.

xPC TargetBox is not available from MathWorks. Consider Speedgoat for other recommendations. See “Speedgoat Real-Time Target Machines” on page 47-2.

xPC TargetBox I/O Options

The following table lists the supported Simulink Real-Time boards that correspond to xPC TargetBox I/O options.

I/O Option Name	Simulink Real-Time Equivalent
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`

Measurement Computing

This topic describes Measurement Computing (formerly Computer Boards) I/O boards supported by the Simulink Real-Time product (`www.measurementcomputing.com`)

Measurement Computing PC104-DAC06 (/12)

Support for the MC PC104–DAC06 (/12) I/O board

Board

Measurement Computing PC104-DAC06 (/12)

General Description

The PC104-DAC06 (12) provides six analog output (D/A) channels (12-bit).

The Simulink Real-Time 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

See Also

External Websites

www.measurementcomputing.com

Measurement Computing PC104-DAC06 (/12) Analog Output (D/A)

PC104-DAC06 (/12) Analog Output block

Library

Simulink Real-Time Library for Measurement Computing

Scaling Input to Output

I/O Module Output	Block Input Data Type	Scaling
Volts	Double	1

Block Parameters

Channel vector

Enter numbers between 1 and 6. This board allows the selection of individual D/A channels in arbitrary 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 this board manufacturer starts numbering the channels with 0.

Range vector

Range code for each of the channels in the channel vector. The range vector must be the same length as the channel vector. This board allows the range of each channel to be different.

The following table is a list of the ranges for this driver and the corresponding range codes.

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

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 replicated over the channel vector. If you specify a value of `1`, the corresponding channel is reset to the value specified in the initial value vector. If you specify a value of `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 replicated as the initial value over the channel vector. 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 (`-1` means sample time is inherited).

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 must be in the XFER position. The Wait-State jumper has to be in the off position.

See Also

External Websites

www.measurementcomputing.com

Measurement Computing PC104-DAS16JR/12

Support for the MC PC104–DAS16JR/12 I/O board

Board

Measurement Computing PC104-DAS16JR/12

General Description

The PC104-DAS16JR/12 provides:

- 16 single or 8 differential analog input channels (12-bit) with a maximum sample rate of 150 kHz
- 4 digital input lines
- 4 digital output lines

The Simulink Real-Time 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

See Also

External Websites

www.measurementcomputing.com

Measurement Computing PC104-DAS16JR/12 Analog Input (A/D)

PC104-DAS16JR/12 Analog Input block

Library

Simulink Real-Time Library for Measurement Computing

Scaling Input to Output

I/O Module Input	Block Output Data Type	Scaling
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 this board manufacturer numbers the lines beginning 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 (`-1` means sample time is inherited).

Base address

Enter the base address of the board. This entry must correspond to the DIP switch setting on the board. For example, if the DIP switch setting is 300 (hexadecimal), enter `0x300`.

See Also

External Websites

www.measurementcomputing.com

Measurement Computing PC104-DAS16JR/12 Digital Input

PC104-DAS16JR/12 Digital Input block

Library

Simulink Real-Time Library for Measurement Computing

Scaling Input to Output

I/O Module 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 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 (`-1` means sample time is inherited).

Base address

Enter the base address of the board. This entry must correspond to the DIP switch setting on the board. For example, if the DIP switch setting is 300 (hexadecimal), enter `0x300`.

See Also

External Websites

www.measurementcomputing.com

Measurement Computing PC104-DAS16JR/12 Digital Output

PC104-DAS16JR/12 Digital Output block

Library

Simulink Real-Time Library for Measurement Computing

Scaling Input to Output

I/O Module 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 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 (`-1` means sample time is inherited).

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 remaining switch and jumper settings do not influence the running of the Simulink Real-Time software.

See Also

External Websites

www.measurementcomputing.com

Measurement Computing PC104-DAS16JR/16

Support for the MC PC104–DAS16JR/16 I/O board

Board

Measurement Computing PC104-DAS16JR/16

General Description

The PC104-DAS16JR/16 provides:

- 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

The Simulink Real-Time 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

See Also

External Websites

www.measurementcomputing.com

Measurement Computing PC104-DAS16JR/16 Analog Input (A/D)

PC104-DAS16JR/16 Analog Input block

Library

Simulink Real-Time Library for Measurement Computing

Scaling Input to Output

I/O Module Input	Block Output Data Type	Scaling
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 this board manufacturer numbers the lines beginning with 0.

Range

From the list, choose either `+-10V` (-10 volts to +10 volts), `+-5V`, `+-2.5V`, `+-1.25V`, `+-0.625V`, `0-10V`, `0-5V`, `0-2.5V`, or `0-1.25V`. This driver does not allow the selection of different range for each channel.

Input coupling

From the list, select one from the following list of input modes:

- `Single-ended channels (16 channels)`
- `Differential channels (8 channels)`

  This choice must correspond to the MUX-switch setting on the board.

Sample time

Enter the base sample time or a multiple of the base sample time (`-1` means sample time is inherited).

Base address

Enter the base address of the board. This entry must correspond to the DIP switch setting on the board. For example, if the DIP switch setting is 300 (hexadecimal), enter `0x300`.

See Also

External Websites

www.measurementcomputing.com

Measurement Computing PC104-DAS16JR/16 Digital Input

PC104-DAS16JR/16 Digital Input block

Library

Simulink Real-Time Library for Measurement Computing

Scaling Input to Output

I/O Module 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 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 (`-1` means sample time is inherited).

Base address

Enter the base address of the board. This entry must correspond to the DIP switch setting on the board. For example, if the DIP switch setting is 300 (hexadecimal), enter `0x300`.

See Also

External Websites

www.measurementcomputing.com

Measurement Computing PC104-DAS16JR/16 Digital Output

PC104-DAS16JR/16 Digital Output

Library

Simulink Real-Time Library for Measurement Computing

Scaling Input to Output

I/O Module 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 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 (`-1` means sample time is inherited).

Base address

Enter the base address of the board. This entry must correspond to the DIP switch setting on the board. For example, if the DIP switch setting is 300 (hexadecimal), enter `0x300`.

See Also

External Websites

www.measurementcomputing.com

Measurement Computing PC104-DIO48

Support for the MC PC104–DIO48 I/O board

Board

Measurement Computing PC104-DIO48

General Description

The PC104-DIO48 provides 48 digital I/O lines.

The Simulink Real-Time block library supports this board with these driver blocks:

- Measurement Computing PC104-DIO48 Digital Input
- Measurement Computing PC104-DIO48 Digital Output

Board Characteristics

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

See Also

External Websites

www.measurementcomputing.com

Measurement Computing PC104-DIO48 Digital Input

PC104-DIO48 Digital Input block

Library

Simulink Real-Time 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.

Scaling Input to Output

I/O Module 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 arbitrary 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 this board manufacturer numbers the lines beginning with 0.

Port

From the list choose either `A`, `B`, or `C`. The I/O board has two 8255 chips with 3 ports. The **Port** parameter defines which port of the 8255 chip is used for this driver block. Each port has a maximum or 8 digital lines that can be configured as inputs or outputs depending on which driver block is chosen. In each case, one block is 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 the base sample time or a multiple of the base sample time (`-1` means sample time is inherited).

Base address

Enter the base address of the board. This entry must correspond to the DIP switch setting on the board. For example, if the DIP switch setting is 300 (hexadecimal), enter `0x300`.

See Also

External Websites

www.measurementcomputing.com

Measurement Computing PC104-DIO48 Digital Output

PC104-DIO48 Digital Output block

Library

Simulink Real-Time 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

I/O Module 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 arbitrary 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 this board manufacturer numbers the lines beginning with 0.

Port

From the list choose either `A`, `B`, or `C`. The I/O board has two 8255 chips with 3 ports. The **Port** parameter defines which port of the 8255 chip is used for this driver block. Each port has a maximum or 8 digital lines that can be configured as inputs or outputs depending on which driver block is chosen. In each case, one block is 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 replicated over the channel vector. If you specify a value of `1`, the corresponding channel is reset to the value specified in the initial value vector. If you specify a value of `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 replicated as the initial value over the channel vector. 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 the base sample time or a multiple of the base sample time (`-1` means sample time is inherited).

Base address

Enter the base address of the board. This entry must correspond to the DIP switch setting on the board. For example, if the DIP switch setting is 300 (hexadecimal), enter `0x300`.

See Also

External Websites

www.measurementcomputing.com

Measurement Computing PCI-DAC6703

Support for the MC PCI-DAC6703 I/O board

Board

Measurement Computing PCI-DAC6703

General Description

The PCI-DAC6703 provides 16 analog output (D/A) channels (16-bit) and 8 digital input and output lines.

The Simulink Real-Time 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

See Also

External Websites

www.measurementcomputing.com

Measurement Computing PCI-DAC6703 Analog Output (D/A)

PCI-DAC6703 Analog Output block

Library

Simulink Real-Time Library for Measurement Computing

Scaling Input to Output

I/O Module Output	Block Input Data Type	Scaling
Volts	Double	1

Block Parameters

Channel vector

Enter numbers between 1 and 16. This driver allows the selection of individual D/A in arbitrary 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 if this board manufacturer numbers the lines 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 replicated over the channel vector. If you specify a value of `1`, the corresponding channel is reset to the value specified in the initial value vector. If you specify a value of `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 replicated as the initial value over the channel vector. 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 (`-1` means sample time is inherited).

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

External Websites

www.measurementcomputing.com

Measurement Computing PCI-DAC6703 Digital Input

PCI-DAC6703 Digital Input block

Library

Simulink Real-Time Library for Measurement Computing

Scaling Input to Output

I/O Module 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. This driver allows the selection of individual digital input lines in arbitrary 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 if this board manufacturer numbers the lines beginning with 0.

Sample time

Enter the base sample time or a multiple of the base sample time (`-1` means sample time is inherited).

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

External Websites

www.measurementcomputing.com

Measurement Computing PCI-DAC6703 Digital Output

PCI-DAC6703 Digital Output block

Library

Simulink Real-Time Library for Measurement Computing

Scaling Input to Output

I/O Module 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 arbitrary 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 if this board manufacturer numbers the lines 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 replicated over the channel vector. If you specify a value of `1`,

the corresponding channel is reset to the value specified in the initial value vector. If you specify a value of `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 replicated as the initial value over the channel vector. 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 (`-1` means sample time is inherited).

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

External Websites

www.measurementcomputing.com

Measurement Computing PCI-DDA02/12

Support for the MC PCI-DDA02/12 I/O board

Board

Measurement Computing PCI-DDA02/12

General Description

The PCI-DDA02/12) provides 2 analog output (D/A) channels (12-bit), and 48 digital I/O lines.

The Simulink Real-Time 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

See Also

External Websites

www.measurementcomputing.com

Measurement Computing PCI-DDA02/12 Analog Output (D/A)

PCI-DDA02/12 Analog Output block

Library

Simulink Real-Time Library for Measurement Computing

Scaling Input to Output

I/O Module 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 arbitrary 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 this 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

Input Range (V)	Range Code
-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

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 replicated over the channel vector. If you specify a value of `1`, the corresponding channel is reset to the value specified in the initial value vector. If you specify a value of `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 replicated as the initial value over the channel vector. 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 (`-1` means sample time is inherited).

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

External Websites

www.measurementcomputing.com

Measurement Computing PCI-DDA02/12 Digital Input

PCI-DDA02/12 Digital Input

Library

Simulink Real-Time 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

I/O Module 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 arbitrary 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 this board manufacturer numbers the lines beginning with 0.

Port

From the list choose either `A`, `B`, or `C`. The I/O board has two 8255 chips with 3 ports. The **Port** parameter defines which port of the 8255 chip is used for this driver block. Each port has a maximum or 8 digital lines that can be configured as inputs or outputs depending on which driver block is chosen. In each case, one block is 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 the base sample time or a multiple of the base sample time (`-1` means sample time is inherited).

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

External Websites

www.measurementcomputing.com

Measurement Computing PCI-DDA02/12 Digital Output

PCI-DDA02/12 Digital Output

Library

Simulink Real-Time 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.

Scaling Input to Output

I/O Module 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 arbitrary 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 this board manufacturer numbers the lines beginning with 0.

Port

From the list choose either `A`, `B`, or `C`. The I/O board has two 8255 chips with 3 ports. The **Port** parameter defines which port of the 8255 chip is used for this driver block. Each port has a maximum or 8 digital lines that can be configured as inputs or outputs depending on which driver block is chosen. In each case, one block is 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 replicated over the channel vector. If you specify a value of `1`, the corresponding channel is reset to the value specified in the initial value vector. If you specify a value of `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 replicated as the initial value over the channel vector. 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 the base sample time or a multiple of the base sample time (`-1` means sample time is inherited).

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
```

```
getPCIInfo(tg, 'installed')
```

See Also

External Websites

www.measurementcomputing.com

Measurement Computing PCI-DDA02/16

Support for the MC PCI-DDA02/16 I/O board

Board

Measurement Computing PCI-DDA02/16

General Description

The PCI-DDA02/16) provides 2 analog output (D/A) channels (12-bit), and 48 digital I/O lines.

The Simulink Real-Time 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

See Also

External Websites

www.measurementcomputing.com

Measurement Computing PCI-DDA02/16 Analog Output (D/A)

PCI-DDA02/16 Analog Output block

Library

Simulink Real-Time Library for Measurement Computing

Scaling Input to Output

I/O Module 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 arbitrary 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 this 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

Input Range (V)	Range Code
-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

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 replicated over the channel vector. If you specify a value of `1`, the corresponding channel is reset to the value specified in the initial value vector. If you specify a value of `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 replicated as the initial value over the channel vector. 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 (`-1` means sample time is inherited).

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

External Websites

www.measurementcomputing.com

Measurement Computing PCI-DDA02/16 Digital Input

PCI-DDA02/16 Digital Input

Library

Simulink Real-Time 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

I/O Module 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 arbitrary 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 this board manufacturer numbers the lines beginning with 0.

Port

From the list choose either `A`, `B`, or `C`. The I/O board has two 8255 chips with 3 ports. The **Port** parameter defines which port of the 8255 chip is used for this driver block. Each port has a maximum or 8 digital lines that can be configured as inputs or outputs depending on which driver block is chosen. In each case, one block is 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 the base sample time or a multiple of the base sample time (`-1` means sample time is inherited).

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

External Websites

www.measurementcomputing.com

Measurement Computing PCI-DDA02/16 Digital Output

PCI-DDA02/16 Digital Output

Library

Simulink Real-Time 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.

Scaling Input to Output

I/O Module 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 arbitrary 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 this board manufacturer numbers the lines beginning with 0.

Port

From the list choose either `A`, `B`, or `C`. The I/O board has two 8255 chips with 3 ports. The **Port** parameter defines which port of the 8255 chip is used for this driver block. Each port has a maximum or 8 digital lines that can be configured as inputs or outputs depending on which driver block is chosen. In each case, one block is 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 replicated over the channel vector. If you specify a value of `1`, the corresponding channel is reset to the value specified in the initial value vector. If you specify a value of `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 replicated as the initial value over the channel vector. 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 the base sample time or a multiple of the base sample time (`-1` means sample time is inherited).

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
```

```
getPCIInfo(tg, 'installed')
```

See Also

External Websites

www.measurementcomputing.com

Measurement Computing PCI-DDA04/12

Support for the MC PCI-DDA04/12 I/O board

Board

Measurement Computing PCI-DDA04/12

General Description

The PCI-DDA04/12) provides 4 analog output (D/A) channels (12-bit), and 48 digital I/O lines.

The Simulink Real-Time 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

See Also

External Websites

www.measurementcomputing.com

Measurement Computing PCI-DDA04/12 Analog Output (D/A)

PCI-DDA04/12 Analog Output block

Library

Simulink Real-Time Library for Measurement Computing

Scaling Input to Output

I/O Module Output	Block Input Data Type	Scaling
Volts	Double	1

Block Parameters

Channel vector

Enter numbers between 1 and 4. This driver allows the selection of individual D/A channels in arbitrary 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 this 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

Input Range (V)	Range Code
-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

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 replicated over the channel vector. If you specify a value of `1`, the corresponding channel is reset to the value specified in the initial value vector. If you specify a value of `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 replicated as the initial value over the channel vector. 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 (`-1` means sample time is inherited).

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

External Websites

www.measurementcomputing.com

Measurement Computing PCI-DDA04/12 Digital Input

PCI-DDA04/12 Digital Input

Library

Simulink Real-Time 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.

Scaling Input to Output

I/O Module 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 arbitrary 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 this board manufacturer numbers the lines beginning with 0.

Port

From the list choose either `A`, `B`, or `C`. The I/O board has two 8255 chips with 3 ports. The **Port** parameter defines which port of the 8255 chip is used for this driver block. Each port has a maximum or 8 digital lines that can be configured as inputs or outputs depending on which driver block is chosen. In each case, one block is 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 the base sample time or a multiple of the base sample time (`-1` means sample time is inherited).

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

External Websites

www.measurementcomputing.com

Measurement Computing PCI-DDA04/12 Digital Output

PCI-DDA04/12 Digital Output

Library

Simulink Real-Time 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.

Scaling Input to Output

I/O Module 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 arbitrary 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 this board manufacturer numbers the lines beginning 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 replicated over the channel vector. If you specify a value of `1`, the corresponding channel is reset to the value specified in the initial value vector. If you specify a value of `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 replicated as the initial value over the channel vector. 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 the base sample time or a multiple of the base sample time (`-1` means sample time is inherited).

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
```

```
getPCIInfo(tg, 'installed')
```

See Also

External Websites

www.measurementcomputing.com

Measurement Computing PCI-DDA04/16

Support for the MC PCI-DDA04/16 I/O board

Board

Measurement Computing PCI-DDA04/16

General Description

The PCI-DDA04/16) provides 4 analog output (D/A) channels (16-bit), and 48 digital I/O lines.

The Simulink Real-Time 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

See Also

External Websites

www.measurementcomputing.com

Measurement Computing PCI-DDA04/16 Analog Output (D/A)

PCI-DDA04/16 Analog Output block

Library

Simulink Real-Time Library for Measurement Computing

Scaling Input to Output

I/O Module Output	Block Input Data Type	Scaling
Volts	Double	1

Block Parameters

Channel vector

Enter numbers between 1 and 4. This driver allows the selection of individual D/A channels in arbitrary 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 this 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

Input Range (V)	Range Code
-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

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 replicated over the channel vector. If you specify a value of `1`, the corresponding channel is reset to the value specified in the initial value vector. If you specify a value of `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 replicated as the initial value over the channel vector. 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 (`-1` means sample time is inherited).

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

External Websites

www.measurementcomputing.com

Measurement Computing PCI-DDA04/16 Digital Input

PCI-DDA04/16 Digital Input block

Library

Simulink Real-Time 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.

Scaling Input to Output

I/O Module 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 arbitrary 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 this board manufacturer numbers the lines beginning with 0.

Port

From the list choose either `A`, `B`, or `C`. The I/O board has two 8255 chips with 3 ports. The **Port** parameter defines which port of the 8255 chip is used for this driver block. Each port has a maximum or 8 digital lines that can be configured as inputs or outputs depending on which driver block is chosen. In each case, one block is 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 the base sample time or a multiple of the base sample time (`-1` means sample time is inherited).

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

External Websites

www.measurementcomputing.com

Measurement Computing PCI-DDA04/16 Digital Output

PCI-DDA04/16 Digital Output

Library

Simulink Real-Time 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.

Scaling Input to Output

I/O Module 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 arbitrary 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 this board manufacturer numbers the lines beginning 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 replicated over the channel vector. If you specify a value of `1`, the corresponding channel is reset to the value specified in the initial value vector. If you specify a value of `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 replicated as the initial value over the channel vector. 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 the base sample time or a multiple of the base sample time (`-1` means sample time is inherited).

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
```

```
getPCIInfo(tg, 'installed')
```

See Also

External Websites

www.measurementcomputing.com

Measurement Computing PCI-DDA08/12

Support for the MC PCI-DDA08/12 I/O board

Board

Measurement Computing PCI-DDA08/12

General Description

The PCI-DDA08/12) provides 8 analog output (A/D) channels (16-bit), and 48 digital I/O lines.

The Simulink Real-Time 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

See Also

External Websites

www.measurementcomputing.com

Measurement Computing PCI-DDA08/12 Analog Output (D/A)

PCI-DDA08/12 Analog Output block

Library

Simulink Real-Time Library for Measurement Computing

Scaling Input to Output

I/O Module 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 arbitrary 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 this 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

Input Range (V)	Range Code
-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

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 replicated over the channel vector. If you specify a value of `1`, the corresponding channel is reset to the value specified in the initial value vector. If you specify a value of `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 replicated as the initial value over the channel vector. 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 (`-1` means sample time is inherited).

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

External Websites

www.measurementcomputing.com

Measurement Computing PCI-DDA08/12 Digital Input

PCI-DDA08/12 Digital Input block

Library

Simulink Real-Time 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.

Scaling Input to Output

I/O Module 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 arbitrary 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 this board manufacturer numbers the lines beginning with 0.

Port

From the list choose either `A`, `B`, or `C`. The I/O board has two 8255 chips with 3 ports. The **Port** parameter defines which port of the 8255 chip is used for this driver block. Each port has a maximum or 8 digital lines that can be configured as inputs or outputs depending on which driver block is chosen. In each case, one block is 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 the base sample time or a multiple of the base sample time (`-1` means sample time is inherited).

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

External Websites

www.measurementcomputing.com

Measurement Computing PCI-DDA08/12 Digital Output

PCI-DDA08/12 Digital Output block

Library

Simulink Real-Time 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.

Scaling Input to Output

I/O Module 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 arbitrary 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 this board manufacturer numbers the lines beginning with 0.

Port

From the list choose either `A`, `B`, or `C`. The I/O board has two 8255 chips with 3 ports. The **Port** parameter defines which port of the 8255 chip is used for this driver block. Each port has a maximum or 8 digital lines that can be configured as inputs or outputs depending on which driver block is chosen. In each case, one block is 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 replicated over the channel vector. If you specify a value of `1`, the corresponding channel is reset to the value specified in the initial value vector. If you specify a value of `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 replicated as the initial value over the channel vector. 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 the base sample time or a multiple of the base sample time (`-1` means sample time is inherited).

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
```

```
getPCIInfo(tg, 'installed')
```

See Also

External Websites

www.measurementcomputing.com

Measurement Computing PCI-DDA08/16

Support for the MC PCI-DDA08/16 I/O board

Board

Measurement Computing PCI-DDA08/16

General Description

The PCI-DDA08/16) provides 8 analog output (A/D) channels (12-bit), and 48 digital I/O lines.

The Simulink Real-Time 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

See Also

External Websites

www.measurementcomputing.com

Measurement Computing PCI-DDA08/16 Analog Output (D/A)

PCI-DDA08/16 Analog Output block

Library

Simulink Real-Time Library for Measurement Computing

Scaling Input to Output

I/O Module 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 arbitrary 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 this 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

Input Range (V)	Range Code
-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

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 replicated over the channel vector. If you specify a value of `1`, the corresponding channel is reset to the value specified in the initial value vector. If you specify a value of `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 replicated as the initial value over the channel vector. 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 (`-1` means sample time is inherited).

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

External Websites

www.measurementcomputing.com

Measurement Computing PCI-DDA08/16 Digital Input

PCI-DDA08/16 Digital Input

Library

Simulink Real-Time 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.

Scaling Input to Output

I/O Module 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 arbitrary 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 this board manufacturer numbers the lines beginning with 0.

Port

From the list choose either `A`, `B`, or `C`. The I/O board has two 8255 chips with 3 ports. The **Port** parameter defines which port of the 8255 chip is used for this driver block. Each port has a maximum or 8 digital lines that can be configured as inputs or outputs depending on which driver block is chosen. In each case, one block is 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 the base sample time or a multiple of the base sample time (`-1` means sample time is inherited).

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

External Websites

www.measurementcomputing.com

Measurement Computing PCI-DDA08/16 Digital Output

PCI-DDA08/16 Digital Output

Library

Simulink Real-Time Library for Measurement Computing

Note

The PCI-DDA08/16 has two 8255 chips (1,2). Each chip has three ports (A,B,C). Each port has a maximum of 8 digital I/O lines that can be configured as inputs or outputs.

Use a separate driver block for each port. By selecting the digital output driver block, the port is configured as output.

Scaling Input to Output

I/O Module 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 arbitrary 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 this board manufacturer numbers the lines beginning with 0.

Port

From the list choose either `A`, `B`, or `C`. The I/O board has two 8255 chips with 3 ports. The **Port** parameter defines which port of the 8255 chip is used for this driver block. Each port has a maximum or 8 digital lines that can be configured as inputs or outputs depending on which driver block is chosen. In each case, one block is 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 replicated over the channel vector. If you specify a value of `1`, the corresponding channel is reset to the value specified in the initial value vector. If you specify a value of `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 replicated as the initial value over the channel vector. 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 the base sample time or a multiple of the base sample time (`-1` means sample time is inherited).

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
```

```
getPCIInfo(tg, 'installed')
```

See Also

External Websites

www.measurementcomputing.com

Measurement Computing PCI-DIO24

Support for the MC PCI-DIO24 I/O board

Board

Measurement Computing PCI-DIO24

General Description

The PCI-DIO24 provides 24 digital I/O lines.

The Simulink Real-Time block library supports this board with these driver blocks:

- `Measurement Computing PCI-DIO24 Digital Input`
- `Measurement Computing PCI-DIO24 Digital Output`

Board Characteristics

Board name	PCI-DIO24
Manufacturer	Measurement Computing
Bus type	PCI
Access method	I/O mapped
Multiple block instance support	Yes
Multiple board support	Yes

See Also

External Websites

www.measurementcomputing.com

Measurement Computing PCI-DIO24 Digital Input

PCI-DIO24 Digital Input

Library

Simulink Real-Time Library for Measurement Computing

Note

The PCI-DIO24 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, the port is configured as input.

Scaling Input to Output

I/O Module 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 arbitrary 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 this board manufacturer numbers the lines beginning 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 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 (`-1` means sample time is inherited).

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

External Websites

www.measurementcomputing.com

Measurement Computing PCI-DIO24 Digital Output

PCI-DIO24 Digital Output

Library

Simulink Real-Time Library for Measurement Computing

Note

The PCI-DIO24 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 output driver block, the port is configured as output.

Scaling Input to Output

I/O Module 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 arbitrary 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 this board manufacturer numbers the lines beginning 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 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 replicated over the channel vector. If you specify a value of `1`, the corresponding channel is reset to the value specified in the initial value vector. If you specify a value of `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 replicated as the initial value over the channel vector. 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 (`-1` means sample time is inherited).

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

External Websites

www.measurementcomputing.com

Measurement Computing PCI-DIO24H

Support for the MC PCI-DIO24H I/O board

Board

Measurement Computing PCI-DIO24H

General Description

The PCI-DIO24H provides 24 digital I/O lines.

The Simulink Real-Time block library supports this board with these driver blocks:

- Measurement Computing PCI-DIO24H Digital Input
- Measurement Computing PCI-DIO24H Digital Output

Board Characteristics

Board name	PCI-DIO24H
Manufacturer	Measurement Computing
Bus type	PCI
Access method	I/O mapped
Multiple block instance support	Yes
Multiple board support	Yes

See Also

External Websites

www.measurementcomputing.com

Measurement Computing PCI-DIO24H Digital Input

PCI-DIO24H Digital Input

Library

Simulink Real-Time Library for Measurement Computing

Note

The PCI-DIO24H 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, the port is configured as input.

Scaling Input to Output

I/O Module 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 arbitrary 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 this board manufacturer numbers the lines beginning 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 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 (`-1` means sample time is inherited).

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

External Websites

www.measurementcomputing.com

Measurement Computing PCI-DIO24H Digital Output

PCI-DIO24H Digital Output

Library

Simulink Real-Time Library for Measurement Computing

Note

The PCI-DIO24H 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 output driver block, the port is configured as output.

Scaling Input to Output

I/O Module 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 arbitrary 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 this board manufacturer numbers the lines beginning 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 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 replicated over the channel vector. If you specify a value of `1`, the corresponding channel is reset to the value specified in the initial value vector. If you specify a value of `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 replicated as the initial value over the channel vector. 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 (`-1` means sample time is inherited).

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

External Websites

www.measurementcomputing.com

Measurement Computing PCI-DIO48H

Support for the MC PCI-DIO48H I/O board

Board

Measurement Computing PCI-DIO48H

General Description

The PCI-DIO48H provides 48 digital I/O lines.

The Simulink Real-Time block library supports this board with these driver blocks:

- Measurement Computing PCI-DIO48H Digital Input
- Measurement Computing PCI-DIO48H Digital Output

Board Characteristics

Board name	PCI-DIO48H
Manufacturer	Measurement Computing
Bus type	PCI
Access method	I/O mapped
Multiple block instance support	Yes
Multiple board support	Yes

See Also

External Websites

www.measurementcomputing.com

Measurement Computing PCI-DIO48H Digital Input

PCI-DIO48H Digital Input

Library

Simulink Real-Time 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.

Scaling Input to Output

I/O Module 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 arbitrary 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 this board manufacturer numbers the lines beginning with 0.

Port

From the list choose either `A`, `B`, or `C`. The I/O board has two 8255 chips with 3 ports. The **Port** parameter defines which port of the 8255 chip is used for this driver block. Each port has a maximum or 8 digital lines that can be configured as inputs or outputs depending on which driver block is chosen. In each case, one block is 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 the base sample time or a multiple of the base sample time (`-1` means sample time is inherited).

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

External Websites

www.measurementcomputing.com

Measurement Computing PCI-DIO48H Digital Output

PCI-DIO48H Digital Output

Library

Simulink Real-Time 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.

Scaling Input to Output

I/O Module 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 arbitrary 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 this board manufacturer numbers the lines beginning with 0.

Port

From the list choose either `A`, `B`, or `C`. The I/O board has two 8255 chips with 3 ports. The **Port** parameter defines which port of the 8255 chip is used for this driver block. Each port has a maximum or 8 digital lines that can be configured as inputs or outputs depending on which driver block is chosen. In each case, one block is 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 replicated over the channel vector. If you specify a value of `1`, the corresponding channel is reset to the value specified in the initial value vector. If you specify a value of `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 replicated as the initial value over the channel vector. 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 the base sample time or a multiple of the base sample time (`-1` means sample time is inherited).

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
```

```
getPCIInfo(tg, 'installed')
```

See Also

External Websites

www.measurementcomputing.com

Measurement Computing PCI-DIO96H

Support for the MC PCI-DIO96H I/O board

Board

Measurement Computing PCI-DIO96H

General Description

The PCI-DIO96H provides 96 digital I/O lines.

The Simulink Real-Time block library supports this board with these driver blocks:

- Measurement Computing PCI-DIO96H Digital Input
- Measurement Computing PCI-DIO96H Digital Output

Board Characteristics

Board name	PCI-DIO96H
Manufacturer	Measurement Computing
Bus type	PCI
Access method	I/O mapped
Multiple block instance support	Yes
Multiple board support	Yes

See Also

External Websites

www.measurementcomputing.com

Measurement Computing PCI-DIO96H Digital Input

PCI-DIO96H Digital Input

Library

Simulink Real-Time 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

I/O Module 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 arbitrary 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 this board manufacturer numbers the lines beginning with 0.

Port

From the list choose either `A`, `B`, or `C`. The I/O board has two 8255 chips with 3 ports. The **Port** parameter defines which port of the 8255 chip is used for this driver block. Each port has a maximum or 8 digital lines that can be configured as inputs or outputs depending on which driver block is chosen. In each case, one block is required for each port.

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 the base sample time or a multiple of the base sample time (`-1` means sample time is inherited).

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

External Websites

www.measurementcomputing.com

Measurement Computing PCI-DIO96H Digital Output

PCI-DIO96H Digital Output

Library

Simulink Real-Time 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

I/O Module 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 arbitrary 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 this board manufacturer numbers the lines beginning with 0.

Port

From the list choose either `A`, `B`, or `C`. The I/O board has two 8255 chips with 3 ports. The **Port** parameter defines which port of the 8255 chip is used for this driver block. Each port has a maximum or 8 digital lines that can be configured as inputs or outputs depending on which driver block is chosen. In each case, one block is 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 replicated over the channel vector. If you specify a value of `1`, the corresponding channel is reset to the value specified in the initial value vector. If you specify a value of `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 replicated as the initial value over the channel vector. 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 the base sample time or a multiple of the base sample time (`-1` means sample time is inherited).

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
```

```
getPCIInfo(tg, 'installed')
```

See Also

External Websites

www.measurementcomputing.com

Measurement Computing PCI-DIO96

Support for the MC PCI-DIO96 I/O board

Board

Measurement Computing PCI-DIO96

General Description

The PCI-DIO96 provides 96 digital I/O lines.

The Simulink Real-Time block library supports this board with these driver blocks:

- Measurement Computing PCI-DIO96 Digital Input
- Measurement Computing PCI-DIO96 Digital Output

Board Characteristics

Board name	PCI-DIO96
Manufacturer	Measurement Computing
Bus type	PCI
Access method	I/O mapped
Multiple block instance support	Yes
Multiple board support	Yes

See Also

External Websites

www.measurementcomputing.com

Measurement Computing PCI-DIO96 Digital Input

PCI-DIO96 Digital Input block

Library

Simulink Real-Time 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.

Scaling Input to Output

I/O Module 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 arbitrary 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 this board manufacturer numbers the lines beginning 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.

Sample time

Enter the base sample time or a multiple of the base sample time (`-1` means sample time is inherited).

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

External Websites

www.measurementcomputing.com

Measurement Computing PCI-DIO96 Digital Output

PCI-DIO96 Digital Output block

Library

Simulink Real-Time 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.

Scaling Input to Output

I/O Module 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 arbitrary 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 this board manufacturer numbers the lines beginning 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 replicated over the channel vector. If you specify a value of `1`, the corresponding channel is reset to the value specified in the initial value vector. If you specify a value of `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 replicated as the initial value over the channel vector. 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 the base sample time or a multiple of the base sample time (`-1` means sample time is inherited).

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

External Websites

www.measurementcomputing.com

Measurement Computing PCI-PDISO8

Support for the PCI-PDISO8 I/O board

Board

Measurement Computing PCI-PDISO8

General Description

The PCI-PDISO8 provides eight inputs and eight relay outputs.

The Simulink Real-Time 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

See Also

External Websites

www.measurementcomputing.com

Measurement Computing PCI-PDISO8 Digital Input

PCI-PDISO8 Digital Input block

Library

Simulink Real-Time Library for Measurement Computing

Scaling Input to Output

I/O Module Input	Block Output Data Type	Scaling
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. Without an 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 arbitrary 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 if this board manufacturer numbers the lines beginning 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 replicated across the channel vector.

Port

From the list choose `A` because this board only has eight channels.

Sample time

Enter the base sample time or a multiple of the base sample time (`-1` means sample time is inherited).

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

External Websites

www.measurementcomputing.com

Measurement Computing PCI-PDISO8 Digital Output

PCI-PDISO8 Digital Output

Library

Simulink Real-Time Library for Measurement Computing

Scaling Input to Output

I/O Module Output	Block Input Data Type	Scaling
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 arbitrary 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 if this board manufacturer numbers the lines beginning with 0.

Port

From the list choose `A` because this board only has eight channels.

Sample time

Enter the base sample time or a multiple of the base sample time (`-1` means sample time is inherited).

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

External Websites

www.measurementcomputing.com

Measurement Computing PCI-PDISO16

Support for the MC PCI-PDISO16 I/O board

Board

Measurement Computing PCI-PDISO16

General Description

The PCI-PDISO16 provides 16 inputs and 16 relay outputs.

The Simulink Real-Time block library supports this board with these driver blocks:

- Measurement Computing PCI-PDISO16 Digital Input
- Measurement Computing PCI-PDISO16 Digital Output

Board Characteristics

Board name	PCI-PDISO16
Manufacturer	Measurement Computing
Bus type	PCI
Access method	Memory mapped
Multiple block instance support	Yes
Multiple board support	Yes

See Also

External Websites

www.measurementcomputing.com

Measurement Computing PCI-PDISO16 Digital Input

PCI-PDISO16 Digital Input block

Library

Simulink Real-Time Library for Measurement Computing

Scaling Input to Output

I/O Module Input	Block Output Data Type	Scaling
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. Without an 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 arbitrary 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 if this board manufacturer numbers the lines beginning 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 replicated across the channel vector.

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 the base sample time or a multiple of the base sample time (`-1` means sample time is inherited).

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

External Websites

www.measurementcomputing.com

Measurement Computing PCI-PDISO16 Digital Output

PCI-PDISO16 Digital Output block

Library

Simulink Real-Time Library for Measurement Computing

Scaling Input to Output

I/O Module Output	Block Input Data Type	Scaling
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 arbitrary 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 if this board manufacturer numbers the lines beginning 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 the base sample time or a multiple of the base sample time (`-1` means sample time is inherited).

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

External Websites

www.measurementcomputing.com

Measurement Computing PCIM-DAS1602/16

Support for the MC PCIM-DAS1602/16 I/O board

Board

Measurement Computing PCIM-DAS1602/16

General Description

The PCIM-DAS1602/16 provides:

- 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)
- 24 digital input lines
- 24 digital output lines
- 3 counters (16-bit)

Note: The Simulink Real-Time software supports only 24 digital I/O lines and does not support the counters on this board.

The Simulink Real-Time 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

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

See Also

External Websites

www.measurementcomputing.com

Measurement Computing PCIM-DAS1602/16 Analog Input (A/D)

PCIM-DAS1602/16 Analog Input block

Library

Simulink Real-Time Library for Measurement Computing

Scaling of Input to Output

I/O Module Input	Block Output Data Type	Scaling
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 this board manufacturer numbers the lines beginning 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)`

  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 (`-1` means sample time is inherited).

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

External Websites

www.measurementcomputing.com

Measurement Computing PCIM-DAS1602/16 Analog Output (D/A)

PCIM-DAS1602/16 Analog Output block

Library

Simulink Real-Time Library for Measurement Computing

Scaling of Input to Output

I/O Module 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 arbitrary 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 this 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.

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

Input Range (V)	Range Code
-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]
```

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 replicated over the channel vector. If you specify a value of `1`, the corresponding channel is reset to the value specified in the initial value vector. If you specify a value of `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 replicated as the initial value over the channel vector. 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 (`-1` means sample time is inherited).

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
```

```
getPCIInfo(tg, 'installed')
```

See Also

External Websites

www.measurementcomputing.com

Measurement Computing PCIM-DAS 1602/16 Digital Input

PCIM-DAS 1602/16 Digital Input block

Library

Simulink Real-Time Library for Measurement Computing

Note

The DAS1601/16 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 of Input to Output

I/O Module 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 arbitrary 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 this board manufacturer numbers the lines beginning 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 the base sample time or a multiple of the base sample time (`-1` means sample time is inherited).

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

External Websites

www.measurementcomputing.com

Measurement Computing PCIM-DAS1602/16 Digital Output

PCIM-DAS1602/16 Digital Output

Library

Simulink Real-Time Library for Measurement Computing

Note

The PCIM-DAS1601/16 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.

Scaling of Input to Output

I/O Module 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 arbitrary 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 this board manufacturer numbers the lines beginning 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 replicated over the channel vector. If you specify a value of `1`, the corresponding channel is reset to the value specified in the initial value vector. If you specify a value of `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 replicated as the initial value over the channel vector. 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 (`-1` means sample time is inherited).

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

External Websites

www.measurementcomputing.com

Measurement Computing PCIM-DDA06/16

Support for the MC PCIM-DDA06/16 I/O board

Board

Measurement Computing PCIM-DDA06/16

General Description

The PCIM-DDA06/16) provides 6 analog output (D/A) channels (16-bit), and 24 digital I/O lines.

The Simulink Real-Time 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

See Also

External Websites

www.measurementcomputing.com

Measurement Computing PCIM-DDA06/16 Analog Output (D/A)

PCIM-DDA06/16 Analog Output block

Library

Simulink Real-Time Library for Measurement Computing

Scaling Input to Output

I/O Module 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 arbitrary 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 this 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

Input Range (V)	Range Code
-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
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 Simulink Real-Time 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 replicated over the channel vector. If you specify a value of `1`, the corresponding channel is reset to the value specified in the initial value vector. If you specify a value of `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 replicated as the initial value over the channel vector. 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 (`-1` means sample time is inherited).

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

External Websites

www.measurementcomputing.com

Measurement Computing PCIM-DDA06/16 Digital Input

PCIM-DDA06/16 Digital Input

Library

Simulink Real-Time 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

I/O Module 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 arbitrary 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 this board manufacturer numbers the lines beginning with 0.

Port

From the list choose either `A`, `B`, or `C`. The I/O board has two 8255 chips with 3 ports. The **Port** parameter defines which port of the 8255 chip is used for this driver block. Each port has a maximum or 8 digital lines that can be configured as inputs or outputs depending on which driver block is chosen. In each case, one block is required for each port.

Sample time

Enter the base sample time or a multiple of the base sample time (`-1` means sample time is inherited).

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

External Websites

www.measurementcomputing.com

Measurement Computing PCIM-DDA06/16 Digital Output

PCIM-DDA06/16 Digital Output

Library

Simulink Real-Time 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.

Scaling Input to Output

I/O Module 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 arbitrary 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 this board manufacturer numbers the lines beginning 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 replicated over the channel vector. If you specify a value of `1`, the corresponding channel is reset to the value specified in the initial value vector. If you specify a value of `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 replicated as the initial value over the channel vector. 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 (`-1` means sample time is inherited).

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

External Websites

www.measurementcomputing.com

Measurement Computing PCI-DUAL-AC5

Support for the MC PCI-DUAL-AC5 I/O board

Board

Measurement Computing PCI-DUAL-AC5

General Description

The PCI-DUAL-AC5 provides 48 digital I/O lines.

The Simulink Real-Time block library supports this board with these driver blocks:

- Measurement Computing PCI-DUAL-AC5 Digital Input
- Measurement Computing PCI-DUAL-AC5 Digital Output

Board Characteristics

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

See Also

External Websites

www.measurementcomputing.com

Measurement Computing PCI-DUAL-AC5 Digital Input

PCI-DUAL-AC5 Digital Input

Library

Simulink Real-Time 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

I/O Module 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 arbitrary 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 this board manufacturer numbers the lines beginning with 0.

Port

From the list choose either `A`, `B`, or `C`. The I/O board has two 8255 chips with 3 ports. The **Port** parameter defines which port of the 8255 chip is used for this driver block. Each port has a maximum or 8 digital lines that can be configured as inputs or outputs depending on which driver block is chosen. In each case, one block is 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 the base sample time or a multiple of the base sample time (`-1` means sample time is inherited).

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

External Websites

www.measurementcomputing.com

Measurement Computing PCI-DUAL-AC5 Digital Output

PCI-DUAL-AC5 Digital Output

Library

Simulink Real-Time 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

I/O Module 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 arbitrary 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 this board manufacturer numbers the lines beginning with 0.

Port

From the list choose either `A`, `B`, or `C`. The I/O board has two 8255 chips with 3 ports. The **Port** parameter defines which port of the 8255 chip is used for this driver block. Each port has a maximum or 8 digital lines that can be configured as inputs or outputs depending on which driver block is chosen. In each case, one block is 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 replicated over the channel vector. If you specify a value of `1`, the corresponding channel is reset to the value specified in the initial value vector. If you specify a value of `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 replicated as the initial value over the channel vector. 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 the base sample time or a multiple of the base sample time (`-1` means sample time is inherited).

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
```

```
getPCIInfo(tg, 'installed')
```

See Also

External Websites

www.measurementcomputing.com

Measurement Computing PCI-QUAD04

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 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 Simulink Real-Time block library supports this board with this driver block:

- Measurement Computing PCI-QUAD04 Incremental Encoder

There is a physical limitation with the PCI-Quad04 boards in the way that they handle unconnected inputs. Because of noise on the index input line, the counter chip can latch a value that differs from the actual value by a multiple of 256. This problem occurs even with the **Index input resets counter** check box disabled. 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.

To minimize crosstalk at high count frequencies, make connections as close as possible to the board.

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

See Also

External Websites

www.measurementcomputing.com

Measurement Computing PCI-QUAD04 Incremental Encoder

PCI-QUAD04 Incremental Encoder

Library

Simulink Real-Time 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. The counter direction depends on the encoder manufacturer and how the encoder is installed.

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.

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`	Upon model start, the output count is initialized to **Count Limit or Reset Value**. The output count increments or decreases one full revolution from the **Count Limit or Reset Value**. The output count is reset to this value each time the index is reached. For example, if **Index input resets counter** is selected, and a rotary encoder has 1024 counts per revolution, and a **Count Limit or Reset Value** of 5000, a full clockwise revolution of the encoder increments the counter to 6023, at which the index is reached. The counter then resets to `5000`.	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.
`Range Limit`	Upon model start, the counter increments or decrements until it reaches the count limit **Count Limit or Reset Value**. If the clock motion is clockwise, the counter stays at **Count Limit or Reset Value** once it reaches that value. If the clock motion is counterclockwise, the counter decrements one revolution then resets to **Count Limit or Reset Value**.	Upon model start, the counter is initialized to 0. In the clockwise direction, the counter increments until it saturates at **Count Limit or Reset Value**. In the counterclockwise direction, the counter decrements until it saturates at `0`. 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

Counting Mode	Index input resets counter Selected	Index input resets counter Deselected
		negative direction stop. At that point, the encoder reports 0 at the stop and the other positions will be positive and repeatable.
`Non-recycle`	Upon model start, the counter is initialized to **Count Limit or Reset Value**. 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 **Count Limit or Reset Value**. With clockwise rotation, the counter increments until the encoder reaches the index. When the encoder reaches the index, the counter resets to **Count Limit or Reset Value**. 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 **Count Limit or Reset Value**, then resets to 0 and repeats. When the counter decrements, it counts to 0, then resets to **Count Limit or Reset Value** and continues to decrement. You can use this setting to repeat intervals that do not match the distance between indexes (resulting, for example, from gear ratios).

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...2^24-1` or `-2^23...+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.

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 computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

External Websites

www.measurementcomputing.com

Measurement Computing PCI-DAS-TC

Support for the MC PCI-DAS-TC I/O board

Board

Measurement Computing PCI-DAS-TC

General Description

The PCI-DAS-TC provides 16 differential analog thermocouple input channels. A high frequency synchronous V-F A/D converts the thermocouple signals. 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 onboard micro-controller off loads the main CPU of these tasks.

The Simulink Real-Time 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

See Also

External Websites

www.measurementcomputing.com

Measurement Computing PCI-DAS-TC Thermocouple

PCI-DAS-TC Thermocouple

Library

Simulink Real-Time Library for Measurement Computing

Scaling Input to Output

I/O Module Input	Block Input Data Type	Scaling
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 an 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 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 the base sample time or a multiple of the base sample time (`-1` means sample time is inherited).

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

Each time a real-time application containing this driver block is downloaded to the target computer, the board automatically does a full calibration. Thermocouple sensor calibration is an extensive procedure. Because takes place for each channel independently, the calibration time can easily exceed several seconds, especially when five or more channels are being calibrated. Because of this long calibration period during the initialization stage of the real-time application, the download procedure can time out and return an error message. To avoid this error, increase the default timeout duration. See “Communications Timeout”.

See Also

External Websites

www.measurementcomputing.com

MIL-STD-1553 Support

MIL-STD-1553 Support

Simulink Real-Time supports the MIL-STD-1553 bus via the PCI-1553, QPCI-1553, QPCX-1553, and Q104 series boards. The manufacturer of these boards is Abaco, formerly GE Intelligent Platforms, formerly Condor Engineering, see the manufacturer documentation (`www.abaco.com`). Simulink Real-Time uses the MIL-STD-1553 blocks provided by the MIL-STD-1553 sublibrary of the Simulink Real-Time I/O block library. 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-1553, QPCI-1553, QPCX-1553 — Use these blocks to communicate with the PCI-1553, QPCI-1553, and QPCX-1553 boards, respectively. 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.

  In basic functionality, these board types are the same. The QPC*-1553 boards have additional functionality, such as internal loopback connections and voltage settings.
- 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.

  The QPC* and Q104 boards have almost the same functionality. The Q104 does not support loopback connections, and you address it by PC104 base address instead of by PCI slot.

Regardless of the operation, initialize your board with the Initialize block for the board type you are using—PCI, QPCI, QPCX, or Q104.

MIL-STD-1553 Initialization

The Initialize block allows you to specify the board operation, which can be Remote Terminal, Bus Controller, or Bus Monitor. The mask dialog box for this block changes depending on the operation that you select.

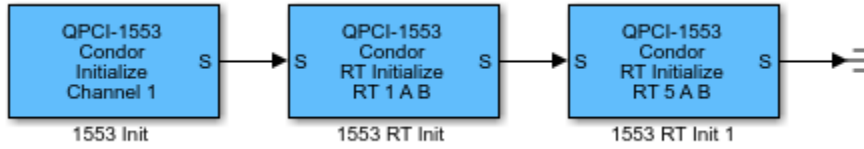
- Select the **Initialize for Bus Controller operation** check box for the Bus Controller.
- Select the **Initialize for Bus Monitor operation** check box for the Bus Monitor.
- Select the **Initialize for Remote Terminal operation** check box for the Remote Terminal.

By default, these check boxes are selected. If you clear an operation check box, the block makes the associated parameters unavailable.

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 example and other examples are located in the `xpcdemos` folder. 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 sub addresses and legal message lengths for each sub address.

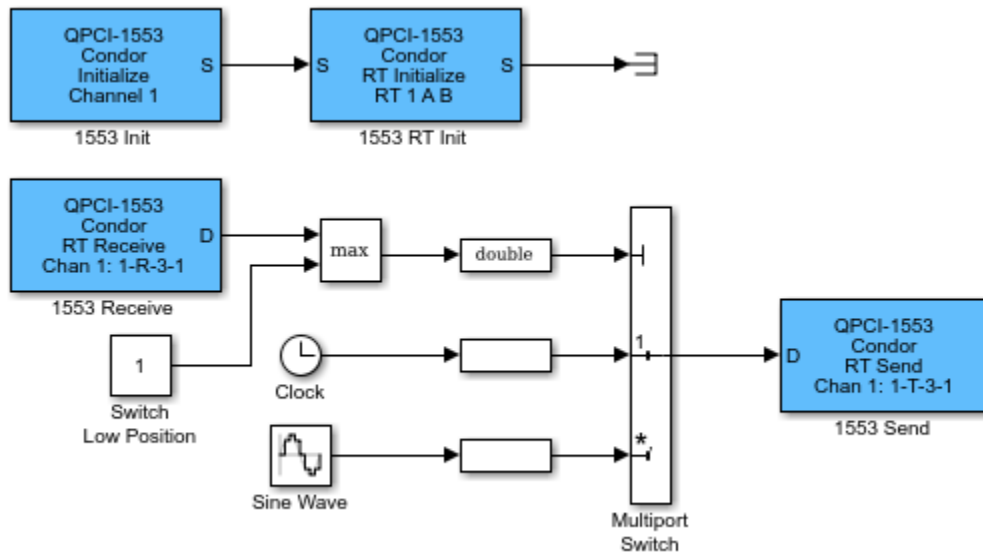
You can configure each sub address for transmit, receive, or both. Configure the sub addresses that you plan to use.



Remote Terminal Operation

The following example uses QPCI-1553 blocks to illustrate how you can configure a Remote Terminal from a Simulink Real-Time model. For standard initialization, use the QPCI Initialize block. Use the QPCI RT Initialize block to set up the board for Remote Terminal operation.

You can run this example on a target computer that has a QPCI-1553 board. If your target computer uses a different board, such as a PCI-1553 board, replace the QPCI-1553 blocks with the blocks for your board.



The QPCI-1553 RT 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 sub addresses. The QPCI-1553 Receive and Send blocks use sub address 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 sub address for the message.

`Number of words` — Indicates the number of 16-bit integers to receive as the data part of a message.

This notation is shorthand that indicates that Remote Terminal 1 is to receive messages on sub address 3 with a length of one 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.

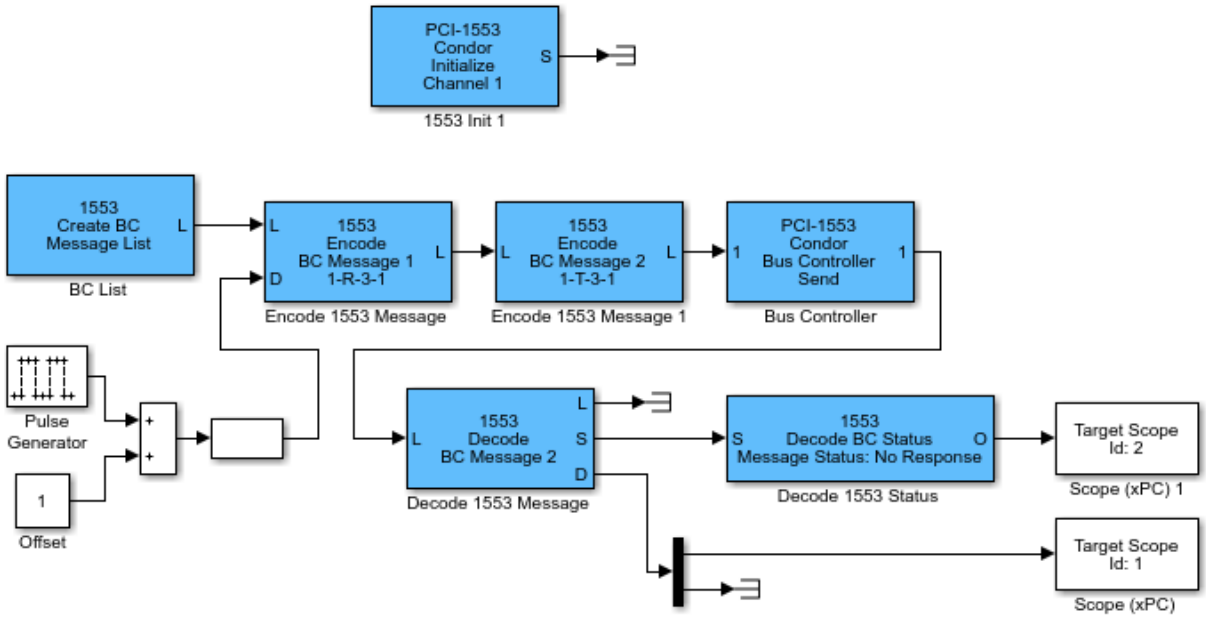
A QPCI-1553 board has a processor that handles the actual transmission and reception of messages. The QPCI-1553 Receive block reads the board receive message buffer for the specified Remote Terminal and sub address. The QPCI-1553 Send block writes data to the board transmit buffer.

Note: When the board receives a transmit or receive message, it must respond quickly because the default timeout is 14 μs. Simulink Real-Time cannot reliably execute a model within so short a timeout period. This situation can 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.

You can run this example on a target computer that has a PCI-1553 board. If your target computer uses a different board, such as a QPCI-1553 board, replace the PCI-1553 blocks with the blocks for your board.



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 send.

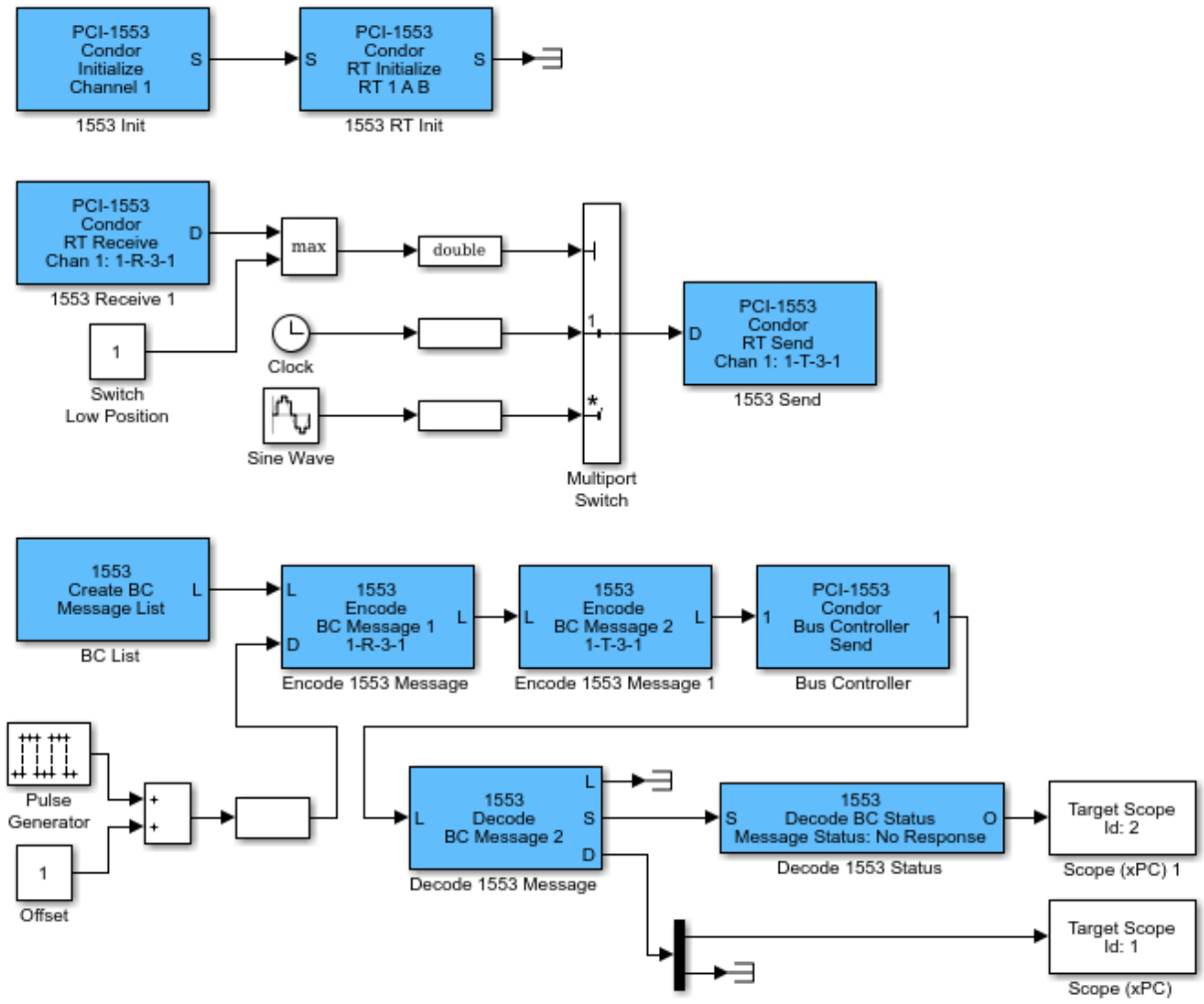
The Create BC Message List block allocates an empty list of five message buffers in Simulink Real-Time memory. Each time this block executes, it sets the entire list of message buffers to no-op messages. The Encode BC Message block fills each message for the time step during which the message must be sent. To send only a specific message at a specific time, enclose the Encode BC Message block in an enabled subsystem.

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. From there, the signal goes to the Bus Controller Send block, executing the Encode blocks before the Send block.

Remote Terminal and Bus Controller Operation

The following example 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`.

The `RTBCSample` real-time application executes 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 defined 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, sub address 3. This word has a value of either `1` or `2`. An Add block creates

this value by adding the constant value `1` to the output of the Pulse Generator, which outputs either `0` or `1`.

Message 2 of Bus Controller message list is a command for Remote Terminal 1, sub address 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 (microseconds)**). 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 arbitrary 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.

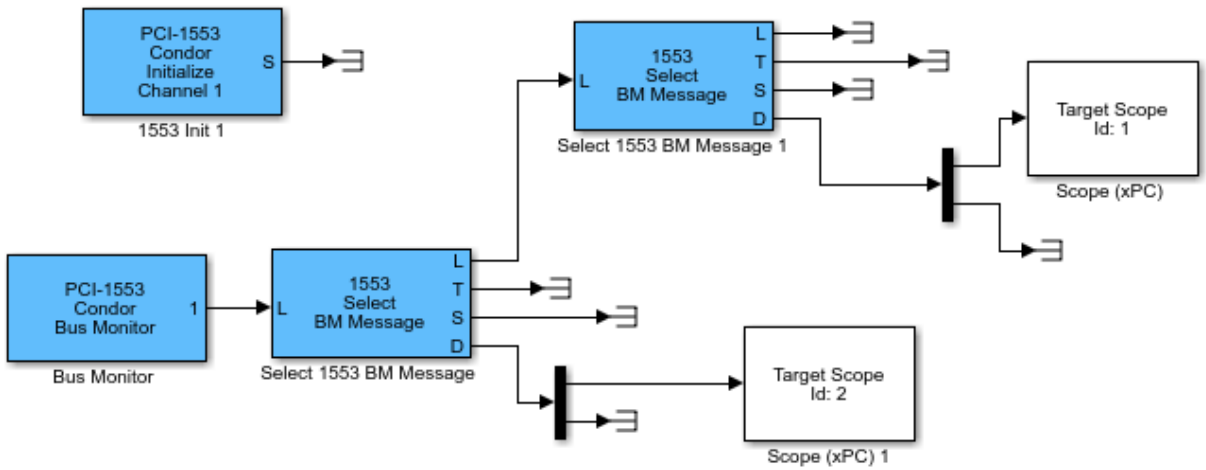
The Decode BC Message block has the following outputs.

- L — Message list passed to other Decode BC Message blocks. If your model contains only one Decode BC Message block, 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 has the maximum message width of 32 `uint16` values. The actual message determines how many of these values are significant. In this example, only the first value is significant. The block sends the last 31 values to a terminator.

Bus Monitor Operation

The following example uses PCI-1553 blocks. It describes how you can use a Bus Monitor in a model. The Bus Monitor operation outputs a list containing the messages seen on the specified bus since the last time it was called. The `BMSample` model shows how to set up a simple bus monitor that looks for the messages in the preceding examples (`RTSample`, `BCSample`, and `RTBCSample`).

You can run this example on a target computer that has a PCI-1553 board. If your target computer uses a different board, such as a QPCI-1553 board, replace the PCI-1553 blocks with the blocks for your board.



The PCI-1553 Initialize block configures a Bus Monitor on channel 1 of the board. This block also instructs the board to monitor bus A.

The PCI-1553 Bus Monitor block specifies a maximum of 10 messages to receive in a list.

The 1553 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 1553 Select BM Message block has the following outputs:

L — Message list passed to other1553 Select BM Message blocks. If this block is the only 1553 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 time is the time, in microseconds, since the board was started, presented as a double. The clock can run at a slightly different rate than the model execution timer. Therefore, this time is likely different from the Simulink Real-Time execution time.

S — Status information. This information contains seven `uint32` entries with status and command information.

D — A vector of 32 uint16 entries with the data from that message. This vector outputs 32 entries, even if only some of the entries are defined.

MPL

This topic describes the MPL board supported by the Simulink Real-Time product (`www.mpl.ch`).

MPL PATI

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. The Simulink Real-Time software supports these channels for PWM measurement, PWM generation, time base selection, incremental encoder measurement, and digital I/O. The PATI board is based on the Freescale™ MPC555 processor. The Simulink Real-Time software supports only the TPU channels for this board. The Simulink Real-Time block library supports the MPL PATI 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
- S3 — off
- S4 — on (enable restart)
- S5 — off
- S6 — off
- S7 — off
- S8 — off

See the MPL PATI user documentation for a description of the DIP switch S1. Some target computer CPU boards support only two PC/104+ boards, others support up to four.

Board Characteristics

Board name	MPL PATI
Manufacturer	MPL
Bus type	PC/104
Multiple block instance support	Yes
Multiple board support	Yes

See Also

External Websites

www.mpl.ch

MPL PATI Digital Input

MPL PATI Digital Input block

Library

Simulink Real-Time 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. 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 and build and execute that model. You can then execute 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 changes 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 (`-1` means sample time is inherited).

PCI slot

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

External Websites

www.mpl.ch

MPL PATI Digital Output

MPL PATI Digital Output block

Library

Simulink Real-Time 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. 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. You can then execute 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 changes the order that individual timers are serviced. See the MPC555 TPU documentation for more information.

Reset

This parameter controls the behavior of the bit at model termination. Enter one of the following values:

- `1`

Resets the output bit to the value in **Initial Value**.

- `0`

  Leaves the output bit at the most recent value attained while the model was running.

Initial value

Enter the initial voltage value for the output bit. The output bit is set to this value between the time the model is downloaded and the time the model is started. Also, if the **Reset** parameter has a value of `1`, the output bit is reset to the initial value when the model is stopped.

Sample time

Enter the base sample time or a multiple of the base sample time (`-1` means sample time is inherited).

PCI slot

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

External Websites

www.mpl.ch

MPL PATI Incremental Encoder

MPL PATI Incremental Encoder block

Library

Simulink Real-Time Library for MPL

General Description

This block has one input port and one output port.

- The block `M` input port determines the counting mode. The incremental encoder function has two counting modes, normal and fast.

  - 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. Set the counting mode to normal when the model is first started and 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.

    In fast mode, the polarity of the two signals does not control the count direction. Instead, the count continues in the same direction as when the counter was last in normal mode. The maximum counting rate is much higher in fast mode than in normal mode.

- The block output port returns the current position count as a double.

To keep the counting mode set to normal mode, connect a constant block with the value `0` to the `M` 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. You can then execute 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 changes 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 (`-1` means sample time is inherited).

PCI slot

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

External Websites

www.mpl.ch

MPL PATI PWM generate

MPL PATI PWM generate block

Library

Simulink Real-Time Library for MPL

Scaling Output to Input

I/O Module Output	Block Input Data Type	Scaling
0–5 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 `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 remains high. You can use a count in the range `0`–`32767`. When you use multiple channels, getting your output width to `0` width smoothly is difficult. If you have a few channels, you can try to use the **Priority** parameter to more smoothly approach `0` width.
- `L` — Number of counts of the chosen time base that the output remains low. You can use a count in the range `0`–`32767`. When you use multiple channels, getting your output width to `0` width smoothly is difficult. If you have a few channels, you can try to use the **Priority** parameter to more smoothly approach `0` width.
- `A` — Specifies whether 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 box. If the `A` is `1`, the output runs. This port does not appear if you do not select **Arm input**. The block behaves as if the `A` input is `1`/

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. You can then execute 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. Even if the output frequency is the same, the relative timing is not repeatable within a few tens of microseconds.

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 changes the order that individual timers are serviced. See the MPC555 TPU documentation for more information.

Initial high count

Enter the high count to be used from initialization until the model is running.

Initial low count

Enter the low count to be used from initialization until the model is running.

Immediate update

Select this check box to force the output to use the new count immediately. Selecting this check box can result in very short output pulses. It terminates long pulses immediately.

Clear this check box to allow the current output cycle to finish before the update. Clearing this check box generates smoother frequency and phase transitions without transients.

The block does not interrupt long pulses. Note that your selection of this parameter value depends on your system requirements.

Arm input

Select this check box to indicate that the A 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 A input becomes 1 and the output port starts changing.

- If the input signal to this port is 1, the channel outputs the signal specified by the H and L ports.
- If the input signal is to this port is 0, the output stops at the level specified by the **Disarm state** drop-down list.

If you do not select this check box, the block behaves as if the A input is 1. Input port A does not appear.

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 (`-1` means sample time is inherited).

PCI slot

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

See Also

`MPL PATI Timebase Setup`

External Websites

www.mpl.ch

MPL PATI PWM measure

MPL PATI PWM measure

Library

Simulink Real-Time Library for MPL

Scaling Input to Output

I/O Module Input	Block Output Data Type	Scaling
0–5 volts	Double	Counts of current time base

General Description

This block measures the full cycle period or high or low time in an input waveform with the PTA module in the time processor units (TPUs). To use the TPUs, you must condition the TPU inputs with a voltage of 0–5 volts. The inputs also must have a reasonably fast rise or fall time; otherwise, false triggering can occur. The inputs use TTL level transitions. See the MPC555 documentation from Freescale Semiconductor at `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. You can then execute 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 changes 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 (`-1` means sample time is inherited).

PCI slot

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
```

```
getPCIInfo(tg, 'installed')
```

See Also

See Also

MPL PATI Timebase Setup

External Websites

www.mpl.ch

MPL PATI Timebase Setup

MPL PATI Timebase Setup block

Library

Simulink Real-Time Library for MPL

General Description

The Freescale MPC555 processor contains two time processor units (TPUs). Each TPU has 16 timer channels (1–16 and 17–32), 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. You can set the frequency for each of the four time bases, two time bases for channels 1–16 and two for channels 17–32.

Each TPU allows two different timer sources, TCR1 and TCR2.

- The board drives TCR1 by a clock source with a frequency of 40 MHz/*N*. *N* is a power of 2 in the range 2–512.
- The board drives TCR2 either by an external clock source or by a clock source with a frequency of 40 MHz/*N*. *N* is a power of 2 in the range 8–32. The MPL PATI user documentation refers to the channel's 17–32 external input as TPU_T2CLKB.

The I/O module architecture restricts allowable values for the time bases. See the MPC555 documentation from Freescale Semiconductor at `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. You can then execute the actual model. See `MPL PATI EEPROM Write` block for further information.

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 MHz/9 = 4.44 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 MHz/9 = 4.44 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 computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

External Websites

www.mpl.ch

MPL PATI EEPROM Write

MPL PATI EEPROM Write

Library

Simulink Real-Time Library for MPL

General Description

The MPL PATI EEPROM Write block writes MPL boot code on the EEPROM. Before executing a Simulink Real-Time 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 reflash the EEPROM, include this block by itself in an empty model. To write the boot code on the EEPROM, run this model only once. Restart the target computer before executing the actual model. Do not include this block in another model.

To flash the EEPROM with the boot code:

1. Add the MPL PATI EEPROM Write block to an empty model. To access the library that contains this block, type `xpcmpllib`.
2. Build and execute this model. The work is completed when the model is downloaded to the target computer.
3. Restart the target computer.
4. Build and execute your actual model.

Block Parameters

PCI slot

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

External Websites

www.mpl.ch

National Instruments

National Instruments Blocks

This topic describes National Instruments I/O boards supported by the Simulink Real-Time product (`www.ni.com`).

National Instruments PCI-6011E (formerly PCI-MIO-16XE-50)

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 provides:

- 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
- 2 counter/timers (24-bit) with a maximum source clock rate of 20 MHz

The Simulink Real-Time 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)

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

See Also

External Websites

www.ni.com

National Instruments PCI-6011E Analog Input (A/D) (formerly PCI-MIO-16XE-50)

PCI-6011E Analog Input block

Library

Simulink Real-Time Library for National Instruments

Scaling of Input to Output

I/O Module Input	Block Output Data Type	Scaling
Volts	Double	1

Block Parameters

Channel vector

Enter numbers between 1 and 16. This driver allows you to enter channel numbers in arbitrary order.

For example, to use the first, second, and fifth channels, enter

```
[1,2,5]
```

Number the channels beginning with 1, even if this 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.

Input Range (V)	Range Code
-10 to +10	-10
-5 to +5	-5
-1 to +1	-1
-0.1 to +0.1	-0.1
0 to +10	10
0 to +5	5
0 to +1	1
0 to +0.1	0.1

For example, if the first channel is -10 volts to +10 volts and the second and fifth channels are 0 volts 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 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

Enter the base sample time or a multiple of the base sample time (`-1` means sample time is inherited).

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

External Websites

www.ni.com

National Instruments PCI-6011E Analog Output (D/A) (formerly PCI-MIO-16XE-50)

PCI-6011E Analog Output block

Library

Simulink Real-Time Library for National Instruments

Note

The analog output range of this board is set -10 volts to +10 volts.

Scaling of Input to Output

I/O Module 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 arbitrary 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 this 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 replicated over the channel vector. If you specify a value of `1`, the corresponding channel is reset to the value specified in the initial value vector. If you specify a value of `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 replicated as the initial value over the channel vector. 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 (`-1` means sample time is inherited).

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

External Websites

www.ni.com

National Instruments PCI-6011E Digital Input (formerly PCI-MIO-16XE-50)

PCI-6011E Digital Input block

Library

Simulink Real-Time Library for National Instruments

Scaling of Input to Output

I/O Module 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 arbitrary 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 this board manufacturer starts numbering the lines with 0.

Sample time

Enter the base sample time or a multiple of the base sample time (`-1` means sample time is inherited).

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

External Websites

www.ni.com

National Instruments PCI-6011E Digital Output (formerly PCI-MIO-16XE-50)

PCI-6011E Digital Output block

Library

Simulink Real-Time Library for National Instruments

Scaling of Input to Output

I/O Module 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 arbitrary 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 this 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 replicated over the channel vector. If you specify a value of `1`, the corresponding channel is reset to the value specified in the initial value vector.

If you specify a value of `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 replicated as the initial value over the channel vector. 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 (`-1` means sample time is inherited).

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

External Websites

www.ni.com

National Instruments PCI-6011E Pulse Generation (formerly PCI-MIO-16XE-50)

PCI-6011E Pulse Generation block

Library

Simulink Real-Time Library for National Instruments

Requirements

The input to the block is a two-element vector. The first element is the number of counts (of the 20 MHz source clock) that the counter maintains at a low level. 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

Enter the base sample time or a multiple of the base sample time (`-1` means sample time is inherited).

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

External Websites

www.ni.com

National Instruments PCI-6011E Pulse Width/Period Measurement (formerly PCI-MIO-16XE-50)

PCI-6011E Pulse Width/Period Measurement

Library

Simulink Real-Time 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 can experience false triggering. Use signal voltage levels of 0–5 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** interact. 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** interact. 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

Measurement Objective	Trigger Mode	Polarity	Output
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

Enter the base sample time or a multiple of the base sample time (`-1` means sample time is inherited).

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

External Websites

www.ni.com

National Instruments PCI-6023E

Support for National Instruments PCI-6023E board

Board

National Instruments PCI-6023E

General Description

The PCI-6023E provides:

- 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
- 2 counter/timers (24-bit) with a maximum source clock rate of 20 MHz

The Simulink Real-Time 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

See Also

External Websites

www.ni.com

National Instruments PCI-6023E Analog Input (A/D)

PCI-6023E Analog Input block

Library

Simulink Real-Time Library for National Instruments

Scaling Input to Output

I/O Module Input	Block Output Data Type	Scaling
Volts	Double	1

Block Parameters

Channel vector

Enter numbers between 1 and 16. This driver allows you to enter channel numbers in arbitrary order.

For example, to use the first, second, and fifth channels, enter

```
[1,2,5]
```

Number the channels beginning with 1 even if this 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

Input Range (V)	Range Code
-5 to +5	-5
-0.5 to +0.5	-0.5
-0.05 to +0.05	-0.05

For example, if the first channel is -10 volts 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]
```

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

Enter the base sample time or a multiple of the base sample time (`-1` means sample time is inherited).

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

External Websites

www.ni.com

National Instruments PCI-6023E Digital Input

PCI-6023E Digital Input

Library

Simulink Real-Time Library for National Instruments

Scaling Input to Output

I/O Module 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 arbitrary 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 this board manufacturer starts numbering the lines with 0.

Sample time

Enter the base sample time or a multiple of the base sample time (`-1` means sample time is inherited).

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

External Websites

www.ni.com

National Instruments PCI-6023E Digital Output

PCI-6023E Digital Output

Library

Simulink Real-Time Library for National Instruments

Scaling Input to Output

I/O Module 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 arbitrary 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 this 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 replicated over the channel vector. If you specify a value of `1`, the corresponding channel is reset to the value specified in the initial value vector. If you specify a value of `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 replicated as the initial value over the channel vector. 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 (`-1` means sample time is inherited).

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

External Websites

www.ni.com

National Instruments PCI-6023E Pulse Generation

PCI-6023E Pulse Generation

Library

Simulink Real-Time Library for National Instruments

Note

The input to the block is a two-element vector. The first element is the number of counts (of the 20 MHz source clock) that the counter maintains at a low level. 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

Enter the base sample time or a multiple of the base sample time (`-1` means sample time is inherited).

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

External Websites

www.ni.com

National Instruments PCI-6023E Pulse Width/Period Measurement

PCI-6023E Pulse Width/Period Measurement

Library

Simulink Real-Time 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 can experience false triggering. Use signal voltage levels of 0–5 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** interact. 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** interact. 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

Measurement Objective	Trigger Mode	Polarity	Output
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

Enter the base sample time or a multiple of the base sample time (`-1` means sample time is inherited).

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

External Websites

www.ni.com

National Instruments PCI-6024E

Support for National Instruments PCI-6024E board

Board

National Instruments PCI-6024E

General Description

The PCI-6024E provides:

- 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
- 2 counter/timers (24-bit) with a maximum source clock rate of 20 MHz

The Simulink Real-Time 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

See Also

External Websites

www.ni.com

National Instruments PCI-6024E Analog Input (A/D)

PCI-6024E Analog Input block

Library

Simulink Real-Time Library for National Instruments

Scaling Input to Output

I/O Module Input	Block Output Data Type	Scaling
Volts	Double	1

Block Parameters

Channel vector

Enter numbers between 1 and 16. This driver allows you to enter channel numbers in arbitrary order.

For example, to use the first, second, and fifth channels, enter

```
[1,2,5]
```

Number the channels beginning with 1 even if this 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

Input Range (V)	Range Code
-5 to +5	-5
-0.5 to +0.5	-0.5
-0.05 to +0.05	-0.05

For example, if the first channel is -10 volts 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]
```

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

Enter the base sample time or a multiple of the base sample time (`-1` means sample time is inherited).

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

External Websites

www.ni.com

National Instruments PCI-6024E Analog Output (D/A)

PCI-6024E Analog Output block

Library

Simulink Real-Time Library for National Instruments

Note

The analog output range of this board is set -10 volts to +10 volts.

Scaling Input to Output

I/O Module 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 arbitrary 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 this 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 replicated over the channel vector. If you specify a value of `1`, the corresponding channel is reset to the value specified in the initial value vector. If you specify a value of `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 replicated as the initial value over the channel vector. 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 (`-1` means sample time is inherited).

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

External Websites

www.ni.com

National Instruments PCI-6024E Digital Input

PCI-6024E Digital Input block

Library

Simulink Real-Time Library for National Instruments

Scaling Input to Output

I/O Module 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 arbitrary 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 this board manufacturer starts numbering the lines with 0.

Sample time

Enter the base sample time or a multiple of the base sample time (`-1` means sample time is inherited).

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

External Websites

www.ni.com

National Instruments PCI-6024E Digital Output

PCI-6024E Digital Output

Library

Simulink Real-Time Library for National Instruments

Scaling Input to Output

I/O Module 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 arbitrary 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 this 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 replicated over the channel vector. If you specify a value of `1`, the corresponding channel is reset to the value specified in the initial value vector. If you specify a value of `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 replicated as the initial value over the channel vector. 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 (`-1` means sample time is inherited).

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

External Websites

www.ni.com

National Instruments PCI-6024E Pulse Generation

PCI-6024E Pulse Generation

Library

Simulink Real-Time Library for National Instruments

Note

The input to the block is a two-element vector. The first element is the number of counts (of the 20 MHz source clock) that the counter maintains at a low level. 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

Enter the base sample time or a multiple of the base sample time (`-1` means sample time is inherited).

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

External Websites

www.ni.com

National Instruments PCI-6024E Pulse Width/Period Measurement

PCI-6024E Pulse Width/Period Measurement

Library

Simulink Real-Time 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 can experience false triggering. Use signal voltage levels of 0–5 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** interact. 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** interact. 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

Measurement Objective	Trigger Mode	Polarity	Output
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

Enter the base sample time or a multiple of the base sample time (`-1` means sample time is inherited).

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

External Websites

www.ni.com

National Instruments PCI-6025E

Support for National Instruments PCI-6025E board

Board

National Instruments PCI-6025E

General Description

The PCI-6025E provides:

- 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
- 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 Simulink Real-Time 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

See Also

External Websites

www.ni.com

National Instruments PCI-6025E Analog Input (A/D)

PCI-6025E Analog Input block

Library

Simulink Real-Time Library for National Instruments

Scaling Input to Output

I/O Module Input	Block Output Data Type	Scaling
Volts	Double	1

Block Parameters

Channel vector

Enter numbers between 1 and 16. This driver allows you to enter channel numbers in arbitrary order.

For example, to use the first, second, and fifth channels, enter

```
[1,2,5]
```

Number the channels beginning with 1 even if this 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

Input Range (V)	Range Code
-5 to +5	-5
-0.5 to +0.5	-0.5
-0.05 to +0.05	-0.05

For example, if the first channel is -10 volts 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]
```

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

Enter the base sample time or a multiple of the base sample time (`-1` means sample time is inherited).

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

External Websites

www.ni.com

National Instruments PCI-6025E Analog Output (D/A)

PCI-6025E Analog Output block

Library

Simulink Real-Time Library for National Instruments

Note

The analog output range of this board is set -10 volts to +10 volts.

Scaling Input to Output

I/O Module 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 arbitrary 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 this 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 replicated over the channel vector. If you specify a value of `1`, the corresponding channel is reset to the value specified in the initial value vector. If you specify a value of `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 replicated as the initial value over the channel vector. 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 (`-1` means sample time is inherited).

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

External Websites

www.ni.com

National Instruments PCI-6025E and PCI-6025E 8255 Digital Input

PCI-6025E and PCI-6025E 8255 Digital Input blocks

Library

Simulink Real-Time Library for National Instruments

Scaling Input to Output

I/O Module 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 arbitrary 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 this 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 the base sample time or a multiple of the base sample time (`-1` means sample time is inherited).

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

External Websites

www.ni.com

National Instruments PCI-6025E Digital Output

PCI-6025E Digital Output block

Library

Simulink Real-Time Library for National Instruments

Scaling Input to Output

I/O Module 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 arbitrary 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 this 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 replicated over the channel vector. If you specify a value of `1`, the corresponding channel is reset to the value specified in the initial value vector. If you specify a value of `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 replicated as the initial value over the channel vector. 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 (`-1` means sample time is inherited).

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

External Websites

www.ni.com

National Instruments PCI-6025E Pulse Generation

PCI-6025E Pulse Generation block

Library

Simulink Real-Time Library for National Instruments

Note

The input to the block is a two-element vector. The first element is the number of counts (of the 20 MHz source clock) that the counter maintains at a low level. 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

Enter the base sample time or a multiple of the base sample time (`-1` means sample time is inherited).

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

External Websites

www.ni.com

National Instruments PCI-6025E Pulse Width/Period Measurement

PCI-6025E Pulse Width/Period Measurement block

Library

Simulink Real-Time 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 can experience false triggering. Use signal voltage levels of 0–5 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** interact. 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** interact. 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

Measurement Objective	Trigger Mode	Polarity	Output
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

Enter the base sample time or a multiple of the base sample time (`-1` means sample time is inherited).

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

External Websites

www.ni.com

National Instruments PCI-6040E (formerly PCI-MIO-16E-4)

Support for National Instruments PCI-6040E board

Board

National Instruments PCI-6040E

General Description

This was formerly PCI-MIO-16E-4.

The PCI-6040E provides:

- 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
- 2 counter/timers (24-bit) with a maximum source clock rate of 20 MHz

The Simulink Real-Time 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)

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

See Also

External Websites

www.ni.com

National Instruments PCI-6040E Analog Input (A/D) (formerly PCI-MIO-16E-4)

PCI-6040E Analog Input block

Library

Simulink Real-Time Library for National Instruments

Scaling Input to Output

I/O Module Input	Block Output Data Type	Scaling
Volts	Double	1

Block Parameters

Channel vector

Enter numbers between 1 and 16. This driver allows you to enter channel numbers in arbitrary order.

For example, to use the first, second, and fifth channels, enter

```
[1,2,5]
```

Number the channels beginning with 1 even if this 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
-2 to +2	-2
-1 to +1	-1
-0.5 to +0.5	-0.5
-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 volts to +10 volts and the second and fifth channels are 0 volts 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.

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.

Coupling	Coupling Code	Description
NRSE	1	Analog input line connected to the positive input of the PGIA. Analog input sense (AISENSE) connected to the negative input of the PGIA. See the board manual.
DIFF	2	First analog input line connected to the positive input of the PGIA. Second analog input line connected to the negative input of the PGIA. See the board manual.

For example, if the first and second channels are single input and the fifth channel is a differential input, enter

```
[0,0,2]
```

The driver selects a second differential input 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

Enter the base sample time or a multiple of the base sample time (`-1` means sample time is inherited).

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

External Websites

www.ni.com

National Instruments PCI-6040E Analog Output (D/A) (formerly PCI-MIO-16E-4)

PCI-6040E Analog Output block

Library

Simulink Real-Time Library for National Instruments

Scaling Input to Output

I/O Module 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 arbitrary 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 this board manufacturer starts numbering the channels with 0.

Range vector

Enter a range code for each of the channels in the channel vector. The range vector must be the same length as the channel vector. This board allows the range of each channel to be different.

The following table is a list of the ranges for this driver and the corresponding range codes.

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

For example, if the first channel is -10 volts 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 replicated over the channel vector. If you specify a value of `1`, the corresponding channel is reset to the value specified in the initial value vector. If you specify a value of `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 replicated as the initial value over the channel vector. 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 (`-1` means sample time is inherited).

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

External Websites

www.ni.com

National Instruments PCI-6040E Digital Input (formerly PCI-MIO-16E-4)

PCI-6040E Digital Input block

Library

Simulink Real-Time Library for National Instruments

Scaling Input to Output

I/O Module 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 arbitrary 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 this board manufacturer starts numbering the lines with 0.

Sample time

Enter the base sample time or a multiple of the base sample time (`-1` means sample time is inherited).

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

External Websites

www.ni.com

National Instruments PCI-6040E Digital Output (formerly PCI-MIO-16E-4)

PCI-6040E Digital Output block

Library

Simulink Real-Time Library for National Instruments

Scaling of Input to Output

I/O Module 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 arbitrary 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 this 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 replicated over the channel vector. If you specify a value of `1`, the corresponding channel is reset to the value specified in the initial value vector.

If you specify a value of `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 replicated as the initial value over the channel vector. 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 (`-1` means sample time is inherited).

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

External Websites

www.ni.com

National Instruments PCI-6040E Pulse Generation (formerly PCI-MIO-16E-4)

PCI-6040E Pulse Generation block

Library

Simulink Real-Time Library for National Instruments

Note

The input to the block is a two-element vector. The first element is the number of counts (of the 20 MHz source clock) that the counter maintains at a low level. 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

Enter the base sample time or a multiple of the base sample time (`-1` means sample time is inherited).

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

External Websites

www.ni.com

National Instruments PCI-6040E Pulse Width/Period Measurement (formerly PCI-MIO-16E-4)

PCI-6040E Pulse Width/Period Measurement block

Library

Simulink Real-Time 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 can experience false triggering. Use signal voltage levels of 0–5 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** interact. 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** interact. 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

Measurement Objective	Trigger Mode	Polarity	Output
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

Enter the base sample time or a multiple of the base sample time (`-1` means sample time is inherited).

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

External Websites

www.ni.com

National Instruments PCI/PXI-6052E

Support for National Instruments PCI/PXI-6052E board

Board

National Instruments PCI/PXI-6052E

General Description

The PXI/PCI-6052E provides:

- 16 single or 8 differential analog input channels (16-bit) with a maximum sample rate of 333 kHz
- 2 analog output channels (16-bit)
- 8 digital input and output lines
- 2 counters/timers (24-bit) with a maximum source clock rate of 20 MHz

The Simulink Real-Time 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

See Also

External Websites

www.ni.com

National Instruments PXI/PCI-6052E Analog Input (A/D)

PXI/PCI-6052E Analog Input block

Library

Simulink Real-Time Library for National Instruments

Scaling of Input to Output

I/O Module Input	Block Output Data Type	Scaling
Volts	Double	1

Block Parameters

Channel vector

Enter numbers between 1 and 16. This driver allows you to enter channel numbers in arbitrary order.

For example, to use the first, second, and fifth channels, enter

```
[1,2,5]
```

Number the channels beginning with 1 even if this 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
-2 to +2	-2
-1 to +1	-1
-0.5 to +0.5	-0.5
-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 volts to +10 volts and the second and fifth channels are 0 volts 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.

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.

Coupling	Coupling Code	Description
NRSE	1	Analog input line connected to the positive input of the PGIA. Analog input sense (AISENSE) connected to the negative input of the PGIA. See the board manual.
DIFF	2	First analog input line connected to the positive input of the PGIA. Second analog input line connected to the negative input of the PGIA. See the board manual.

For example, if the first and second channels are single input and the fifth channel is a differential input, enter

```
[0,0,2]
```

The driver selects a second differential input 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

Enter the base sample time or a multiple of the base sample time (`-1` means sample time is inherited).

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

External Websites

www.ni.com

National Instruments PXI/PCI-6052E Analog Output (D/A)

PXI/PCI-6052E Analog Output block

Library

Simulink Real-Time Library for National Instruments

Scaling of Input to Output

I/O Module 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 arbitrary 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 this board manufacturer starts numbering the channels with 0.

Range vector

Enter a range code for each of the channels in the channel vector. The range vector must be the same length as the channel vector. This board allows the range of each channel to be different.

The following table is a list of the ranges for this driver and the corresponding range codes.

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

For example, if the first channel is -10 volts 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 replicated over the channel vector. If you specify a value of `1`, the corresponding channel is reset to the value specified in the initial value vector. If you specify a value of `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 replicated as the initial value over the channel vector. 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 (`-1` means sample time is inherited).

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

External Websites

www.ni.com

National Instruments PXI/PCI-6052E Digital Input

PXI/PCI-6052E Digital Input block

Library

Simulink Real-Time Library for National Instruments

Scaling of Input to Output

I/O Module 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 arbitrary 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 this board manufacturer starts numbering the lines with 0.

Sample time

Enter the base sample time or a multiple of the base sample time (`-1` means sample time is inherited).

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

External Websites

www.ni.com

National Instruments PXI/PCI-6052E Digital Output

PXI/PCI-6052E Digital Output block

Library

Simulink Real-Time Library for National Instruments

Scaling of Input to Output

I/O Module 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 arbitrary 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 this 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 replicated over the channel vector. If you specify a value of `1`, the corresponding channel is reset to the value specified in the initial value vector. If you specify a value of `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 replicated as the initial value over the channel vector. 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 (`-1` means sample time is inherited).

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

External Websites

www.ni.com

National Instruments PXI/PCI-6052E Pulse Generation

PXI/PCI-6052E Pulse Generation

Library

Simulink Real-Time Library for National Instruments

Note

The input to the block is a two-element vector. The first element is the number of counts (of the 20 MHz source clock) that the counter maintains at a low level. 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

Enter the base sample time or a multiple of the base sample time (`-1` means sample time is inherited).

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

External Websites

www.ni.com

National Instruments PXI/PCI-6052E Pulse Width/ Period Measurement

PXI/PCI-6052E Pulse Width/Period Measurement

Library

Simulink Real-Time 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 can experience false triggering. Use signal voltage levels of 0–5 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** interact. 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** interact. 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

Measurement Objective	Trigger Mode	Polarity	Output
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

Enter the base sample time or a multiple of the base sample time (`-1` means sample time is inherited).

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

External Websites

www.ni.com

National Instruments PCI-6070E (formerly PCI-MIO-16E-1)

Support for National Instruments PCI-6070E board

Board

National Instruments PCI-6070E

General Description

This was formerly PCI-MIO-16E-1.

The PCI-6070E provides:

- 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
- 2 counter/timers (24-bit) with a maximum source clock rate of 20 MHz

The Simulink Real-Time 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)

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

See Also

External Websites

www.ni.com

National Instruments PCI-6070E Analog Input (A/D) (formerly PCI-MIO-16E-1)

PCI-6070E Analog Input block

Library

Simulink Real-Time Library for National Instruments

Scaling Input to Output

I/O Module Input	Block Output Data Type	Scaling
Volts	Double	1

Block Parameters

Channel vector

Enter numbers between 1 and 16. This driver allows you to enter channel numbers in arbitrary order.

For example, to use the first, second, and fifth channels, enter

```
[1,2,5]
```

Number the channels beginning with 1 even if this 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
-2 to +2	-2
-1 to +1	-1
-0.5 to +0.5	-0.5
-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 volts to +10 volts and the second and fifth channels are 0 volts 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.

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.

Coupling	Coupling Code	Description
NRSE	1	Analog input line connected to the positive input of the PGIA. Analog input sense (AISENSE) connected to the negative input of the PGIA. See the board manual.
DIFF	2	First analog input line connected to the positive input of the PGIA. Second analog input line connected to the negative input of the PGIA. See the board manual.

For example, if the first and second channels are single input and the fifth channel is a differential input, enter

```
[0,0,2]
```

The driver selects a second differential input 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

Enter the base sample time or a multiple of the base sample time (`-1` means sample time is inherited).

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

External Websites

www.ni.com

National Instruments PCI-6070E Analog Output (D/A) (formerly PCI-MIO-16E-1)

PCI-6070E Analog Output block

Library

Simulink Real-Time Library for National Instruments

Scaling Input to Output

I/O Module 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 arbitrary 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 this board manufacturer starts numbering the channels with 0.

Range vector

Enter a range code for each of the channels in the channel vector. The range vector must be the same length as the channel vector. This board allows the range of each channel to be different.

The following table is a list of the ranges for this driver and the corresponding range codes.

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

For example, if the first channel is -10 volts 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 replicated over the channel vector. If you specify a value of `1`, the corresponding channel is reset to the value specified in the initial value vector. If you specify a value of `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 replicated as the initial value over the channel vector. 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 (`-1` means sample time is inherited).

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

External Websites

www.ni.com

National Instruments PCI-6070E Digital Input (formerly PCI-MIO-16E-1)

PCI-6070E Digital Input

Library

Simulink Real-Time Library for National Instruments

Scaling Input to Output

I/O Module 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 arbitrary 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 this board manufacturer starts numbering the lines with 0.

Sample time

Enter the base sample time or a multiple of the base sample time (`-1` means sample time is inherited).

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

External Websites

www.ni.com

National Instruments PCI-6070E Digital Output (formerly PCI-MIO-16E-1)

PCI-6070E Digital Output

Library

Simulink Real-Time Library for National Instruments

Scaling Input to Output

I/O Module 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 arbitrary 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 this 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 replicated over the channel vector. If you specify a value of `1`, the corresponding channel is reset to the value specified in the initial value vector.

If you specify a value of `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 replicated as the initial value over the channel vector. 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 (`-1` means sample time is inherited).

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

External Websites

www.ni.com

National Instruments PCI-6070E Pulse Generation (formerly PCI-MIO-16E-1)

PCI-6070E Pulse Generation

Library

Simulink Real-Time Library for National Instruments

Note

The input to the block is a two-element vector. The first element is the number of counts (of the 20 MHz source clock) that the counter maintains at a low level. 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

Enter the base sample time or a multiple of the base sample time (`-1` means sample time is inherited).

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

External Websites

www.ni.com

National Instruments PCI-6070E Pulse Width/Period Measurement (formerly PCI-MIO-16E-1)

PCI-6070E Pulse Width/Period Measurement

Library

Simulink Real-Time 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 can experience false triggering. Use signal voltage levels of 0-5 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** interact. 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** interact. 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

Measurement Objective	Trigger Mode	Polarity	Output
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

Enter the base sample time or a multiple of the base sample time (`-1` means sample time is inherited).

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

External Websites

www.ni.com

National Instruments PCI-6071E

Support for National Instruments PCI-6071E board

Board

National Instruments PCI-6071E

General Description

The PCI-6071E provides:

- 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
- 2 counter/timers (24-bit) with a maximum source clock rate of 20 MHz

The Simulink Real-Time 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

See Also

External Websites

www.ni.com

National Instruments PCI-6071E Analog Input (A/D)

PCI-6071E Analog Input (A/D)

Library

Simulink Real-Time Library for National Instruments

Scaling Input to Output

I/O Module Input	Block Output Data Type	Scaling
Volts	Double	1

Block Parameters

Channel vector

Enter numbers between 1 and 64. This driver allows you to enter channel numbers in arbitrary order.

For example, to use the first, second, and fifth channels, enter

```
[1,2,5]
```

Number the channels beginning with 1 even if this 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

Input Range (V)	Range Code
-5 to +5	-5
-2 to +2	-2
-1 to +1	-1
-0.5 to +0.5	-0.5
-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 volts to +10 volts and the second and fifth channels are 0 volts 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.

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.

Coupling	Coupling Code	Description
NRSE	1	Analog input line connected to the positive input of the PGIA. Analog input sense (AISENSE) connected to the negative input of the PGIA. See the board manual.
DIFF	2	First analog input line connected to the positive input of the PGIA. Second analog input line connected to the negative input of the PGIA. See the board manual.

For example, if the first and second channels are single input and the fifth channel is a differential input, enter

```
[0,0,2]
```

The driver selects a second differential input 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

Enter the base sample time or a multiple of the base sample time (`-1` means sample time is inherited).

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

External Websites

www.ni.com

National Instruments PCI-6071E Analog Output (D/A)

PCI-6071E Analog Output block

Library

Simulink Real-Time Library for National Instruments

Scaling Input to Output

I/O Module 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 arbitrary 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 this board manufacturer starts numbering the channels with 0.

Range vector

Enter a range code for each of the channels in the channel vector. The range vector must be the same length as the channel vector. This board allows the range of each channel to be different.

The following table is a list of the ranges for this driver and the corresponding range codes.

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

For example, if the first channel is -10 volts 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 replicated over the channel vector. If you specify a value of `1`, the corresponding channel is reset to the value specified in the initial value vector. If you specify a value of `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 replicated as the initial value over the channel vector. 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 (`-1` means sample time is inherited).

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

External Websites

www.ni.com

National Instruments PCI-6071E Digital Input

PCI-6071E Digital Input block

Library

Simulink Real-Time Library for National Instruments

Scaling Input to Output

I/O Module 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 arbitrary 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 this board manufacturer starts numbering the lines with 0.

Sample time

Enter the base sample time or a multiple of the base sample time (`-1` means sample time is inherited).

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

External Websites

www.ni.com

National Instruments PCI-6071E Digital Output

PCI-6071E Digital Output block

Library

Simulink Real-Time Library for National Instruments

Scaling Input to Output

I/O Module 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 arbitrary 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 this 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 replicated over the channel vector. If you specify a value of `1`, the corresponding channel is reset to the value specified in the initial value vector. If you specify a value of `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 replicated as the initial value over the channel vector. 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 (`-1` means sample time is inherited).

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

External Websites

www.ni.com

National Instruments PCI-6071E Pulse Generation

PCI-6071E Pulse Generation block

Library

Simulink Real-Time Library for National Instruments

Note

The input to the block is a two-element vector. The first element is the number of counts (of the 20 MHz source clock) that the counter maintains at a low level. 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

Enter the base sample time or a multiple of the base sample time (`-1` means sample time is inherited).

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

External Websites

www.ni.com

National Instruments PCI-6071E Pulse Width/Period Measurement

PCI-6071E Pulse Width/Period Measurement block

Library

Simulink Real-Time 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 can experience false triggering. Use signal voltage levels of 0-5 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** interact. 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** interact. 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

Measurement Objective	Trigger Mode	Polarity	Output
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

Enter the base sample time or a multiple of the base sample time (`-1` means sample time is inherited).

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

External Websites

www.ni.com

National Instruments PCI-6221

Support for PCI-6221 and PXI-6221 boards

Board

National Instruments PCI-6221, PXI-6221

General Description

The 6221 M-series boards are data acquisition I/O boards that provide:

- Up to 16 single-ended or 8 differential 16-bit analog inputs
- Two 16-bit analog outputs
- 24 digital I/O lines
- Two 32-bit counters

The Simulink Real-Time block library supports the 6221 boards 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, PXI-6221

Manufacturer	National Instruments
Bus type	PCI, PXI (both use PCI version of blocks)
Multiple block instance support	A/D: No, D/A: No, Digital I/O: Yes
Multiple board support	Yes

See Also

External Websites

www.ni.com

National Instruments PCI-6221 Analog Input

Analog Input block for PCI-6221 and PXI-6221 boards

Library

Simulink Real-Time Library for National Instruments

Scaling of Input to Output

I/O Module Input	Block Output Data Type	Scaling
Volts	Double	1

Block Parameters

Channel vector

Enter numbers from 1 through 16. This driver allows you to enter channel numbers in arbitrary order.

For example, to use the first, second, and fifth channels, enter

```
[1,2,5]
```

Number the lines beginning with 1 even if this 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 replicate over the channel vector, 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.

Input Range (V)	Range Code
-10 to +10	10
-5 to +5	5
-1 to +1	1
-0.2 to +0.2	0.2

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]
```

Input coupling vector

Enter a coupling code for each of the channels in the channel vector. You can replicate one value over the channel vector, 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

```
[0,0,2]
```

National Instruments boards use a zero-based number scheme for differential mode channel pairs. Simulink Real-Time drivers use a one-based number scheme that adds a 1 to the I/O module number, as shown in the following table:

Zero-Based Channel Number		One-Based Channel Number	
+	-	+	-
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 I/O module uses a second input 8 channels higher than the first channel as the negative input of the pair. In the example above, the I/O module would use the 13th channel as a differential input with the fifth channel.

If the module 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

Enter the base sample time or a multiple of the base sample time (-1 means sample time is inherited).

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

External Websites

www.ni.com

National Instruments PCI-6221 Analog Output

Analog Output block for PCI-6221 and PXI-6221 boards

Library

Simulink Real-Time Library for National Instruments

Scaling of Output to Input

I/O Module 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 arbitrary 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 this 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 replicated over the channel vector. If you specify a value of `1`, the corresponding channel is reset to the value specified in the initial value vector. If you specify a value of `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 replicated as the initial value over the channel vector. 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 (`-1` means sample time is inherited).

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

External Websites

www.ni.com

National Instruments PCI-6221 Digital Input

Digital Input block for PCI-6221 and PXI-6221 boards

Library

Simulink Real-Time Library for National Instruments

Note

Models can have multiple digital input blocks for one physical board. In this case, the Simulink Real-Time 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.

A digital input block can share channels with a digital output block. In this configuration, the input block reads the last value written to the digital output block on the shared channels.

Scaling of Input to Output

I/O Module 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 arbitrary 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 if this board manufacturer starts numbering the lines with 0.

Sample time

Enter the base sample time or a multiple of the base sample time (`-1` means sample time is inherited).

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

External Websites

www.ni.com

National Instruments PCI-6221 Digital Output

Digital Output block for PCI-6221 and PXI-6221 boards

Library

Simulink Real-Time Library for National Instruments

Note

Models can have multiple digital output blocks for one physical board. In this case, the Simulink Real-Time 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.

A digital input block can share channels with a digital output block. In this configuration, the input block reads the last value written to the digital output block on the shared channels.

The block compares the value written to each port with 0.5.

Value written	Block writes digital output with a value of...
>= 0.5	1
<0.5	0

Scaling of Output to Input

I/O Module 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 arbitrary 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 this 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 replicated over the channel vector. If you specify a value of `1`, the corresponding channel is reset to the value specified in the initial value vector. If you specify a value of `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 replicated as the initial value over the channel vector. 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 (`-1` means sample time is inherited).

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

External Websites

www.ni.com

National Instruments PCI-6221 Incremental Encoder

Incremental Encoder block for PCI-6221 and PXI-6221 boards

Library

Simulink Real-Time Library for National Instruments

Scaling Input to Output

I/O Module 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

Mode	Description
`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`

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 the input is not filtered.

- `None`
- `Minimum pulse width 125 nanoseconds`
- `Minimum pulse width 6.25 microseconds`
- `Minimum pulse width 1.25 milliseconds`

Sample time

Enter the base sample time or a multiple of the base sample time (`-1` means sample time is inherited).

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

External Websites

www.ni.com

National Instruments PCI-6221 PFI Digital Input

PFI Digital Input block for PCI-6221 and PXI-6221 boards

Library

Simulink Real-Time Library for National Instruments

Description

A model can have more than one PFI digital input block for a 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

I/O Module Input	Block Output Data Type	Scaling
TTL	Double	TTL low ≥ 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 the bits you want to monitor. For example, to read bits PFI 0, PFI 5, and PFI 7, enter

```
[0, 5, 7]
```

Sample time

Enter the base sample time or a multiple of the base sample time (`-1` means sample time is inherited).

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

External Websites

www.ni.com

National Instruments PCI-6221 PFI Digital Output

PFI Digital Output block for PCI-6221 and PXI-6221 boards

Library

Simulink Real-Time Library for National Instruments

Description

A model can have more than one PFI digital input block for a 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

I/O Module 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 the 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 of the channel at model termination. Enter a scalar or a vector that is the same length as the channel vector. If you specify a scalar value, that setting is replicated over the channel vector. If you specify a value of `1`, the corresponding channel is reset to the value specified in the initial value vector. If you specify a value of `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 block outputs. Enter a scalar or a vector of values, one for each output. If you enter a scalar, the driver replicates that value over the output vector.

The block applies this value as follows:

- The model uses this value when you first download the model to the target computer.
- 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

Enter the base sample time or a multiple of the base sample time (`-1` means sample time is inherited).

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

External Websites

www.ni.com

National Instruments PCI-6221 PWM Generate

PWM Generate block for PCI-6221 and PXI-6221 boards

Library

Simulink Real-Time Library for National Instruments

Description

Use this block to generate variable frequency and pulse width output.

The 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 number of counts the output is high 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 low until the first time the Simulink engine calls `mdlOutputs`.

Initial low count

Enter the number of counts the output is low 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

Enter the base sample time or a multiple of the base sample time (`-1` means sample time is inherited).

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

External Websites

www.ni.com

National Instruments PCI-6221 PWM Measure

PWM Measure block for PCI-6221 and PXI-6221 boards

Library

Simulink Real-Time 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 the previous value.

This block returns 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 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 the input is not filtered

- `None`
- `Minimum pulse width 125 nanoseconds`
- `Minimum pulse width 6.25 microseconds`
- `Minimum pulse width 1.25 milliseconds`

Sample time

Enter the base sample time or a multiple of the base sample time (`-1` means sample time is inherited).

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

External Websites

www.ni.com

National Instruments PCI-6221/37

Support for PCI-6221/37 and PXI-6221/37 boards

Board

National Instruments PCI-6221/37, PXI-6221/37

General Description

The 6221/37 M-series boards are data acquisition I/O boards that provide:

- Up to 16 single-ended or 8 differential 16-bit analog inputs
- Two 16-bit analog outputs
- 10 digital I/O lines
- Two 32-bit counters

The Simulink Real-Time block library supports the 6221/37 boards 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, PXI-6221/37

Manufacturer	National Instruments
Bus type	PCI, PXI (both use PCI version of blocks)
Multiple block instance support	A/D: No, D/A: No, Digital I/O: Yes
Multiple board support	Yes

See Also

External Websites

www.ni.com

National Instruments PCI-6221/37 Analog Input

Analog Input block for PCI-6221/37 and PXI-6221/37 boards

Library

Simulink Real-Time Library for National Instruments

Scaling of Input to Output

I/O Module Input	Block Output Data Type	Scaling
Volts	Double	1

Block Parameters

Channel vector

Enter numbers from 1 through 16. This driver allows you to enter channel numbers in arbitrary order.

For example, to use the first, second, and fifth channels, enter

```
[1,2,5]
```

Number the lines beginning with 1 even if this 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 replicate over the channel vector, 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.

Input Range (V)	Range Code
-10 to +10	10
-5 to +5	5
-1 to +1	1
-0.2 to +0.2	0.2

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]
```

Input coupling vector

Enter a coupling code for each of the channels in the channel vector. You can replicate one value over the channel vector, 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

```
[0,0,2]
```

National Instruments boards use a zero-based number scheme for differential mode channel pairs. Simulink Real-Time drivers use a one-based number scheme that adds a 1 to the I/O module number, as shown in the following table:

Zero-Based Channel Number		One-Based Channel Number	
+	-	+	-
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 I/O module uses a second input 8 channels higher than the first channel as the negative input of the pair. In the example above, the I/O module 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

Enter the base sample time or a multiple of the base sample time (-1 means sample time is inherited).

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

External Websites

www.ni.com

National Instruments PCI-6221/37 Analog Output

Analog Output block for PCI-6221/37 and PXI-6221/37 boards

Library

Simulink Real-Time Library for National Instruments

Scaling of Output to Input

I/O Module 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 arbitrary 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 this 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 replicated over the channel vector. If you specify a value of `1`, the corresponding channel is reset to the value specified in the initial value vector. If you specify a value of `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 replicated as the initial value over the channel vector. 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 (`-1` means sample time is inherited).

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

External Websites

www.ni.com

National Instruments PCI-6221/37 Digital Input

Digital Input block for PCI-6221/37 and PXI-6221/37 boards

Library

Simulink Real-Time Library for National Instruments

Note

Models can have multiple digital input blocks for one physical board. In this case, the Simulink Real-Time 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.

A digital input block can share channels with a digital output block. In this configuration, the input block reads the last value written to the digital output block on the shared channels.

Scaling of Input to Output

I/O Module 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 arbitrary 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 if this board manufacturer starts numbering the lines with 0.

Sample time

Enter the base sample time or a multiple of the base sample time (`-1` means sample time is inherited).

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

External Websites

www.ni.com

National Instruments PCI-6221/37 Digital Output

Digital Output block for PCI-6221/37 and PXI-6221/37 boards

Library

Simulink Real-Time Library for National Instruments

Note

Models can have multiple digital output blocks for one physical board. In this case, the Simulink Real-Time 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.

A digital input block can share channels with a digital output block. In this configuration, the input block reads the last value written to the digital output block on the shared channels.

The block compares the value written to each port with 0.5.

Value written	Block writes digital output with a value of...
>= 0.5	1
<0.5	0

Scaling of Output to Input

I/O Module Output	Block Input Data Type	Scaling
TTL	Double	< 0.5 = TTL low ≥ 0.5 = 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 arbitrary 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 if this 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 replicated over the channel vector. If you specify a value of `1`, the corresponding channel is reset to the value specified in the initial value vector. If you specify a value of `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 replicated as the initial value over the channel vector. 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 (`-1` means sample time is inherited).

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

External Websites

www.ni.com

National Instruments PCI-6221/37 Incremental Encoder

Incremental Encoder block for PCI-6221/37 and PXI-6221/37 boards

Library

Simulink Real-Time Library for National Instruments

Scaling Input to Output

I/O Module 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 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

Mode	Description
`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`

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 the input is not filtered.

- `None`
- `Minimum pulse width 125 nanoseconds`
- `Minimum pulse width 6.25 microseconds`
- `Minimum pulse width 1.25 milliseconds`

Sample time

Enter the base sample time or a multiple of the base sample time (`-1` means sample time is inherited).

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

External Websites

www.ni.com

National Instruments PCI-6221/37 PFI Digital Input

PFI Digital Input block for PCI-6221/37 and PXI-6221/37 boards

Library

Simulink Real-Time Library for National Instruments

Description

A model can have more than one PFI digital input block for a 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:

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

I/O Module Input	Block Output Data Type	Scaling
TTL	Double	TTL low ≥ 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 the bits you want to monitor. For example, to read bits PFI 0, PFI 5, and PFI 7, enter

```
[0, 5, 7]
```

Sample time

Enter the base sample time or a multiple of the base sample time (`-1` means sample time is inherited).

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

External Websites

www.ni.com

National Instruments PCI-6221/37 PFI Digital Output

PFI Digital Output block for PCI-6221/37 and PXI-6221/37 boards

Library

Simulink Real-Time Library for National Instruments

Description

A model can have more than one PFI digital input block for a 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

I/O Module 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 the bits you want to control. For example, to write bits PFI 1, PFI 3, and PFI 7, enter

```
[1, 3, 7]
```

Reset vector

The reset vector controls the behavior of the channel at model termination. Enter a scalar or a vector that is the same length as the channel vector. If you specify a scalar value, that setting is replicated over the channel vector. If you specify a value of `1`, the corresponding channel is reset to the value specified in the initial value vector. If you specify a value of `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 block outputs. Enter a scalar or a vector of values, one for each output. If you enter a scalar, the driver replicates that value over the output vector.

The block applies this value as follows:

- The model uses this value when you first download the model to the target computer.
- 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

Enter the base sample time or a multiple of the base sample time (`-1` means sample time is inherited).

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

External Websites

www.ni.com

National Instruments PCI-6221/37 PWM Generate

PWM Generate block for PCI-6221/37 and PXI-6221/37 boards

Library

Simulink Real-Time Library for National Instruments

Description

Use this block to generate variable frequency and pulse width output.

The 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 number of counts the output is high 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 low until the first time the Simulink engine calls `mdlOutputs`.

Initial low count

Enter the number of counts the output is low 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

Enter the base sample time or a multiple of the base sample time (`-1` means sample time is inherited).

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

External Websites

www.ni.com

National Instruments PCI-6221/37 PWM Measure

PWM Measure block for PCI-6221/37 and PXI-6221/37 boards

Library

Simulink Real-Time 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 the previous value.

This block returns 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 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 the input is not filtered

- `None`
- `Minimum pulse width 125 nanoseconds`
- `Minimum pulse width 6.25 microseconds`
- `Minimum pulse width 1.25 milliseconds`

Sample time

Enter the base sample time or a multiple of the base sample time (`-1` means sample time is inherited).

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

External Websites

www.ni.com

National Instruments PCI-6225

Support for PCI-6225 and PXI-6225 boards

Board

National Instruments PCI-6225, PXI-6225

General Description

The 6225 M-series boards are data acquisition I/O boards that provide:

- 2 16-bit analog outputs
- 24 digital I/O lines
- 2 32-bit counters

The Simulink Real-Time block library supports the 6225 boards 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, PXI-6225
Manufacturer	National Instruments

Bus type	PCI, PXI (both use PCI version of blocks)
Multiple block instance support	• A/D One block per board • D/A One block per board • Digital I/O Multiple blocks per board • Incremental Encoder One block per counter per board • PFI Digital Input/Output Multiple blocks per board • PWM Generate and Measure One block per counter per board
Multiple board support	Yes

See Also

External Websites

www.ni.com

National Instruments PCI-6225 Analog Input

Analog Input block for PCI-6225 and PXI-6225 boards

Library

Simulink Real-Time Library for National Instruments

Scaling of Input to Output

I/O Module Input	Block Output Data Type	Scaling
Volts	Double	1

Block Parameters

Channel vector

Enter numbers between 1 and 80. This driver allows you to enter channel numbers in arbitrary order.

For example, to use the first, second, and fifth channels, enter

```
[1,2,5]
```

Number the lines beginning with 1 even if this 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 replicate over the channel vector, 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.

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

Input Range (V)	Range Code
-5 to +5	5
-1 to +1	1
-0.2 to +0.2	0.2

For example, if the first channel is -10 volts to +10 volts and the second and fifth channels are -1 to +1 volts, enter

```
[10,1,1]
```

Input coupling vector

Enter a coupling code for each of the channels in the channel vector. You can replicate one value over the channel vector, 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

```
[0,0,2]
```

National Instruments boards use a zero-based number scheme for differential mode channel pairs. Simulink Real-Time drivers use a one-based number scheme that adds a `1` to the I/O module number, as shown in the following table:

Zero-Based Channel Number		One-Based Channel Number	
+	-	+	-
0–7	8–15	1–8	9–16
16–23	24–31	17–24	25–32
32–39	40–47	33–40	41–48
48–55	56–63	49–56	57–64
64–71	72–79	65–72	73–80

The I/O module uses a second input 8 channels higher than the first channel as the negative input of the pair. In the example above, the module 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. 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

Enter the base sample time or a multiple of the base sample time (`-1` means sample time is inherited).

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

External Websites

www.ni.com

National Instruments PCI-6225 Analog Output

Analog Output block for PCI-6225 and PXI-6225 boards

Library

Simulink Real-Time Library for National Instruments

Scaling of Output to Input

I/O Module Output	Block Input Data Type	Scaling
Volts	Double	1

Block Parameters

Channel vector

Enter 1 to 2,. This driver allows the selection of individual D/A channels in arbitrary 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 this 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 replicated over the channel vector. If you specify a value of `1`, the corresponding channel is reset to the value specified in the initial value vector. If you specify a value of `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 replicated as the initial value over the channel vector. 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 (`-1` means sample time is inherited).

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

External Websites

www.ni.com

National Instruments PCI-6225 Digital Input

Digital Input block for PCI-6225 and PXI-6225 boards

Library

Simulink Real-Time Library for National Instruments

Note

Models can have multiple digital input blocks for one physical board. In this case, the Simulink Real-Time 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.

A digital input block can share channels with a digital output block. In this configuration, the input block reads the last value written to the digital output block on the shared channels.

Scaling of Input to Output

I/O Module 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 arbitrary 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 if this board manufacturer starts numbering the lines with 0.

Sample time

Enter the base sample time or a multiple of the base sample time (`-1` means sample time is inherited).

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

External Websites

www.ni.com

National Instruments PCI-6225 Digital Output

Digital Output block for PCI-6225 and PXI-6225 boards

Library

Simulink Real-Time Library for National Instruments

Note

Models can have multiple digital output blocks for one physical board. In this case, the Simulink Real-Time 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.

A digital input block can share channels with a digital output block. In this configuration, the input block reads the last value written to the digital output block on the shared channels.

The block compares the value written to each port with 0.5.

Value written	Block writes digital output with a value of...
>= 0.5	1
<0.5	0

Scaling of Output to Input

I/O Module 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 arbitrary 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 if this 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 replicated over the channel vector. If you specify a value of `1`, the corresponding channel is reset to the value specified in the initial value vector. If you specify a value of `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 replicated as the initial value over the channel vector. 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 (`-1` means sample time is inherited).

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

External Websites

www.ni.com

National Instruments PCI-6225 Incremental Encoder

Incremental Encoder block for PCI-6225 and PXI-6225 boards

Library

Simulink Real-Time Library for National Instruments

Scaling Input to Output

I/O Module 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

Mode	Description
`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`

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 the input is not filtered.

- `None`
- `Minimum pulse width 125 nanoseconds`
- `Minimum pulse width 6.25 microseconds`
- `Minimum pulse width 1.25 milliseconds`

Sample time

Enter the base sample time or a multiple of the base sample time (`-1` means sample time is inherited).

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

External Websites

www.ni.com

National Instruments PCI-6225 PFI Digital Input

PFI Digital Input block for PCI-6225 and PXI-6225 boards

Library

Simulink Real-Time Library for National Instruments

Description

A model can have more than one PFI digital input block for a 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

I/O Module Input	Block Output Data Type	Scaling
TTL	Double	TTL low ≥ 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 the bits you want to monitor. For example, to read bits PFI 0, PFI 5, and PFI 7, enter

```
[0, 5, 7]
```

Sample time

Enter the base sample time or a multiple of the base sample time (`-1` means sample time is inherited).

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

External Websites

www.ni.com

National Instruments PCI-6225 PFI Digital Output

PFI Digital Output block for PCI-6225 and PXI-6225 boards

Library

Simulink Real-Time Library for National Instruments

Description

A model can have more than one PFI digital input block for a 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

I/O Module 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 the 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 of the channel at model termination. Enter a scalar or a vector that is the same length as the channel vector. If you specify a scalar value, that setting is replicated over the channel vector. If you specify a value of `1`, the corresponding channel is reset to the value specified in the initial value vector. If you specify a value of `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 block outputs. Enter a scalar or a vector of values, one for each output. If you enter a scalar, the driver replicates that value over the output vector.

The block applies this value as follows:

- The model uses this value when you first download the model to the target computer.
- 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

Enter the base sample time or a multiple of the base sample time (`-1` means sample time is inherited).

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

External Websites

www.ni.com

National Instruments PCI-6225 PWM Generate

PWM Generate block for PCI-6225 and PXI-6225 boards

Library

Simulink Real-Time Library for National Instruments

Description

Use this block to generate variable frequency and pulse width output.

The 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 number of counts the output is high 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 low until the first time the Simulink engine calls `mdlOutputs`.

Initial low count

Enter the number of counts the output is low 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

Enter the base sample time or a multiple of the base sample time (`-1` means sample time is inherited).

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

External Websites

www.ni.com

National Instruments PCI-6225 PWM Measure

PWM Measure block for PCI-6225 and PXI-6225 boards

Library

Simulink Real-Time 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 the previous value.

This block returns 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 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 the input is not filtered

- `None`
- `Minimum pulse width 125 nanoseconds`
- `Minimum pulse width 6.25 microseconds`
- `Minimum pulse width 1.25 milliseconds`

Sample time

Enter the base sample time or a multiple of the base sample time (`-1` means sample time is inherited).

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

External Websites

www.ni.com

National Instruments PCI-6229

Support for PCI-6229 and PXI-6229 boards

Board

National Instruments PCI-6229, PXI-6229

General Description

The 6229 M-series boards are data acquisition I/O boards that provide:

- 32 single-ended or 16 differential 16-bit analog inputs
- Four 16-bit analog outputs
- 48 digital I/O lines
- Two 32-bit counters

The Simulink Real-Time block library supports the 6229 boards 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, PXI-6229

Manufacturer	National Instruments
Bus type	PCI, PXI (both use PCI version of blocks)
Multiple block instance support	A/D: No, D/A: No, Digital I/O: Yes
Multiple board support	Yes

See Also

External Websites

www.ni.com

National Instruments PCI-6229 Analog Input

Analog Input block for PCI-6229 and PXI-6229 boards

Library

Simulink Real-Time Library for National Instruments

Scaling of Input to Output

I/O Module Input	Block Output Data Type	Scaling
Volts	Double	1

Block Parameters

Channel vector

Enter numbers between 1 and 32. This driver allows you to enter channel numbers in arbitrary order.

For example, to use the first, second, and fifth channels, enter

```
[1,2,5]
```

Number the lines beginning with 1 even if this 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 replicate over the channel vector, 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.

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

Input Range (V)	Range Code
-5 to +5	5
-1 to +1	1
-0.2 to +0.2	0.2

For example, if the first channel is -10 volts to +10 volts and the second and fifth channels are -1 to +1 volts, enter

```
[10,1,1]
```

Input coupling vector

Enter a coupling code for each of the channels in the channel vector. You can replicate one value over the channel vector, 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

```
[0,0,2]
```

National Instruments boards use a zero-based number scheme for differential mode channel pairs. Simulink Real-Time drivers use a one-based number scheme that adds a `1` to the I/O module number, as shown in the following table:

Zero-Based Channel Number		One-Based Channel Number	
+	-	+	-
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 I/O module uses a second input 8 channels higher than the first channel as the negative input of the pair. In the example above, the module would use the 13 as a differential input with the fifth channel.

If the module 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

Enter the base sample time or a multiple of the base sample time (`-1` means sample time is inherited).

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

External Websites

www.ni.com

National Instruments PCI-6229 Analog Output

Analog Output block for PCI-6229 and PXI-6229 boards

Library

Simulink Real-Time Library for National Instruments

Scaling of Output to Input

I/O Module 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 arbitrary order. The number of elements defines the number of D/A channels used. For example, to use the four analog output channels, enter

```
[1,2, 3, 4]
```

Number the channels begin with 1 even if this 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 replicated over the channel vector. If you specify a value of `1`, the corresponding channel is reset to the value specified in the initial value vector. If you specify a value of `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 replicated as the initial value over the channel vector. 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 (`-1` means sample time is inherited).

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

External Websites

www.ni.com

National Instruments PCI-6229 Digital Input

Digital Input block for PCI-6229 and PXI-6229 boards

Library

Simulink Real-Time Library for National Instruments

Note

Models can have multiple digital input blocks for one physical board. In this case, the Simulink Real-Time 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.

A digital input block can share channels with a digital output block. In this configuration, the input block reads the last value written to the digital output block on the shared channels.

Scaling of Input to Output

I/O Module 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 arbitrary 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 if this board manufacturer starts numbering the lines with 0.

Sample time

Enter the base sample time or a multiple of the base sample time (`-1` means sample time is inherited).

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

External Websites

www.ni.com

National Instruments PCI-6229 Digital Output

PCI-6229 and PXI-6229 Digital Output block

Library

Simulink Real-Time Library for National Instruments

Note

Models can have multiple digital output blocks for one physical board. In this case, the Simulink Real-Time 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.

A digital input block can share channels with a digital output block. In this configuration, the input block reads the last value written to the digital output block on the shared channels.

The block compares the value written to each port with 0.5.

Value written	Block writes digital output with a value of...
>= 0.5	1
<0.5	0

Scaling of Output to Input

I/O Module 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 32 to select the digital output lines used with this port. This driver allows the selection of individual digital output lines in arbitrary 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 if this 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 replicated over the channel vector. If you specify a value of `1`, the corresponding channel is reset to the value specified in the initial value vector. If you specify a value of `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 replicated as the initial value over the channel vector. 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 (`-1` means sample time is inherited).

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

External Websites

www.ni.com

National Instruments PCI-6229 Incremental Encoder

Incremental Encoder block for PCI-6229 and PXI-6229 boards

Library

Simulink Real-Time Library for National Instruments

Scaling Input to Output

I/O Module 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

Mode	Description
`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`

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 the input is not filtered.

- `None`
- `Minimum pulse width 125 nanoseconds`
- `Minimum pulse width 6.25 microseconds`
- `Minimum pulse width 1.25 milliseconds`

Sample time

Enter the base sample time or a multiple of the base sample time (`-1` means sample time is inherited).

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

External Websites

www.ni.com

National Instruments PCI-6229 PFI Digital Input

PCI-6229 and PXI-6229 PFI Digital Input block

Library

Simulink Real-Time Library for National Instruments

Description

A model can have more than one PFI digital input block for a 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

I/O Module Input	Block Output Data Type	Scaling
TTL	Double	TTL low ≥ 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 the bits you want to monitor. For example, to read bits PFI 0, PFI 5, and PFI 7, enter

```
[0, 5, 7]
```

Sample time

Enter the base sample time or a multiple of the base sample time (`-1` means sample time is inherited).

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

External Websites

www.ni.com

National Instruments PCI-6229 PFI Digital Output

PFI Digital Output block for PCI-6229 and PXI-6229 boards

Library

Simulink Real-Time Library for National Instruments

Description

A model can have more than one PFI digital input block for a 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

I/O Module 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 the 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 of the channel at model termination. Enter a scalar or a vector that is the same length as the channel vector. If you specify a scalar value, that setting is replicated over the channel vector. If you specify a value of `1`, the corresponding channel is reset to the value specified in the initial value vector. If you specify a value of `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 block outputs. Enter a scalar or a vector of values, one for each output. If you enter a scalar, the driver replicates that value over the output vector.

The block applies this value as follows:

- The model uses this value when you first download the model to the target computer.
- 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

Enter the base sample time or a multiple of the base sample time (`-1` means sample time is inherited).

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

External Websites

www.ni.com

National Instruments PCI-6229 PWM Generate

PWM Generate block for PCI-6229 and PXI-6229 boards

Library

Simulink Real-Time Library for National Instruments

Description

Use this block to generate variable frequency and pulse width output.

The 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 number of counts the output is high 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 low until the first time the Simulink engine calls `mdlOutputs`.

Initial low count

Enter the number of counts the output is low 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

Enter the base sample time or a multiple of the base sample time (`-1` means sample time is inherited).

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

External Websites

www.ni.com

National Instruments PCI-6229 PWM Measure

PWM Measure block for PCI-6229 and PXI-6229 boards

Library

Simulink Real-Time 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 the previous value.

This block returns 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 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 the input is not filtered

- `None`
- `Minimum pulse width 125 nanoseconds`
- `Minimum pulse width 6.25 microseconds`
- `Minimum pulse width 1.25 milliseconds`

Sample time

Enter the base sample time or a multiple of the base sample time (`-1` means sample time is inherited).

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

External Websites

www.ni.com

National Instruments PCI-6251

Support for PCI-6251, PCIe-6251, PXI-6251, and PXIe-6251 boards

Board

National Instruments PCI-6251, PCIe-6251, PXI-6251, PXIe-6251

General Description

The 6251 M-series boards are high-speed data acquisition I/O boards that provide:

- Up to 16 single-ended or 8 differential 16-bit analog inputs
- Two 16-bit analog outputs
- 24 digital I/O lines
- Two 32-bit counters

The Simulink Real-Time block library supports the 6251 boards 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

Board Characteristics

Board name	PCI-6251, PCIe-6251, PXI-6251, PXIe-6251

Manufacturer	National Instruments
Bus type	PCI, PCIe, PXI, PXIe (use PCI version of blocks)
Multiple block instance support	A/D: No, D/A: No, Digital I/O: Yes
Multiple board support	Yes

See Also

External Websites

www.ni.com

National Instruments PCI-6251 Analog Input

Analog Input block for PCI-6251, PCIe-6251, PXI-6251, and PXIe-6251 boards

Library

Simulink Real-Time Library for National Instruments

Scaling of Input to Output

I/O Module Input	Block Output Data Type	Scaling
Volts	Double	1

Block Parameters

Channel vector

Enter numbers between 1 and 16. This driver allows you to enter channel numbers in arbitrary order.

For example, to use the first, second, and fifth channels, enter

```
[1,2,5]
```

Number the lines beginning with 1 even if this 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 replicate over the channel vector, 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.

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

Input Range (V)	Range Code
-2 to +2	2
-1 to +1	1
-.5 to +.5	.5
-0.2 to +0.2	0.2
-0.1 to +0.1	0.1

For example, if the first channel is -10 volts to +10 volts and the second and fifth channels are -1 to +1 volts, enter

```
[10,1,1]
```

Input coupling vector

Enter a coupling code for each of the channels in the channel vector. You can replicate one value over the channel vector, 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

```
[0,0,2]
```

National Instruments boards use a zero-based number scheme for differential mode channel pairs. Simulink Real-Time drivers use a one-based number scheme that adds a 1 to the I/O module number, as shown in the following table:

Zero-Based Channel Number		One-Based Channel Number	
+	-	+	-
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 I/O module uses a second input 8 channels higher than the first channel as the negative input of the pair. In the example above, the I/O module would use the 13th channel as a differential input with the fifth channel.

If the module 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

Enter the base sample time or a multiple of the base sample time (`-1` means sample time is inherited).

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

External Websites

www.ni.com

National Instruments PCI-6251 Analog Output

Analog Output block for PCI-6251, PCIe-6251, PXI-6251, and PXIe-6251 boards

Library

Simulink Real-Time Library for National Instruments

Scaling of Output to Input

I/O Module 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 arbitrary 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 this 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

Input Range (V)	Range Code
-5 to +5	-5
$V_{ref} \times sig$, where $-1 \le sig \le +1$	APFI0

For example, if the first channel is -10 volts 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 replicated over the channel vector. If you specify a value of `1`, the corresponding channel is reset to the value specified in the initial value vector. If you specify a value of `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 replicated as the initial value over the channel vector. 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 (`-1` means sample time is inherited).

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

External Websites

www.ni.com

National Instruments PCI-6251 Digital Input

Digital Input block for PCI-6251, PCIe-6251, PXI-6251, and PXIe-6251 boards

Library

Simulink Real-Time Library for National Instruments

Note

Models can have multiple digital input blocks for one physical board. In this case, the Simulink Real-Time 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.

A digital input block can share channels with a digital output block. In this configuration, the input block reads the last value written to the digital output block on the shared channels.

Scaling of Input to Output

I/O Module 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 arbitrary 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 if this board manufacturer starts numbering the lines with 0.

Sample time

Enter the base sample time or a multiple of the base sample time (`-1` means sample time is inherited).

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

External Websites

www.ni.com

National Instruments PCI-6251 Digital Output

Digital Output block for PCI-6251, PCIe-6251, PXI-6251, and PXIe-6251 boards

Library

Simulink Real-Time Library for National Instruments

Note

Models can have multiple digital output blocks for one physical board. In this case, the Simulink Real-Time 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.

A digital input block can share channels with a digital output block. In this configuration, the input block reads the last value written to the digital output block on the shared channels.

The block compares the value written to each port with 0.5.

Value written	Block writes digital output with a value of...
>= 0.5	1
<0.5	0

Scaling of Output to Input

I/O Module 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 arbitrary 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 this 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 replicated over the channel vector. If you specify a value of `1`, the corresponding channel is reset to the value specified in the initial value vector. If you specify a value of `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 replicated as the initial value over the channel vector. 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 (`-1` means sample time is inherited).

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

External Websites

www.ni.com

National Instruments PCI-6251 Incremental Encoder

Incremental Encoder block for PCI-6251, PCIe-6251, PXI-6251, and PXIe-6251 boards

Library

Simulink Real-Time Library for National Instruments

Scaling Input to Output

I/O Module 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

Mode	Description
`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`

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 the input is not filtered.

- `None`
- `Minimum pulse width 125 nanoseconds`
- `Minimum pulse width 6.25 microseconds`
- `Minimum pulse width 1.25 milliseconds`

Sample time

Enter the base sample time or a multiple of the base sample time (`-1` means sample time is inherited).

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

External Websites

www.ni.com

National Instruments PCI-6251 PFI Digital Input

PFI Digital Input block for PCI-6251, PCIe-6251, PXI-6251, and PXIe-6251 boards

Library

Simulink Real-Time Library for National Instruments

Description

A model can have more than one PFI digital input block for a 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

I/O Module Input	Block Output Data Type	Scaling
TTL	Double	TTL low ≥ 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 the bits you want to monitor. For example, to read bits PFI 0, PFI 5, and PFI 7, enter

```
[0, 5, 7]
```

Sample time

Enter the base sample time or a multiple of the base sample time (`-1` means sample time is inherited).

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

External Websites

www.ni.com

National Instruments PCI-6251 PFI Digital Output

PFI Digital Output block for PCI-6251, PCIe-6251, PXI-6251, and PXIe-6251 boards

Library

Simulink Real-Time Library for National Instruments

Description

A model can have more than one PFI digital input block for a 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

I/O Module 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 the 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 of the channel at model termination. Enter a scalar or a vector that is the same length as the channel vector. If you specify a scalar value, that setting is replicated over the channel vector. If you specify a value of `1`, the corresponding channel is reset to the value specified in the initial value vector. If you specify a value of `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 block outputs. Enter a scalar or a vector of values, one for each output. If you enter a scalar, the driver replicates that value over the output vector.

The block applies this value as follows:

- The model uses this value when you first download the model to the target computer.
- 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

Enter the base sample time or a multiple of the base sample time (`-1` means sample time is inherited).

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

External Websites

www.ni.com

National Instruments PCI-6251 PWM Generate

PWM Generate block for PCI-6251, PCIe-6251, PXI-6251, and PXIe-6251 boards

Library

Simulink Real-Time Library for National Instruments

Description

Use this block to generate variable frequency and pulse width output.

The 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 number of counts the output is high 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 low until the first time the Simulink engine calls `mdlOutputs`.

Initial low count

Enter the number of counts the output is low 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

Enter the base sample time or a multiple of the base sample time (`-1` means sample time is inherited).

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

External Websites

www.ni.com

National Instruments PCI-6251 PWM Measure

PWM Measure block for PCI-6251, PCIe-6251, PXI-6251, and PXIe-6251 boards

Library

Simulink Real-Time 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 the previous value.

This block returns 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 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 the input is not filtered

- `None`
- `Minimum pulse width 125 nanoseconds`
- `Minimum pulse width 6.25 microseconds`
- `Minimum pulse width 1.25 milliseconds`

Sample time

Enter the base sample time or a multiple of the base sample time (`-1` means sample time is inherited).

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

External Websites

www.ni.com

National Instruments PCI-6259

Support for PCI-6259, PCIe-6259, PXI-6259, and PXIe-6259 boards

Board

National Instruments PCI-6259, PCIe-6259, PXI-6259, PXIe-6259

General Description

The 6259 M-series boards are data acquisition I/O boards that provide:

- Up to 32 single-ended or 16 differential 16-bit analog inputs
- Four 16-bit analog outputs
- 48 digital I/O lines
- Two 32-bit counters

The Simulink Real-Time block library supports the 6259 boards 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, PCIe-6259, PXI-6259, PXIe-6259

Manufacturer	National Instruments
Bus type	PCI, PCIe, PXI, PXIe (use PCI version of blocks)
Multiple block instance support	A/D: No, D/A: No, Digital I/O: Yes
Multiple board support	Yes

See Also

External Websites

www.ni.com

National Instruments PCI-6259 Analog Input

Analog Input block for PCI-6259, PCIe-6259, PXI-6259, and PXIe-6259 boards

Library

Simulink Real-Time Library for National Instruments

Scaling of Input to Output

I/O Module Input	Block Output Data Type	Scaling
Volts	Double	1

Block Parameters

Channel vector

Enter numbers between 1 and 32. This driver allows you to enter channel numbers in arbitrary order.

For example, to use the first, second, and fifth channels, enter

```
[1,2,5]
```

Number the lines beginning with 1 even if this 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 replicate over the channel vector, 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.

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

Input Range (V)	Range Code
-5 to +5	5
-2 to +2	2
-1 to +1	1
-0.5 to +0.5	0.5
-0.2 to +0.2	0.2
-0.1 to +0.1	0.1

For example, if the first channel is -10 volts to +10 volts and the second and fifth channels are -1 to +1 volts, enter

```
[10,1,1]
```

Input coupling vector

Enter a coupling code for each of the channels in the channel vector. You can replicate one value over the channel vector, 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

```
[0,0,2]
```

National Instruments boards use a zero-based number scheme for differential mode channel pairs. Simulink Real-Time drivers use a one-based number scheme that adds a 1 to the channel number, as shown in the following table:

Zero-Based Channel Number		One-Based Channel Number	
+	-	+	-
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 I/O module uses a second input 8 channels higher than the first channel as the negative input of the pair. In the example above, the I/O module would use the 13th channel as a differential input with the fifth channel.

If the module 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

Enter the base sample time or a multiple of the base sample time (`-1` means sample time is inherited).

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

External Websites

www.ni.com

National Instruments PCI-6259 Analog Output

Analog Output block for PCI-6259, PCIe-6259, PXI-6259, and PXIe-6259 boards

Library

Simulink Real-Time Library for National Instruments

Scaling of Output to Input

I/O Module 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 arbitrary 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 if this 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

Input Range (V)	Range Code
-5 to +5	-5
$V_{ref} \times sig$, where $-1 \le sig \le +1$	APFI0
$V_{ref} \times sig$, where $-1 \le sig \le +1$	APFI1

For example, if the first channel is -10 volts 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 replicated over the channel vector. If you specify a value of `1`, the corresponding channel is reset to the value specified in the initial value vector. If you specify a value of `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 replicated as the initial value over the channel vector. 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 (`-1` means sample time is inherited).

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

External Websites

www.ni.com

National Instruments PCI-6259 Digital Input

Digital Input block for PCI-6259, PCIe-6259, PXI-6259, and PXIe-6259 boards

Library

Simulink Real-Time Library for National Instruments

Note

Models can have multiple digital input blocks for one physical board. In this case, the Simulink Real-Time 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.

A digital input block can share channels with a digital output block. In this configuration, the input block reads the last value written to the digital output block on the shared channels.

Scaling of Input to Output

I/O Module 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 arbitrary 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 if this board manufacturer starts numbering the lines with 0.

Sample time

Enter the base sample time or a multiple of the base sample time (`-1` means sample time is inherited).

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

External Websites

www.ni.com

National Instruments PCI-6259 Digital Output

Digital Output block for PCI-6259, PCIe-6259, PXI-6259, and PXIe-6259 boards

Library

Simulink Real-Time Library for National Instruments

Note

Models can have multiple digital output blocks for one physical board. In this case, the Simulink Real-Time 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.

A digital input block can share channels with a digital output block. In this configuration, the input block reads the last value written to the digital output block on the shared channels.

The block compares the value written to each port with 0.5.

Value written	Block writes digital output with a value of...
>= 0.5	1
<0.5	0

Scaling of Output to Input

I/O Module 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 32 to select the digital output lines used with this port. This driver allows the selection of individual digital output lines in arbitrary 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 this 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 replicated over the channel vector. If you specify a value of `1`, the corresponding channel is reset to the value specified in the initial value vector. If you specify a value of `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 replicated as the initial value over the channel vector. 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 (`-1` means sample time is inherited).

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

External Websites

www.ni.com

National Instruments PCI-6259 Incremental Encoder

Incremental Encoder block for PCI-6259, PCIe-6259, PXI-6259, and PXIe-6259 boards

Library

Simulink Real-Time Library for National Instruments

Scaling Input to Output

I/O Module 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

Mode	Description
`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`

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 the input is not filtered.

- `None`
- `Minimum pulse width 125 nanoseconds`
- `Minimum pulse width 6.25 microseconds`
- `Minimum pulse width 1.25 milliseconds`

Sample time

Enter the base sample time or a multiple of the base sample time (`-1` means sample time is inherited).

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

External Websites

www.ni.com

National Instruments PCI-6259 PFI Digital Input

PFI Digital Input block for PCI-6259, PCIe-6259, PXI-6259, and PXIe-6259 boards

Library

Simulink Real-Time Library for National Instruments

Description

A model can have more than one PFI digital input block for a 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

I/O Module Input	Block Output Data Type	Scaling
TTL	Double	TTL low ≥ 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 the bits you want to monitor. For example, to read bits PFI 0, PFI 5, and PFI 7, enter

```
[0, 5, 7]
```

Sample time

Enter the base sample time or a multiple of the base sample time (`-1` means sample time is inherited).

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

External Websites

www.ni.com

National Instruments PCI-6259 PFI Digital Output

PFI Digital Output block for PCI-6259, PCIe-6259, PXI-6259, and PXIe-6259 boards

Library

Simulink Real-Time Library for National Instruments

Description

A model can have more than one PFI digital input block for a 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

I/O Module 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 the 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 of the channel at model termination. Enter a scalar or a vector that is the same length as the channel vector. If you specify a scalar value, that setting is replicated over the channel vector. If you specify a value of `1`, the corresponding channel is reset to the value specified in the initial value vector. If you specify a value of `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 block outputs. Enter a scalar or a vector of values, one for each output. If you enter a scalar, the driver replicates that value over the output vector.

The block applies this value as follows:

- The model uses this value when you first download the model to the target computer.
- 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

Enter the base sample time or a multiple of the base sample time (`-1` means sample time is inherited).

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

External Websites

www.ni.com

National Instruments PCI-6259 PWM Generate

PWM Generate block for PCI-6259, PCIe-6259, PXI-6259, and PXIe-6259 boards

Library

Simulink Real-Time Library for National Instruments

Description

Use this block to generate variable frequency and pulse width output.

The 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 number of counts the output is high 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 low until the first time the Simulink engine calls `mdlOutputs`.

Initial low count

Enter the number of counts the output is low 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

Enter the base sample time or a multiple of the base sample time (`-1` means sample time is inherited).

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

External Websites

www.ni.com

National Instruments PCI-6259 PWM Measure

PWM Measure block for PCI-6259, PCIe-6259, PXI-6259, and PXIe-6259 boards

Library

Simulink Real-Time 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 the previous value.

This block returns 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 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 the input is not filtered

- `None`
- `Minimum pulse width 125 nanoseconds`
- `Minimum pulse width 6.25 microseconds`
- `Minimum pulse width 1.25 milliseconds`

Sample time

Enter the base sample time or a multiple of the base sample time (`-1` means sample time is inherited).

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

External Websites

www.ni.com

National Instruments PCI-6280

Support for PCI-6280 and PXI-6280 boards

Board

National Instruments PCI-6280, PXI-6280

General Description

The 6280 M-series boards are data acquisition I/O boards that provide:

- Up to 16 single-ended or 8 differential 18-bit analog inputs
- 24 digital I/O lines
- Two 32-bit 80 MHz counters

The Simulink Real-Time block library supports the 6280 boards 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

Board Characteristics

Board name	PCI-6280, PXI-6280
Manufacturer	National Instruments
Bus type	PCI, PXI (both use PCI version of blocks)

Multiple block instance support	A/D: No, Digital I/O: Yes
Multiple board support	Yes

See Also

External Websites

www.ni.com

National Instruments PCI-6280 Analog Input

Analog Input block for PCI-6280 and PXI-6280 boards

Library

Simulink Real-Time Library for National Instruments

Scaling of Input to Output

I/O Module Input	Block Output Data Type	Scaling
Volts	Double	1

Block Parameters

Channel vector

Enter numbers between 1 and 16. This driver allows you to enter channel numbers in arbitrary order.

For example, to use the first, second, and fifth channels, enter

```
[1,2,5]
```

Number the lines beginning with 1 even if this 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 replicate over the channel vector, 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.

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

Input Range (V)	Range Code
-2 to +2	2
-1 to +1	1
-0.5 to +0.5	0.5
-0.2 to +0.2	0.2
-0.1 to +0.1	0.1

For example, if the first channel is -10 volts to +10 volts and the second and fifth channels are -1 to +1 volts, enter

```
[10,1,1]
```

Input coupling vector

Enter a coupling code for each of the channels in the channel vector. You can replicate one value over the channel vector, 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

```
[0,0,2]
```

National Instruments boards use a zero-based number scheme for differential mode channel pairs. Simulink Real-Time drivers use a one-based number scheme that adds a 1 to the I/O module number, as shown in the following table:

Zero-Based Channel Number		One-Based Channel Number	
+	-	+	-
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 I/O module uses a second input 8 channels higher than the first channel as the negative input of the pair. In the example above, the I/O module would use the 13th channel as a differential input with the fifth channel.

If the module 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

Enter the base sample time or a multiple of the base sample time (`-1` means sample time is inherited).

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

External Websites

www.ni.com

National Instruments PCI-6280 Digital Input

Digital Input block for PCI-6280 and PXI-6280 boards

Library

Simulink Real-Time Library for National Instruments

Note

Models can have multiple digital input blocks for one physical board. In this case, the Simulink Real-Time 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.

A digital input block can share channels with a digital output block. In this configuration, the input block reads the last value written to the digital output block on the shared channels.

Scaling of Input to Output

I/O Module 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 arbitrary 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 if this board manufacturer starts numbering the lines with 0.

Sample time

Enter the base sample time or a multiple of the base sample time (`-1` means sample time is inherited).

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

External Websites

www.ni.com

National Instruments PCI-6280 Digital Output

Digital Output block for PCI-6280 and PXI-6280 boards

Library

Simulink Real-Time Library for National Instruments

Note

Models can have multiple digital output blocks for one physical board. In this case, the Simulink Real-Time 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.

A digital input block can share channels with a digital output block. In this configuration, the input block reads the last value written to the digital output block on the shared channels.

The block compares the value written to each port with 0.5.

Value written	Block writes digital output with a value of...
>= 0.5	1
<0.5	0

Scaling of Output to Input

I/O Module 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 arbitrary 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 this 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 replicated over the channel vector. If you specify a value of `1`, the corresponding channel is reset to the value specified in the initial value vector. If you specify a value of `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 replicated as the initial value over the channel vector. 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 (`-1` means sample time is inherited).

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

External Websites

www.ni.com

National Instruments PCI-6280 Incremental Encoder

Incremental Encoder block for PCI-6280 and PXI-6280 boards

Library

Simulink Real-Time Library for National Instruments

Scaling Input to Output

I/O Module 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

Mode	Description
`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`

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 the input is not filtered.

- `None`
- `Minimum pulse width 125 nanoseconds`
- `Minimum pulse width 6.25 microseconds`
- `Minimum pulse width 1.25 milliseconds`

Sample time

Enter the base sample time or a multiple of the base sample time (`-1` means sample time is inherited).

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

External Websites

www.ni.com

National Instruments PCI-6280 PWM Generate

PWM Generate block for PCI-6280 and PXI-6280 boards

Library

Simulink Real-Time Library for National Instruments

Description

Use this block to generate variable frequency and pulse width output.

The 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 number of counts the output is high 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 low until the first time the Simulink engine calls `mdlOutputs`.

Initial low count

Enter the number of counts the output is low 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

Enter the base sample time or a multiple of the base sample time (`-1` means sample time is inherited).

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

External Websites

www.ni.com

National Instruments PCI-6280 PFI Digital Input

PFI Digital Input block for PCI-6280 and PXI-6280 boards

Library

Simulink Real-Time Library for National Instruments

Description

A model can have more than one PFI digital input block for a 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

I/O Module Input	Block Output Data Type	Scaling
TTL	Double	TTL low ≥ 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 the bits you want to monitor. For example, to read bits PFI 0, PFI 5, and PFI 7, enter

```
[0, 5, 7]
```

Sample time

Enter the base sample time or a multiple of the base sample time (`-1` means sample time is inherited).

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

External Websites

www.ni.com

National Instruments PCI-6280 PFI Digital Output

PFI Digital Output block for PCI-6280 and PXI-6280 boards

Library

Simulink Real-Time Library for National Instruments

Description

A model can have more than one PFI digital input block for a 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

I/O Module 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 the 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 of the channel at model termination. Enter a scalar or a vector that is the same length as the channel vector. If you specify a scalar value, that setting is replicated over the channel vector. If you specify a value of `1`, the corresponding channel is reset to the value specified in the initial value vector. If you specify a value of `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 block outputs. Enter a scalar or a vector of values, one for each output. If you enter a scalar, the driver replicates that value over the output vector.

The block applies this value as follows:

- The model uses this value when you first download the model to the target computer.
- 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

Enter the base sample time or a multiple of the base sample time (`-1` means sample time is inherited).

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

External Websites

www.ni.com

National Instruments PCI-6280 PWM Measure

PWM Measure block for PCI-6280 and PXI-6280 boards

Library

Simulink Real-Time 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 the previous value.

This block returns 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 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:

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 the input is not filtered

- `None`
- `Minimum pulse width 125 nanoseconds`
- `Minimum pulse width 6.25 microseconds`
- `Minimum pulse width 1.25 milliseconds`

Sample time

Enter the base sample time or a multiple of the base sample time (`-1` means sample time is inherited).

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

External Websites

www.ni.com

National Instruments PCI-6281

Support for PCI-6281 and PXI-6281 boards

Board

National Instruments PCI-6281, PXI-6281

General Description

The 6281 M-series boards are data acquisition I/O boards that provide:

- Up to 16 single-ended or 8 differential 18-bit analog inputs
- Two 16-bit analog outputs
- 24 digital I/O lines
- Two 32-bit 80 MHz counters

The Simulink Real-Time block library supports the 6281 boards with these driver blocks:

- 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 Simulink Real-Time block library supports this board with these driver blocks:

Board Characteristics

Board name	PCI-6281, PXI-6281

Manufacturer	National Instruments
Bus type	PCI, PXI (both use PCI version of blocks)
Multiple block instance support	A/D: No, D/A: No, Digital I/O: Yes
Multiple board support	Yes

See Also

External Websites

www.ni.com

National Instruments PCI-6281 Analog Input

Analog Input block for PCI-6281 and PXI-6281 boards

Library

Simulink Real-Time Library for National Instruments

Scaling of Input to Output

I/O Module Input	Block Output Data Type	Scaling
Volts	Double	1

Block Parameters

Channel vector

Enter numbers between 1 and 16. This driver allows you to enter channel numbers in arbitrary order.

For example, to use the first, second, and fifth channels, enter

```
[1,2,5]
```

Number the lines beginning with 1 even if this 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 replicate over the channel vector, 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.

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

Input Range (V)	Range Code
-2 to +2	2
-1 to +1	1
-0.5 to +0.5	0.5
-0.2 to +0.2	0.2
-0.1 to +0.1	0.1

For example, if the first channel is -10 volts to +10 volts and the second and fifth channels are -1 to +1 volts, enter

```
[10,1,1]
```

Input coupling vector

Enter a coupling code for each of the channels in the channel vector. You can replicate one value over the channel vector, 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

```
[0,0,2]
```

National Instruments boards use a zero-based number scheme for differential mode channel pairs. Simulink Real-Time drivers use a one-based number scheme that adds a 1 to the I/O module number, as shown in the following table:

Zero-Based Channel Number		One-Based Channel Number	
+	-	+	-
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 I/O module uses a second input 8 channels higher than the first channel as the negative input of the pair. In the example above, the I/O module would use the 13th channel as a differential input with the fifth channel.

If the module 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

Enter the base sample time or a multiple of the base sample time (`-1` means sample time is inherited).

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

External Websites

www.ni.com

National Instruments PCI-6281 Analog Output

Analog Output block for PCI-6281 and PXI-6281 boards

Library

Simulink Real-Time Library for National Instruments

Scaling of Output to Input

I/O Module 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 arbitrary 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 this 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

Input Range (V)	Range Code
-5 to +5	-5
$V_{ref} \times sig$, where $-1 \le sig \le +1$	APFI0

For example, if the first channel is -10 volts 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 replicated over the channel vector. If you specify a value of `1`, the corresponding channel is reset to the value specified in the initial value vector. If you specify a value of `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 replicated as the initial value over the channel vector. 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 (`-1` means sample time is inherited).

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

External Websites

www.ni.com

National Instruments PCI-6281 Digital Input

Digital Input block for PCI-6281 and PXI-6281 boards

Library

Simulink Real-Time Library for National Instruments

Note

Models can have multiple digital input blocks for one physical board. In this case, the Simulink Real-Time 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.

A digital input block can share channels with a digital output block. In this configuration, the input block reads the last value written to the digital output block on the shared channels.

Scaling of Input to Output

I/O Module 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 arbitrary 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 if this board manufacturer starts numbering the lines with 0.

Sample time

Enter the base sample time or a multiple of the base sample time (`-1` means sample time is inherited).

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

External Websites

www.ni.com

National Instruments PCI-6281 Digital Output

Digital Output block for PCI-6281 and PXI-6281 boards

Library

Simulink Real-Time Library for National Instruments

Note

Models can have multiple digital output blocks for one physical board. In this case, the Simulink Real-Time 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.

A digital input block can share channels with a digital output block. In this configuration, the input block reads the last value written to the digital output block on the shared channels.

The block compares the value written to each port with 0.5.

Value written	Block writes digital output with a value of...
>= 0.5	1
<0.5	0

Scaling of Output to Input

I/O Module 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 arbitrary 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 this 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 replicated over the channel vector. If you specify a value of `1`, the corresponding channel is reset to the value specified in the initial value vector. If you specify a value of `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 replicated as the initial value over the channel vector. 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 (`-1` means sample time is inherited).

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

External Websites

www.ni.com

National Instruments PCI-6281 Incremental Encoder

Incremental Encoder block for PCI-6281 and PXI-6281 boards

Library

Simulink Real-Time Library for National Instruments

Scaling Input to Output

I/O Module 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

Mode	Description
`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`

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 the input is not filtered.

- `None`
- `Minimum pulse width 125 nanoseconds`
- `Minimum pulse width 6.25 microseconds`
- `Minimum pulse width 1.25 milliseconds`

Sample time

Enter the base sample time or a multiple of the base sample time (`-1` means sample time is inherited).

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

External Websites

www.ni.com

National Instruments PCI-6281 PFI Digital Input

PFI Digital Input block for PCI-6281 and PXI-6281 boards

Library

Simulink Real-Time Library for National Instruments

Description

A model can have more than one PFI digital input block for a 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

I/O Module Input	Block Output Data Type	Scaling
TTL	Double	TTL low ≥ 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 the bits you want to monitor. For example, to read bits PFI 0, PFI 5, and PFI 7, enter

```
[0, 5, 7]
```

Sample time

Enter the base sample time or a multiple of the base sample time (`-1` means sample time is inherited).

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

External Websites

www.ni.com

National Instruments PCI-6281 PFI Digital Output

PFI Digital Output block for PCI-6281 and PXI-6281 boards

Library

Simulink Real-Time Library for National Instruments

Description

A model can have more than one PFI digital input block for a 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

I/O Module 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 the 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 of the channel at model termination. Enter a scalar or a vector that is the same length as the channel vector. If you specify a scalar value, that setting is replicated over the channel vector. If you specify a value of `1`, the corresponding channel is reset to the value specified in the initial value vector. If you specify a value of `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 block outputs. Enter a scalar or a vector of values, one for each output. If you enter a scalar, the driver replicates that value over the output vector.

The block applies this value as follows:

- The model uses this value when you first download the model to the target computer.
- 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

Enter the base sample time or a multiple of the base sample time (`-1` means sample time is inherited).

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

External Websites

www.ni.com

National Instruments PCI-6281 PWM Generate

PWM Generate block for PCI-6281 and PXI-6281 boards

Library

Simulink Real-Time Library for National Instruments

Description

Use this block to generate variable frequency and pulse width output.

The 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 number of counts the output is high 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 low until the first time the Simulink engine calls `mdlOutputs`.

Initial low count

Enter the number of counts the output is low 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

Enter the base sample time or a multiple of the base sample time (`-1` means sample time is inherited).

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

External Websites

www.ni.com

National Instruments PCI-6281 PWM Measure

PWM Measure block for PCI-6281 and PXI-6281 boards

Library

Simulink Real-Time 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 the previous value.

This block returns 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 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 the input is not filtered

- `None`
- `Minimum pulse width 125 nanoseconds`
- `Minimum pulse width 6.25 microseconds`
- `Minimum pulse width 1.25 milliseconds`

Sample time

Enter the base sample time or a multiple of the base sample time (`-1` means sample time is inherited).

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

External Websites

www.ni.com

National Instruments PCI-6289

Support for PCI-6289 and PXI-6289 boards

Board

National Instruments PCI-6289, PXI-6289

General Description

The 6289 M-series boards are data acquisition I/O boards that provide:

- Up to 32 single-ended or 16 differential 18-bit analog inputs
- Four 16-bit analog outputs
- 48 digital I/O lines
- Two 32-bit 80 MHz counters

The Simulink Real-Time block library supports the 6289 boards 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, PXI-6289

Manufacturer	National Instruments
Bus type	PCI, PXI (both use PCI version of blocks)
Multiple block instance support	A/D: No, D/A: No, Digital I/O: Yes
Multiple board support	Yes

See Also

External Websites

www.ni.com

National Instruments PCI-6289 Analog Input

Analog Input block for PCI-6289 and PXI-6289 boards

Library

Simulink Real-Time Library for National Instruments

Scaling of Input to Output

I/O Module Input	Block Output Data Type	Scaling
Volts	Double	1

Block Parameters

Channel vector

Enter numbers between 1 and 32. This driver allows you to enter channel numbers in arbitrary order.

For example, to use the first, second, and fifth channels, enter

```
[1,2,5]
```

Number the lines beginning with 1 even if this 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 replicate over the channel vector, 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.

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

Input Range (V)	Range Code
-5 to +5	5
-2 to +2	2
-1 to +1	1
-0.5 to +0.5	0.5
-0.2 to +0.2	0.2
-0.1 to +0.1	0.1

For example, if the first channel is -10 volts to +10 volts and the second and fifth channels are -1 to +1 volts, enter

```
[10,1,1]
```

Input coupling vector

Enter a coupling code for each of the channels in the channel vector. You can replicate one value over the channel vector, 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

```
[0,0,2]
```

National Instruments boards use a zero-based number scheme for differential mode channel pairs. Simulink Real-Time drivers use a one-based number scheme that adds a 1 to the I/O module number, as shown in the following table:

Zero-Based Channel Number		One-Based Channel Number	
+	-	+	-
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 I/O module uses a second input 8 channels higher than the first channel as the negative input of the pair. In the example above, the I/O module would use the 13th channel as a differential input with the fifth channel.

If the module 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

Enter the base sample time or a multiple of the base sample time (`-1` means sample time is inherited).

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

External Websites

www.ni.com

National Instruments PCI-6289 Analog Output

Analog Output block for PCI-6289 and PXI-6289 boards

Library

Simulink Real-Time Library for National Instruments

Scaling of Output to Input

I/O Module 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 arbitrary 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 if this 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

Input Range (V)	Range Code
-5 to +5	-5
$V_{ref} \times sig$, where $-1 \le sig \le +1$	APFI0
$V_{ref} \times sig$, where $-1 \le sig \le +1$	APFI1

For example, if the first channel is -10 volts 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 replicated over the channel vector. If you specify a value of `1`, the corresponding channel is reset to the value specified in the initial value vector. If you specify a value of `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 replicated as the initial value over the channel vector. 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 (`-1` means sample time is inherited).

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

External Websites

www.ni.com

National Instruments PCI-6289 Digital Input

Digital Input block for PCI-6289 and PXI-6289 boards

Library

Simulink Real-Time Library for National Instruments

Note

Models can have multiple digital input blocks for one physical board. In this case, the Simulink Real-Time 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.

A digital input block can share channels with a digital output block. In this configuration, the input block reads the last value written to the digital output block on the shared channels.

Scaling of Input to Output

I/O Module 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 32 to select the digital input lines used with this port. This driver allows the selection of individual digital input lines in arbitrary 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 if this board manufacturer starts numbering the lines with 0.

Sample time

Enter the base sample time or a multiple of the base sample time (`-1` means sample time is inherited).

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

External Websites

www.ni.com

National Instruments PCI-6289 Digital Output

Digital Output block for PCI-6289 and PXI-6289 boards

Library

Simulink Real-Time Library for National Instruments

Note

Models can have multiple digital output blocks for one physical board. In this case, the Simulink Real-Time 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.

A digital input block can share channels with a digital output block. In this configuration, the input block reads the last value written to the digital output block on the shared channels.

The block compares the value written to each port with 0.5.

Value written	Block writes digital output with a value of...
>= 0.5	1
<0.5	0

Scaling of Output to Input

I/O Module 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 32 to select the digital output lines used with this port. This driver allows the selection of individual digital output lines in arbitrary 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 this 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 replicated over the channel vector. If you specify a value of `1`, the corresponding channel is reset to the value specified in the initial value vector. If you specify a value of `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 replicated as the initial value over the channel vector. 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 (`-1` means sample time is inherited).

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

External Websites

www.ni.com

National Instruments PCI-6289 Incremental Encoder

Incremental Encoder block for PCI-6289 and PXI-6289 boards

Library

Simulink Real-Time Library for National Instruments

Scaling Input to Output

I/O Module 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

Mode	Description
`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`

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 the input is not filtered.

- `None`
- `Minimum pulse width 125 nanoseconds`
- `Minimum pulse width 6.25 microseconds`
- `Minimum pulse width 1.25 milliseconds`

Sample time

Enter the base sample time or a multiple of the base sample time (`-1` means sample time is inherited).

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

External Websites

www.ni.com

National Instruments PCI-6289 PFI Digital Input

PFI Digital Input block for PCI-6289 and PXI-6289 boards

Library

Simulink Real-Time Library for National Instruments

Description

A model can have more than one PFI digital input block for a 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

I/O Module Input	Block Output Data Type	Scaling
TTL	Double	TTL low ≥ 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 the bits you want to monitor. For example, to read bits PFI 0, PFI 5, and PFI 7, enter

```
[0, 5, 7]
```

Sample time

Enter the base sample time or a multiple of the base sample time (`-1` means sample time is inherited).

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

External Websites

www.ni.com

National Instruments PCI-6289 PFI Digital Output

PFI Digital Output block for PCI-6289 and PXI-6289 boards

Library

Simulink Real-Time Library for National Instruments

Description

A model can have more than one PFI digital input block for a 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

I/O Module 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 the 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 of the channel at model termination. Enter a scalar or a vector that is the same length as the channel vector. If you specify a scalar value, that setting is replicated over the channel vector. If you specify a value of `1`, the corresponding channel is reset to the value specified in the initial value vector. If you specify a value of `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 block outputs. Enter a scalar or a vector of values, one for each output. If you enter a scalar, the driver replicates that value over the output vector.

The block applies this value as follows:

- The model uses this value when you first download the model to the target computer.
- 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

Enter the base sample time or a multiple of the base sample time (`-1` means sample time is inherited).

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

External Websites

www.ni.com

National Instruments PCI-6289 PWM Generate

PWM Generate block for PCI-6289 and PXI-6289 boards

Library

Simulink Real-Time Library for National Instruments

Description

Use this block to generate variable frequency and pulse width output.

The 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 number of counts the output is high 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 low until the first time the Simulink engine calls `mdlOutputs`.

Initial low count

Enter the number of counts the output is low 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

Enter the base sample time or a multiple of the base sample time (`-1` means sample time is inherited).

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

External Websites

www.ni.com

National Instruments PCI-6289 PWM Measure

PWM Measure block for PCI-6289 and PXI-6289 boards

Library

Simulink Real-Time 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 the previous value.

This block returns 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 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 the input is not filtered

- `None`
- `Minimum pulse width 125 nanoseconds`
- `Minimum pulse width 6.25 microseconds`
- `Minimum pulse width 1.25 milliseconds`

Sample time

Enter the base sample time or a multiple of the base sample time (`-1` means sample time is inherited).

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

External Websites

www.ni.com

National Instruments PCI-6503

Support for National Instruments PCI-6503 board

Board

National Instruments PCI-6503

General Description

The PCI-6503 provides:

- 24 digital input lines
- 24 digital output lines

The Simulink Real-Time block library supports this board with these driver blocks:

- National Instruments PCI-6503 Digital Input
- National Instruments PCI-6503 Digital Output

Board Characteristics

Board name	PCI-6503
Manufacturer	National Instruments
Bus type	PCI
Access method	I/O mapped
Multiple block instance support	Yes
Multiple board support	Yes

See Also

External Websites

www.ni.com

National Instruments PCI-6503 Digital Input

National Instruments PCI-6503 Digital Input block

Library

Simulink Real-Time 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

I/O Module 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 arbitrary 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 this 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 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 (`-1` means sample time is inherited).

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

External Websites

www.ni.com

National Instruments PCI-6503 Digital Output

National Instruments PCI-6503 Digital Output block

Library

Simulink Real-Time 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

I/O Module 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 arbitrary 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 this 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 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 replicated over the channel vector. If you specify a value of `1`, the corresponding channel is reset to the value specified in the initial value vector. If you specify a value of `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 replicated as the initial value over the channel vector. 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 (`-1` means sample time is inherited).

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

External Websites

www.ni.com

National Instruments PCI-6527

Support for National Instruments PCI-6527 board

Board

National Instruments PCI-6527

General Description

The PCI-6527 provides:

- 24 filtered digital input lines
- 24 digital output lines

You can turn input filtering on or off individually for each of the 24 input lines, but the same filter time applies to each line.

The Simulink Real-Time block library supports this board with these driver blocks:

- National Instruments PCI-6527 Digital Input
- National Instruments PCI-6527 Digital Output

Board Characteristics

Board name	PCI-6527
Manufacturer	National Instruments
Bus type	PCI
Access method	Memory mapped
Multiple block instance support	Digital I/O: Yes
Multiple board support	Yes

See Also

External Websites

www.ni.com

National Instruments PCI-6527 Digital Input

National Instruments PCI-6527 Digital Input block

Library

Simulink Real-Time Library for National Instruments

Note

The PCI-6527 has one chip with three 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

I/O Module 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 arbitrary 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 this 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.

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 scaler 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 filter uses to determine a stable pulse. If the input pule is shorter then this interval, it is ignored. The same filter interval applies to each filtered input, and if you filter on more than one digital input block for this board, you must enter the same **Filter interval** for each block.

A reasonable value for the **Filter interval** would be 200 to 300 us.

Sample time

Enter the base sample time or a multiple of the base sample time (`-1` means sample time is inherited).

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

External Websites

www.ni.com

National Instruments PCI-6527 Digital Output

PCI-6527 Digital Output block

Library

Simulink Real-Time Library for National Instruments

Note

The PCI-6527 has one chip with three ports (A,B,C) for digital output. Each port has a maximum of eight digital output lines. Use a separate driver block for each port.

Scaling of Input to Output

I/O Module Output	Block Output Data Type	Scaling
OPTO	Double	<0.5 = OPTO is OFF, current not 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 arbitrary 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 this 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 or 8 digital output lines. In each case, one block is required for each port.

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 replicated over the channel vector. If you specify a value of 1, the corresponding channel is reset to the value specified in the initial value vector. If you specify a value of 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 replicated as the initial value over the channel vector. 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 (-1 means sample time is inherited).

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter -1 to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

External Websites

www.ni.com

National Instruments PCI-6528

Support for National Instruments PCI-6528 board

Board

National Instruments PCI-6528

General Description

The PCI-6528 provides:

- 24 filtered digital input lines
- 24 digital output lines

You can individually turn input filtering on or off for each of the 24 input lines, but the same filter time applies to each line.

The Simulink Real-Time block library supports this board with these driver blocks:

- National Instruments PCI-6527 Digital Input
- National Instruments PCI-6527 Digital Output

Board Characteristics

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

See Also

External Websites

www.ni.com

National Instruments PCI-6528 Digital Input

PCI-6528 Digital Input block

Library

Simulink Real-Time Library for National Instruments

Note

The PCI-6528 has one chip with three 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

I/O Module 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 arbitrary 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 this 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.

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 filter uses to determine a stable pulse. If the input pulse is shorter than this interval, it is ignored. The same filter interval applies to each filtered input, and if you filter on more than one digital input block for this board, you must enter the same **Filter interval** for each block.

A reasonable value for the **Filter interval** would be 200 to 300 µs.

Sample time

Enter the base sample time or a multiple of the base sample time (`-1` means sample time is inherited).

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

External Websites

www.ni.com

National Instruments PCI-6528 Digital Output

PCI-6528 Digital Output block

Library

Simulink Real-Time Library for National Instruments

Note

The PCI-6528 has one chip with three ports (A,B,C) for digital output. Each port has a maximum of eight digital output lines. Use a separate driver block for each port.

Scaling of Input to Output

I/O Module Output	Block Output Data Type	Scaling
OPTO	Double	<0.5 = OPTO is OFF, current not 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 arbitrary 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 this 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 or 8 digital output lines. In each case, one block is required for each port.

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 replicated over the channel vector. If you specify a value of `1`, the corresponding channel is reset to the value specified in the initial value vector. If you specify a value of `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 replicated as the initial value over the channel vector. 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 (`-1` means sample time is inherited).

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

External Websites

www.ni.com

National Instruments PCI-6601

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 that provides four 32-bit counter channels. The reference frequency for this board is 20 MHz.

The Simulink Real-Time 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 Simulink Real-Time software does not support the interrupt or timer functionality of the board.

Board Characteristics

Board name	PCI-6601
Manufacturer	National Instruments
Bus type	PCI
Access method	I/O mapped

Multiple block instance support	Yes
Multiple board support	Yes

See Also

External Websites

www.ni.com

National Instruments PCI-6601 Digital Input

PCI-6601 Digital Input block

Library

Simulink Real-Time Library for National Instruments

Note

Only one input block can exist in a model for a particular 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 arbitrary 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 if this 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 the base sample time or a multiple of the base sample time (`-1` means sample time is inherited).

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

External Websites

www.ni.com

National Instruments PCI-6601 Digital Output

PCI-6601 Digital Output block

Library

Simulink Real-Time Library for National Instruments

Note

Only one output block can exist in a model for a particular board.

Block Parameters

Channel

Enter numbers between 1 and 8 to select the digital output ports. This driver allows the selection of individual digital output ports in arbitrary 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 if this 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 value is replicated over

the channel vector. If you specify a value of 1, the corresponding channel is reset to the value specified in 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 replicated over the channel vector. 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 real-time 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 real-time 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.

Sample time

Enter the base sample time or a multiple of the base sample time (`-1` means sample time is inherited).

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

External Websites

www.ni.com

National Instruments PCI-6601 Incremental Encoder

PCI-6601 Incremental Encoder block

Library

Simulink Real-Time 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 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 (`-1` means sample time is inherited).

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

External Websites

www.ni.com

National Instruments PCI-6601 Pulse Generation

PCI-6601 Pulse Generation block

Library

Simulink Real-Time 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

Enter the base sample time or a multiple of the base sample time (`-1` means sample time is inherited).

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

External Websites

www.ni.com

National Instruments PCI-6601 Pulse Width/Period Measurement

PCI-6601 Pulse Width/Period Measurement block

Library

Simulink Real-Time 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 can experience false triggering. Use signal voltage levels of 0-5 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** interact. 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** interact. 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.

Sample time

Enter the base sample time or a multiple of the base sample time (`-1` means sample time is inherited).

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

External Websites

www.ni.com

National Instruments PCI-6601 Armed Pulse Generation

PCI-6601 Armed Pulse Generation block

Library

Simulink Real-Time 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

Enter the base sample time or a multiple of the base sample time (`-1` means sample time is inherited).

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

External Websites

www.ni.com

National Instruments PCI-6601 Event Counter

PCI-6601 Event Counter block

Library

Simulink Real-Time Library for National Instruments

Description

Use the Event Counter block for measuring the frequency of a pulse train and for counting binary events, such as switch closures and photocell interruptions. See "Examples" on page 42-412.

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 (I/O module channels 0–3). 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 without latching.

- `HW gated cumulative counter`

Accumulates count of edges of the source signal when the gate condition is true. Counter is reset according to **Reset Mode** (the gate end does *not* reset the count). Block returns the current count without 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 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 (≤ 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 )`
- `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 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 each counter. 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.

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` remains 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

Enter the base sample time or a multiple of the base sample time (`-1` means sample time is inherited).

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

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:

Block Parameter	Value
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 a choice faster than the signal being measured
HW Gate Input	`Default Gate Pin`, or some 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	Ignored

Note: To use a **Gate Trigger Mode** of `Level Active High` or `Level Active Low` for a frequency counter, configure the gate to be both on and off for longer than the period of the signal. The gate must be on for long enough to latch an edge. It must be off for long enough that a source edge occurs while the gate is off. If an edge does not occur while the gate is off, the gate end condition will be missed, and the counter will add the next interval.

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:

Block Parameter	Value
Counting Mode	`Cumulative counter`

Block Parameter	Value
Arm Control Input	Checked or cleared. If checked, use arm input `A` in the model to control when to accept events.
Source	`Default Source Pin`, or some other source pin
Source Polarity	`Rising Edge` or `Falling Edge` (choose the cleaner edge)
Filter	`None`, or a pulse width large enough to eliminate switch bounce and similar artifacts
HW Gate Input	Ignored
Gate Trigger Mode	Ignored
Reset Mode	`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.

See Also

External Websites

www.ni.com

National Instruments PCI/PXI-6602

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 that provides eight 32-bit counter channels. The reference frequency for this board is 80 MHz.

The Simulink Real-Time 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 Simulink Real-Time software does not support the interrupt or timer functionality of the board.

Board Characteristics

Board name	PCI/PXI-6602
Manufacturer	National Instruments
Bus type	PCI/PXI
Access method	I/O mapped

Multiple block instance support	Yes
Multiple board support	Yes

See Also

External Websites

www.ni.com

National Instruments PCI/PXI-6602 Digital Input

PCI/PXI-6602 Digital Input block

Library

Simulink Real-Time Library for National Instruments

Note

Only one input block can exist in a model for a particular 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 arbitrary 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 if this 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 the base sample time or a multiple of the base sample time (`-1` means sample time is inherited).

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

External Websites

www.ni.com

National Instruments PCI/PXI-6602 Digital Output

PCI/PXI-6602 Digital Output block

Library

Simulink Real-Time Library for National Instruments

Note

Only one output block can exist in a model for a particular board.

Block Parameters

Channel

Enter numbers between 1 and 8 to select the digital output ports. This driver allows the selection of individual digital output ports in arbitrary 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 if this 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 value is replicated over

the channel vector. If you specify a value of 1, the corresponding channel is reset to the value specified in 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 replicated over the channel vector. 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 real-time 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 real-time 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.

Sample time

Enter the base sample time or a multiple of the base sample time (`-1` means sample time is inherited).

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

External Websites

www.ni.com

National Instruments PCI/PXI-6602 Incremental Encoder

PCI/PXI-6602 Incremental Encoder block

Library

Simulink Real-Time 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 if this 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.

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 (`-1` means sample time is inherited).

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

External Websites

www.ni.com

National Instruments PCI/PXI-6602 Pulse Generation

PCI/PXI-6602 Pulse Generation block

Library

Simulink Real-Time 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

Enter the base sample time or a multiple of the base sample time (`-1` means sample time is inherited).

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

External Websites

www.ni.com

National Instruments PCI/PXI-6602 Pulse Width/ Period Measurement

PCI/PXI-6602 Pulse Width/Period Measurement block

Library

Simulink Real-Time 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 can experience false triggering. Use signal voltage levels of 0-5 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** interact. 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** interact. 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.

Sample time

Enter the base sample time or a multiple of the base sample time (`-1` means sample time is inherited).

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

External Websites

www.ni.com

National Instruments PCI/PXI-6602 Armed Pulse Generation

PCI/PXI-6602 Armed Pulse Generation block

Library

Simulink Real-Time 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.

Sample time

Enter the base sample time or a multiple of the base sample time (`-1` means sample time is inherited).

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

External Websites

www.ni.com

National Instruments PCI/PXI-6602 Event Counter

PCI-6602 Event Counter block

Library

Simulink Real-Time Library for National Instruments

Description

Use the Event Counter for measuring the frequency of a pulse train and counting binary events, such as switch closures and photocell interruptions. See “Examples” on page 42-436.

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 (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 without latching.

- `HW gated cumulative counter`

  Accumulates count of edges of the source signal when the gate condition is true. Counter is reset according to **Reset Mode** (the gate end does *not* reset the count). Block returns the current count without latching.

  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 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 (≤ 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 one of three different standard frequencies.

- `Default Source Pin`
- `PFI 39 ( Pin 2 )`
- `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`

HW Gate Input

Select the PFI input pin to be used as gate when **Counting Mode** is set to a 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 each counter. 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 `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.

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` remains 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

Enter the base sample time or a multiple of the base sample time (`-1` means sample time is inherited).

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

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:

Block Parameter	Value
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 a choice faster than the signal being measured
HW Gate Input	`Default Gate Pin`, or some 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	Ignored

To use a **Gate Trigger Mode** of `Level Active High` or `Level Active Low` for a frequency counter, configure the gate to be both on and off for longer than the period of the signal. The gate must be on for long enough to latch an edge. It must be off for long enough that a source edge occurs while the gate is off. If an edge does not occur while the gate is off, the gate end condition will be missed, and the counter will add the next interval.

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:

Block Parameter	Value
Counting Mode	`Cumulative counter`

Block Parameter	Value
Arm Control Input	Checked or cleared. If checked, use arm input `A` in the model to control when to accept events.
Source	`Default Source Pin`, or some other source pin
Source Polarity	`Rising Edge` or `Falling Edge` (choose the cleaner edge)
Filter	`None`, or a pulse width required to eliminate switch bounce and similar artifacts
HW Gate Input	Ignored
Gate Trigger Mode	Ignored
Reset Mode	`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.

See Also

External Websites

www.ni.com

National Instruments PCI-6703

Support for National Instruments PCI-6703 board

Board

National Instruments PCI-6703

General Description

The PCI-6703 provides:

- 16 voltage outputs
- 8 digital I/O lines

The Simulink Real-Time block library supports this board with this driver block:

- National Instruments PCI-6703 Analog Output (D/A)

Simulink Real-Time does not support the 8 digital I/O lines.

You will notice a variable delay in the voltage output from the board after your real-time application writes the register on the board. This delay occurs because the PCI-6703 has a single D/A converter that is time-share multiplexed through the outputs. It can take up to 0.9 milliseconds to scan through the channels.

Board Characteristics

Board name	PCI-6703
Manufacturer	National Instruments
Bus type	PCI
Access method	Memory mapped
Multiple block instance support	No
Multiple board support	Yes

See Also

External Websites

www.ni.com

National Instruments PCI-6703 Analog Output (D/A)

PCI-6703 Analog Output block

Library

Simulink Real-Time Library for National Instruments

Scaling Input to Output

I/O Module Output	Block Input Data Type	Scaling
Volts	Double	1

Block Parameters

Channel vector

Enter numbers between 1 and 16. This driver allows the selection of individual D/A channels in arbitrary 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 this board manufacturer starts numbering the channels with 0.

Sample time

Enter the base sample time or a multiple of the base sample time (`-1` means sample time is inherited).

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

External Websites

www.ni.com

National Instruments PCI-6704

Support for National Instruments PCI-6704 board

Board

National Instruments PCI-6704

General Description

The PCI-6704 provides:

- 16 voltage outputs
- 16 current outputs
- 8 digital I/O lines

The Simulink Real-Time 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 real-time application writes the register on the board. This delay occurs because the PCI-6704 has a single D/A converter that is time-share multiplexed through the outputs. It can take up to 1.8 milliseconds to scan through the channels.

Board Characteristics

Board name	PCI-6704
Manufacturer	National Instruments
Bus type	PCI
Access method	Memory mapped
Multiple block instance support	No
Multiple board support	Yes

See Also

External Websites

www.ni.com

National Instruments PCI-6704 Analog Output (D/A)

PCI-6704 Analog Output block

Library

Simulink Real-Time Library for National Instruments

Scaling Input to Output

I/O Module Output	Block Input Data Type	Scaling
Volts — Channels 1–16	Double	-10 V to 10 V
Milliamperes — Channels 17–32	Double	0–20 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 arbitrary 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 this board manufacturer starts numbering the channels with 0.

Sample time

Enter the base sample time or a multiple of the base sample time (`-1` means sample time is inherited).

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

External Websites

www.ni.com

National Instruments PCI/PXI-6711

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 that provides:

- 4 analog output (D/A) channels (12-bit)
- 8 digital input and output lines
- 2 24-bit, 20 MHz counters/timers

The Simulink Real-Time 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/PXI
Multiple block instance support	D/A: No, Digital I/O: Yes
Multiple board support	Yes

See Also

External Websites

www.ni.com

National Instruments PCI/PXI-6711 Analog Output (D/A)

PCI/PXI-6711 Analog Output block

Library

Simulink Real-Time Library for National Instruments

Note

The analog output range of this board is set -10 volts to +10 volts.

Scaling Input to Output

I/O Module Output	Block Input Data Type	Scaling
Volts	Double	1

Block Parameters

Channel vector

Enter numbers between 1 and 4. This driver allows the selection of individual D/A channels in arbitrary 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 if this 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 replicated over the channel vector. If you specify a value of `1`, the corresponding channel is reset to the value specified in the initial value vector. If you specify a value of `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 replicated as the initial value over the channel vector. 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 (`-1` means sample time is inherited).

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

External Websites

www.ni.com

National Instruments PCI/PXI-6711 Digital Input

PCI/PXI-6711 Digital Input block

Library

Simulink Real-Time Library for National Instruments

Scaling Input to Output

I/O Module 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 block. This driver allows the selection of individual digital input lines in arbitrary 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 if this board manufacturer starts numbering the lines with 0.

Sample time

Enter the base sample time or a multiple of the base sample time (`-1` means sample time is inherited).

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

External Websites

www.ni.com

National Instruments PCI/PXI-6711 Digital Output

PCI/PXI-6711 Digital Output

Library

Simulink Real-Time Library for National Instruments

Scaling Input to Output

I/O Module 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 arbitrary 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 if this 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 replicated over the channel vector. If you specify a value of `1`, the corresponding channel is reset to the value specified in the initial value vector.

If you specify a value of `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 replicated as the initial value over the channel vector. 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 (`-1` means sample time is inherited).

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

External Websites

www.ni.com

National Instruments PCI/PXI-6711 Pulse Generation

PCI/PXI-6711 Pulse Generation block

Library

Simulink Real-Time Library for National Instruments

Note

The input to the block is a two-element vector. The first element is the number of counts (of the 20 MHz source clock) that the counter maintains at a low level. 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

Enter the base sample time or a multiple of the base sample time (`-1` means sample time is inherited).

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

External Websites

www.ni.com

National Instruments PCI/PXI-6711 Pulse Width/ Period Measurement

PCI/PXI-6711 Pulse Width/Period Measurement block

Library

Simulink Real-Time 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** interact. 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** interact. 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.

Sample time

Enter the base sample time or a multiple of the base sample time (`-1` means sample time is inherited).

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

External Websites

www.ni.com

National Instruments PCI/PXI-6713

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 that provides:

- 8 analog output (D/A) channels (12-bit)
- 8 digital input and output lines
- 2 24-bit, 20 MHz counters/timers

The Simulink Real-Time 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

See Also

External Websites

www.ni.com

National Instruments PCI/PXI-6713 Analog Output (D/A)

PCI/PXI-6713 Analog Output block

Library

Simulink Real-Time Library for National Instruments

Note

The analog output range of this board is set -10 volts to +10 volts.

Scaling Input to Output

I/O Module 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 arbitrary 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 if this 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 replicated over the channel vector. If you specify a value of `1`, the corresponding channel is reset to the value specified in the initial value vector. If you specify a value of `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 replicated as the initial value over the channel vector. 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 (`-1` means sample time is inherited).

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

External Websites

www.ni.com

National Instruments PCI/PXI-6713 Digital Input

PCI/PXI-6713 Digital Input

Library

Simulink Real-Time Library for National Instruments

Scaling Input to Output

I/O Module 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 block. This driver allows the selection of individual digital input lines in arbitrary 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 if this board manufacturer starts numbering the lines with 0.

Sample time

Enter the base sample time or a multiple of the base sample time (`-1` means sample time is inherited).

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

External Websites

www.ni.com

National Instruments PCI/PXI-6713 Digital Output

PCI/PXI-6713 Digital Output block

Library

Simulink Real-Time Library for National Instruments

Scaling Input to Output

I/O Module 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 arbitrary 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 if this 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 replicated over the channel vector. If you specify a value of `1`, the corresponding channel is reset to the value specified in the initial value vector.

If you specify a value of `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 replicated as the initial value over the channel vector. 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 (`-1` means sample time is inherited).

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

External Websites

www.ni.com

National Instruments PCI/PXI-6713 Pulse Generation

PCI/PXI-6713 Pulse Generation block

Library

Simulink Real-Time Library for National Instruments

Note

The input to the block is a two-element vector. The first element is the number of counts (of the 20 MHz source clock) that the counter maintains at a low level. 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

Enter the base sample time or a multiple of the base sample time (`-1` means sample time is inherited).

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

External Websites

www.ni.com

National Instruments PCI/PXI-6713 Pulse Width/ Period Measurement

PCI/PXI-6713 Pulse Width/Period Measurement block

Library

Simulink Real-Time 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** interact. 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** interact. 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.

Sample time

Enter the base sample time or a multiple of the base sample time (`-1` means sample time is inherited).

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

External Websites

www.ni.com

National Instruments PCI/PXI-6731

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 that provides:

- 4 analog output (D/A) channels (16-bit)
- 8 digital input and output lines
- 2 24-bit, 20 MHz counters/timers

The Simulink Real-Time 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/PXI
Multiple block instance support	D/A: No, Digital I/O: Yes
Multiple board support	Yes

See Also

External Websites

www.ni.com

National Instruments PCI/PXI-6731 Analog Output (D/A)

PCI/PXI-6731 Analog Output block

Library

Simulink Real-Time Library for National Instruments

Note

The analog output range of this board is set -10 volts to +10 volts.

Scaling Input to Output

I/O Module Output	Block Input Data Type	Scaling
Volts	Double	1

Block Parameters

Channel vector

Enter numbers between 1 and 4. This driver allows the selection of individual D/A channels in arbitrary 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 if this 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 replicated over the channel vector. If you specify a value of `1`, the corresponding channel is reset to the value specified in the initial value vector. If you specify a value of `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 replicated as the initial value over the channel vector. 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 (`-1` means sample time is inherited).

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

External Websites

www.ni.com

National Instruments PCI/PXI-6731 Digital Input

PCI/PXI-6731 Digital Input block

Library

Simulink Real-Time Library for National Instruments

Scaling Input to Output

I/O Module 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 block. This driver allows the selection of individual digital input lines in arbitrary 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 if this board manufacturer starts numbering the lines with 0.

Sample time

Enter the base sample time or a multiple of the base sample time (`-1` means sample time is inherited).

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

External Websites

www.ni.com

National Instruments PCI/PXI-6731 Digital Output

PCI/PXI-6731 Digital Output

Library

Simulink Real-Time Library for National Instruments

Scaling Input to Output

I/O Module 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 arbitrary 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 if this 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 replicated over the channel vector. If you specify a value of `1`, the corresponding channel is reset to the value specified in the initial value vector.

If you specify a value of `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 replicated as the initial value over the channel vector. 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 (`-1` means sample time is inherited).

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

External Websites

www.ni.com

National Instruments PCI/PXI-6731 Pulse Generation

PCI/PXI-6731 Pulse Generation block

Library

Simulink Real-Time Library for National Instruments

Note

The input to the block is a two-element vector. The first element is the number of counts (of the 20 MHz source clock) that the counter maintains at a low level. 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

Enter the base sample time or a multiple of the base sample time (`-1` means sample time is inherited).

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

External Websites

www.ni.com

National Instruments PCI/PXI-6731 Pulse Width/ Period Measurement

PCI/PXI-6731 Pulse Width/Period Measurement block

Library

Simulink Real-Time 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** interact. 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** interact. 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.

Sample time

Enter the base sample time or a multiple of the base sample time (`-1` means sample time is inherited).

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

External Websites

www.ni.com

National Instruments PCI/PXI-6733

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 that provides:

- 8 analog output (D/A) channels (16-bit)
- 8 digital input lines
- 8 digital output lines
- 2 24-bit counters

The Simulink Real-Time 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/PXI
Multiple block instance support	D/A: No, Digital I/O: Yes
Multiple board support	Yes

See Also

External Websites

www.ni.com

National Instruments PCI/PXI-6733 Analog Output (D/A)

PCI/PXI-6733 Analog Output block

Library

Simulink Real-Time Library for National Instruments

Note

The analog output range of this board is set -10 volts to +10 volts.

Scaling Input to Output

I/O Module 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 arbitrary 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 if this 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 replicated over the channel vector. If you specify a value of `1`, the corresponding channel is reset to the value specified in the initial value vector. If you specify a value of `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 replicated as the initial value over the channel vector. 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 (`-1` means sample time is inherited).

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

External Websites

www.ni.com

National Instruments PCI/PXI-6733 Digital Input

PCI/PXI-6733 Digital Input block

Library

Simulink Real-Time Library for National Instruments

Scaling Input to Output

I/O Module 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 block. This driver allows the selection of individual digital input lines in arbitrary 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 if this board manufacturer starts numbering the lines with 0.

Sample time

Enter the base sample time or a multiple of the base sample time (`-1` means sample time is inherited).

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

External Websites

www.ni.com

National Instruments PCI/PXI-6733 Digital Output

PCI/PXI-6733 Digital Output

Library

Simulink Real-Time Library for National Instruments

Scaling Input to Output

I/O Module 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 arbitrary 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 if this 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 replicated over the channel vector. If you specify a value of `1`, the corresponding channel is reset to the value specified in the initial value vector.

If you specify a value of `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 replicated as the initial value over the channel vector. 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 (`-1` means sample time is inherited).

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

External Websites

www.ni.com

National Instruments PCI/PXI-6733 Pulse Generation

PCI/PXI-6733 Pulse Generation block

Library

Simulink Real-Time Library for National Instruments

Note

The input to the block is a two-element vector. The first element is the number of counts (of the 20 MHz source clock) that the counter maintains at a low level. 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

Enter the base sample time or a multiple of the base sample time (`-1` means sample time is inherited).

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

External Websites

www.ni.com

National Instruments PCI/PXI-6733 Pulse Width/ Period Measurement

PCI/PXI-6733 Pulse Width/Period Measurement block

Library

Simulink Real-Time 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** interact. 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** interact. 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.

Sample time

Enter the base sample time or a multiple of the base sample time (`-1` means sample time is inherited).

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

External Websites

www.ni.com

National Instruments PCI-DIO-96

Support for National Instruments PCI-DIO-96 board

Board

National Instruments PCI-DIO-96

General Description

The PC-DIO-96 provides 96 digital input and output lines.

The Simulink Real-Time block library supports this board with these driver blocks:

- National Instruments PCI-DIO-96 Digital Input
- National Instruments PCI-DIO-96 Digital Output

Board Characteristics

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

See Also

External Websites

www.ni.com

National Instruments PCI-DIO-96 Digital Input

PCI-DIO-96 Digital Input block

Library

Simulink Real-Time Library for National Instruments

Note

The PC-DIO-96 has four 8255 chips 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 of Input to Output

I/O Module 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 arbitrary 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 this board manufacturer starts numbering the lines with 0.

Port

From the list, choose either `A`, `B`, or `C`. The I/O board has four 8255 chips with 3 ports. The **Port** parameter defines which port of the 8255 chip is used for this driver block. Each port has a maximum or 8 digital lines that can be configured as inputs or outputs 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 the base sample time or a multiple of the base sample time (`-1` means sample time is inherited).

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

External Websites

www.ni.com

National Instruments PCI-DIO-96 Digital Output

PCI-DIO-96 Digital Output block

Library

Simulink Real-Time Library for National Instruments

Note

The PC-DIO24 has four 8255 chips 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.

Scaling of Input to Output

I/O Module 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 arbitrary 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 this 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 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 replicated over the channel vector. If you specify a value of `1`, the corresponding channel is reset to the value specified in the initial value vector. If you specify a value of `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 replicated as the initial value over the channel vector. 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 the base sample time or a multiple of the base sample time (`-1` means sample time is inherited).

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

External Websites

www.ni.com

National Instruments PXI-6040E

Support for National Instruments PXI-6040E board

Board

National Instruments PXI-6040E

General Description

The PXI-6040E provides:

- 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
- 2 counter/timers (24-bit) with a maximum source clock rate of 20 MHz

The Simulink Real-Time 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

See Also

External Websites

www.ni.com

National Instruments PXI-6040E Analog Input (A/D)

PXI-6040E Analog Input block

Library

Simulink Real-Time Library for National Instruments

Scaling Input to Output

I/O Module Input	Block Output Data Type	Scaling
Volts	Double	1

Block Parameters

Channel vector

Enter numbers between 1 and 16. This driver allows you to enter channel numbers in arbitrary order.

For example, to use the first, second, and fifth channels, enter

```
[1,2,5]
```

Number the channels beginning with 1 even if this 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

Input Range (V)	Range Code
-5 to +5	-5
-2 to +2	-2
-1 to +1	-1
-0.5 to +0.5	-0.5
-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 volts to +10 volts and the second and fifth channels are 0 volts 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.

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.

Coupling	Coupling Code	Description
NRSE	1	Analog input line connected to the positive input of the PGIA. Analog input sense (AISENSE) connected to the negative input of the PGIA. See the board manual.
DIFF	2	First analog input line connected to the positive input of the PGIA. Second analog input line connected to the negative input of the PGIA. See the board manual.

For example, if the first and second channels are single input and the fifth channel is a differential input, enter

```
[0,0,2]
```

The driver selects a second differential input 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

Enter the base sample time or a multiple of the base sample time (`-1` means sample time is inherited).

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

External Websites

www.ni.com

National Instruments PXI-6040E Analog Output (D/A)

PXI-6040E Analog Output block

Library

Simulink Real-Time Library for National Instruments

Scaling of Input to Output

I/O Module 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 arbitrary 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 this board manufacturer starts numbering the channels with 0.

Range vector

Enter a range code for each of the channels in the channel vector. The range vector must be the same length as the channel vector. This board allows the range of each channel to be different.

The following table is a list of the ranges for this driver and the corresponding range codes.

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

For example, if the first channel is -10 volts 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 replicated over the channel vector. If you specify a value of `1`, the corresponding channel is reset to the value specified in the initial value vector. If you specify a value of `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 replicated as the initial value over the channel vector. 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 (`-1` means sample time is inherited).

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

External Websites

www.ni.com

National Instruments PXI-6040E Digital Input

PXI-6040E Digital Input block

Library

Simulink Real-Time Library for National Instruments

Scaling Input to Output

I/O Module 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 arbitrary 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 this board manufacturer starts numbering the lines with 0.

Sample time

Enter the base sample time or a multiple of the base sample time (`-1` means sample time is inherited).

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

External Websites

www.ni.com

National Instruments PXI-6040E Digital Output

PXI-6040E Digital Output block

Library

Simulink Real-Time Library for National Instruments

Scaling Input to Output

I/O Module Output	Block Input 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 arbitrary 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 this 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 replicated over the channel vector. If you specify a value of `1`, the corresponding channel is reset to the value specified in the initial value vector. If you specify a value of `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 replicated as the initial value over the channel vector. 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 (`-1` means sample time is inherited).

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

External Websites

www.ni.com

National Instruments PXI-6040E Pulse Generation

PXI-6040E Pulse Generation block

Library

Simulink Real-Time Library for National Instruments

Note

The input to the block is a two-element vector. The first element is the number of counts (of the 20 MHz source clock) that the counter maintains at a low level. 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

Enter the base sample time or a multiple of the base sample time (`-1` means sample time is inherited).

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

External Websites

www.ni.com

National Instruments PXI-6040E Pulse Width/Period Measurement

PXI-6040E Pulse Width/Period Measurement block

Library

Simulink Real-Time 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 can experience false triggering. Use signal voltage levels of 0-5 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** interact. 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** interact. 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

Measurement Objective	Trigger Mode	Polarity	Output
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

Enter the base sample time or a multiple of the base sample time (`-1` means sample time is inherited).

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

External Websites

www.ni.com

National Instruments PXI-6070E

Support for National Instruments PXI-6070E board

Board

National Instruments PXI-6070E

General Description

The PXI-6070E provides:

- 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
- 2 counter/timers (24-bit) with a maximum source clock rate of 20 MHz

The Simulink Real-Time 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

Board Characteristics

Board name	PXI-6070E
Manufacturer	National Instruments
Bus type	PXI (Compact PCI)

Multiple block instance support	A/D: No, D/A: No, Digital I/O: Yes; Pulse Width/Period measurement: Yes; Pulse Train Generation: No
Multiple board support	Yes

See Also

External Websites

www.ni.com

National Instruments PXI-6070E Analog Input (A/D)

PXI-6070E Analog Input block

Library

Simulink Real-Time Library for National Instruments

Scaling Input to Output

I/O Module Input	Block Output Data Type	Scaling
Volts	Double	1

Block Parameters

Channel vector

Enter numbers between 1 and 16. This driver allows you to enter channel numbers in arbitrary order.

For example, to use the first, second, and fifth channels, enter

```
[1,2,5]
```

Number the channels beginning with 1 even if this 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

Input Range (V)	Range Code
-5 to +5	-5
-2 to +2	-2
-1 to +1	-1
-0.5 to +0.5	-0.5
-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 volts to +10 volts and the second and fifth channels are 0 volts 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.

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.

Coupling	Coupling Code	Description
NRSE	1	Analog input line connected to the positive input of the PGIA. Analog input sense (AISENSE) connected to the negative input of the PGIA. See the board manual.
DIFF	2	First analog input line connected to the positive input of the PGIA. Second analog input line connected to the negative input of the PGIA. See the board manual.

For example, if the first and second channels are single input and the fifth channel is a differential input, enter

```
[0,0,2]
```

The driver selects a second differential input 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

Enter the base sample time or a multiple of the base sample time (`-1` means sample time is inherited).

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

External Websites

www.ni.com

National Instruments PXI-6070E Analog Output (D/A)

PXI-6070E Analog Output block

Library

Simulink Real-Time Library for National Instruments

Scaling Input to Output

I/O Module 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 arbitrary 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 this board manufacturer starts numbering the channels with 0.

Range vector

Enter a range code for each of the channels in the channel vector. The range vector must be the same length as the channel vector. This board allows the range of each channel to be different.

The following table is a list of the ranges for this driver and the corresponding range codes.

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

For example, if the first channel is -10 volts 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 replicated over the channel vector. If you specify a value of `1`, the corresponding channel is reset to the value specified in the initial value vector. If you specify a value of `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 replicated as the initial value over the channel vector. 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 (`-1` means sample time is inherited).

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

External Websites

www.ni.com

National Instruments PXI-6070E Digital Input

PXI-6070E Digital Input block

Library

Simulink Real-Time Library for National Instruments

Scaling Input to Output

I/O Module 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 arbitrary 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 this board manufacturer starts numbering the lines with 0.

Sample time

Enter the base sample time or a multiple of the base sample time (`-1` means sample time is inherited).

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

External Websites

www.ni.com

National Instruments PXI-6070E Digital Output

PXI-6070E Digital Output block

Library

Simulink Real-Time Library for National Instruments

Scaling Input to Output

I/O Module 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 arbitrary 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 this 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 replicated over the channel vector. If you specify a value of `1`, the corresponding channel is reset to the value specified in the initial value vector. If you specify a value of `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 replicated as the initial value over the channel vector. 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 (`-1` means sample time is inherited).

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

External Websites

www.ni.com

National Instruments PXI-6070E Pulse Generation

PXI-6070E Pulse Generation block

Library

Simulink Real-Time Library for National Instruments

Note

The input to the block is a two-element vector. The first element is the number of counts (of the 20 MHz source clock) that the counter maintains at a low level. 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

Enter the base sample time or a multiple of the base sample time (`-1` means sample time is inherited).

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

External Websites

www.ni.com

National Instruments PXI-6070E Pulse Width/Period Measurement

PXI-6070E Pulse Width/Period Measurement block

Library

Simulink Real-Time 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 can experience false triggering. Use signal voltage levels of 0-5 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** interact. 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** interact. 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

Measurement Objective	Trigger Mode	Polarity	Output
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

Enter the base sample time or a multiple of the base sample time (`-1` means sample time is inherited).

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

External Websites

www.ni.com

National Instruments PXI-6071E

Support for National Instruments PXI-6071E board

Board

National Instruments PXI-6071E

General Description

The PXI-6071E provides:

- 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
- 2 counter/timers (24-bit) with a maximum source clock rate of 20 MHz

The Simulink Real-Time 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	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

See Also

External Websites

www.ni.com

National Instruments PXI-6071E Analog Input (A/D)

PXI-6071E Analog Input block

Library

Simulink Real-Time Library for National Instruments

Scaling Input to Output

I/O Module Input	Block Output Data Type	Scaling
Volts	Double	1

Block Parameters

Channel vector

Enter numbers between 1 and 64. This driver allows you to enter channel numbers in arbitrary order.

For example, to use the first, second, and fifth channels, enter

```
[1,2,5]
```

Number the channels beginning with 1 even if this 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

Input Range (V)	Range Code
-5 to +5	-5
-2 to +2	-2
-1 to +1	-1
-0.5 to +0.5	-0.5
-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 volts 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.

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.

Coupling	Coupling Code	Description
NRSE	1	Analog input line connected to the positive input of the PGIA. Analog input sense (AISENSE) connected to the negative input of the PGIA. See the board manual.
DIFF	2	First analog input line connected to the positive input of the PGIA. Second analog input line connected to the negative input of the PGIA. See the board manual.

For example, if the first and second channels are single input and the fifth channel is a differential input, enter

```
[0,0,2]
```

The driver selects a second differential input 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

Enter the base sample time or a multiple of the base sample time (`-1` means sample time is inherited).

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

External Websites

www.ni.com

National Instruments PXI-6071E Analog Output (D/A)

PXI-6071E Analog Output block

Library

Simulink Real-Time Library for National Instruments

Scaling Input to Output

I/O Module 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 arbitrary 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 this board manufacturer starts numbering the channels with 0.

Range vector

Enter a range code for each of the channels in the channel vector. The range vector must be the same length as the channel vector. This board allows the range of each channel to be different.

The following table is a list of the ranges for this driver and the corresponding range codes.

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

For example, if the first channel is -10 volts 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 replicated over the channel vector. If you specify a value of `1`, the corresponding channel is reset to the value specified in the initial value vector. If you specify a value of `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 replicated as the initial value over the channel vector. 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 (`-1` means sample time is inherited).

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

External Websites

www.ni.com

National Instruments PXI-6071E Digital Input

PXI-6071E Digital Input block

Library

Simulink Real-Time Library for National Instruments

Scaling Input to Output

I/O Module Input	Block Output Data Type	Scaling
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 arbitrary 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 this board manufacturer starts numbering the lines with 0.

Sample time

Enter the base sample time or a multiple of the base sample time (`-1` means sample time is inherited).

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

External Websites

www.ni.com

National Instruments PXI-6071E Digital Output

PXI-6071E Digital Output block

Library

Simulink Real-Time Library for National Instruments

Scaling Input to Output

I/O Module 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 arbitrary 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 this 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 replicated over the channel vector. If you specify a value of `1`, the corresponding channel is reset to the value specified in the initial value vector. If you specify a value of `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 replicated as the initial value over the channel vector. 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 (`-1` means sample time is inherited).

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

External Websites

www.ni.com

National Instruments PXI-6071E Pulse Generation

PXI-6071E Pulse Generation block

Library

Simulink Real-Time Library for National Instruments

Note

The input to the block is a two-element vector. The first element is the number of counts (of the 20 MHz source clock) that the counter maintains at a low level. 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

Enter the base sample time or a multiple of the base sample time (`-1` means sample time is inherited).

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

External Websites

www.ni.com

National Instruments PXI-6071E Pulse Width/Period Measurement

PXI-6071E Pulse Width/Period Measurement block

Library

Simulink Real-Time 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 can experience false triggering. Use signal voltage levels of 0-5 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** interact. 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** interact. 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

Measurement Objective	Trigger Mode	Polarity	Output
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

Enter the base sample time or a multiple of the base sample time (`-1` means sample time is inherited).

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

External Websites

www.ni.com

National Instruments PXI-6508

Support for National Instruments PXI-6508 board

Board

National Instruments PXI-6508

General Description

The PXI-6508 provides 96 digital input and output lines.

The Simulink Real-Time block library supports this board with these driver blocks:

- National Instruments PXI-6508 Digital Input
- National Instruments PXI-6508 Digital Output

Board Characteristics

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

See Also

External Websites

www.ni.com

National Instruments PXI-6508 Digital Input

PXI-6508 Digital Input block

Library

Simulink Real-Time 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

I/O Module 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 arbitrary 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 this 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 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 the base sample time or a multiple of the base sample time (`-1` means sample time is inherited).

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

External Websites

www.ni.com

National Instruments PXI-6508 Digital Output

PXI-6508 Digital Output block

Library

Simulink Real-Time 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

I/O Module 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 arbitrary 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 this 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 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 replicated over the channel vector. If you specify a value of `1`, the corresponding channel is reset to the value specified in the initial value vector. If you specify a value of `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 replicated as the initial value over the channel vector. 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 the base sample time or a multiple of the base sample time (`-1` means sample time is inherited).

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

External Websites

www.ni.com

National Instruments PXI-6527

Support for National Instruments PXI-6527 board

Board

National Instruments PXI-6527

General Description

The PXI-6527 provides:

- 24 filtered digital input lines
- 24 digital output lines

You can individually turn input filtering on or off for each of the 24 input lines, but the same filter time applies to each line.

The Simulink Real-Time 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

See Also

External Websites

www.ni.com

National Instruments PXI-6527 Digital Input

PXI-6527 Digital Input block

Library

Simulink Real-Time 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

I/O Module 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 arbitrary 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 this 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.

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 scaler 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 filter uses to determine a stable pulse. If the input pule is shorter then this interval, it is ignored. The same filter interval applies to each filtered input, and if you filter on more than one digital input block for this board, you must enter the same **Filter interval** for each block.

A reasonable value for the **Filter interval** would be 200 to 300 us.

Sample time

Enter the base sample time or a multiple of the base sample time (`-1` means sample time is inherited).

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

External Websites

www.ni.com

National Instruments PXI-6527 Digital Output

PXI-6527 Digital Output block

Library

Simulink Real-Time 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

I/O Module Output	Block Output Data Type	Scaling
OPTO	Double	<0.5 = OPTO is OFF, current not 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 arbitrary 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 this 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 or 8 digital output lines. In each case, one block is required for each port.

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 replicated over the channel vector. If you specify a value of 1, the corresponding channel is reset to the value specified in the initial value vector. If you specify a value of 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 replicated as the initial value over the channel vector. 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 (-1 means sample time is inherited).

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter -1 to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

External Websites

www.ni.com

National Instruments PXI-6704

Support for National Instruments PXI-6704 board

Board

National Instruments PXI-6704

General Description

The PXI-6704 provides:

- 16 voltage outputs
- 16 current outputs
- 8 digital I/O lines

The Simulink Real-Time block library supports this board with this driver block:

- National Instruments PXI-6704 Analog Output (D/A)

Board Characteristics

Board name	PXI-6704
Manufacturer	National Instruments
Bus type	PXI
Access method	Memory mapped
Multiple block instance support	No
Multiple board support	Yes

See Also

External Websites

www.ni.com

National Instruments PXI-6704 Analog Output (D/A)

PXI-6704 Analog Output block

Library

Simulink Real-Time Library for National Instruments

Scaling Input to Output

I/O Module Output	Block Input Data Type	Scaling
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 this board manufacturer starts numbering the channels with 0.

Sample time

Enter the base sample time or a multiple of the base sample time (`-1` means sample time is inherited).

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

External Websites

www.ni.com

NAII, Quanser, Real Time Devices

North Atlantic Industries, Inc.

This topic describes the North Atlantic Industries, Inc. (NAII, formerly Apex) I/O boards supported by the Simulink Real-Time product (`www.naii.com`).

NAII (Apex) 73LD3

Support for NAII (Apex) 73LD3 positioning sensor I/O board

Board

NAII (Apex) 73LD3

General Description

The NAII (Apex) 73LD3provides 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 two, four, or six channels. The board can 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 Simulink Real-Time block library supports this board with this driver:

- NAII 73LD3 LVDT/RVDT Converter

The Simulink Real-Time product does not support the digital I/O functionality of this board.

Board Characteristics

Board name	73LD3
Manufacturer	NAII
Bus type	ISA
Access method	I/O mapped
Multiple block instance support	No
Multiple board support	Yes

See Also

External Websites

www.naii.com

NAII 73LD3 LVDT/RVDT Converter

73LD3 LVDT/RVDT Converter block

Library

Simulink Real-Time Library for NAII

Scaling Input to Output

Hardware Input	Block Output Data Type	Scaling
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 arbitrary order. Number them beginning with 1 even if this 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 two-wire systems, enter the transformation ratio so that the full range of motion results in a full scale output signal from -32768 through 32767. For three- or four-wire systems, a setting of 65535 results in a full scale output signal. You can choose this scale factor for each channel independently. If you enter a scalar value, 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 displays one input port for each output channel selected. These input ports are 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. If the board does not provide a reference/excitation, the test status and reference status do nothing.

  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 (`-1` means sample time is inherited).

Base address

Enter the base address of the board. This entry must correspond to the DIP switch setting on the board. For example, if the DIP switch setting is 300 (hexadecimal), enter `0x300`.

See Also

External Websites

www.naii.com

NAII (Apex) 73SD3

Support for NAII (Apex) 73SD3 positioning sensor I/O board

Board

NAII (Apex) 73SD3

General Description

The NAII (Apex) 73SD3provides up to six Synchro/Resolver-to-digital channels. The capabilities of this board vary according to its part number designation. There can be two, four, or six channels. The board can 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 Simulink Real-Time block library supports this board with one driver block:

- NAII 73SD3 Synchro/Resolver

Board Characteristics

Board name	73SD3
Manufacturer	NAII
Bus type	ISA
Access method	Memory mapped
Multiple block instance support	No
Multiple board support	Yes

See Also

External Websites

www.naii.com

NAII 73SD3 Synchro/Resolver

NAII 73SD3 Synchro/Resolver block

Library

Simulink Real-Time Library for NAII

Scaling Input to Output

Hardware Input	Block Output Data Type	Scaling
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 arbitrary 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:

- 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 do nothing.

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. The 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 (`-1` means sample time is inherited).

Base address

Enter the base address of the board. This entry must correspond to the DIP switch setting on the board. For example, if the DIP switch setting is 300 (hexadecimal), enter `0x300`.

See Also

External Websites

www.naii.com

NAII (Apex) 76LD1

Support for NAII (Apex) 76LD1 positioning sensor I/O board

Board

NAII (Apex) 76LD1

General Description

The NAII (Apex) 76LD1provides 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 can 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 Simulink Real-Time block library supports this board with one driver block:

- NAII 76LD1 L/D

Board Characteristics

Board name	76LD1
Manufacturer	NAII
Bus type	PCI
Access method	Memory mapped
Multiple block instance support	No
Multiple board support	Yes

See Also

External Websites

www.naii.com

NAII 76LD1 L/D

NAII 76LD1 L/D block

Library

Simulink Real-Time Library for NAII

Scaling Input to Output

Hardware Input	Block Output Data Type	Scaling
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 arbitrary 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 do nothing.

  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. The 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 (`-1` means sample time is inherited).

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

External Websites

www.naii.com

NAII (Apex) 76CL1

Support for NAII (Apex) 76CL1 positioning sensor I/O board

Board

NAII (Apex) 76CL1

Library

Simulink Real-Time Library for NAII

Description

The NAII (Apex) 76CL1 I/O board provides:

- Up to 8 2-wire or up to four 3-wire or 4-wire LVDT/RVDT measurement (input) channels
- 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 can 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 Simulink Real-Time block library supports this board with two driver blocks:

- NAII 76CL1 L/D
- NAII 76CL1 D/L

Board Characteristics

Board name	76Cl1
Manufacturer	NAII

Bus type	PCI
Access method	Memory mapped
Multiple block instance support	No
Multiple board support	Yes

See Also

External Websites

www.naii.com

NAII 76CL1 L/D

NAII 76CL1 L/D block

Library

Simulink Real-Time Library for NAII

Scaling Input to Output

Hardware Input	Block Output Data Type	Scaling
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 arbitrary 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 do nothing.

  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. The 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 (`-1` means sample time is inherited).

PCI slot

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

External Websites

www.naii.com

NAII 76CL1 D/L

NAII 76CL1 D/L block

Library

Simulink Real-Time Library for NAII

Scaling Input to Output

Hardware Input	Block Output Data Type	Scaling
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 arbitrary 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 the 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

The reset vector controls the behavior of the channel at model termination. Enter a scalar or a vector that is the same length as the channel vector. If you specify a scalar

value, that setting is replicated over the channel vector. If you specify a value of 1, the corresponding channel is reset to the value specified in the initial value 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 replicated as the initial value over the channel vector. 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 (`-1` means sample time is inherited).

PCI slot

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

External Websites

www.naii.com

NAII (Apex) 76CS1

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
- Up to 6 stimulus (output, D/S) channels

The capabilities of this board vary according to its part number designation. The board can contain an on-board reference. If this reference is present, the supported frequency range is board dependent.

The Simulink Real-Time block library supports this board with two driver blocks:

- NAII 76CS1 S/D
- NAII 76CS1 D/S

Board Characteristics

Board name	76CS1
Manufacturer	NAII
Bus type	PCI
Access method	Memory mapped
Multiple block instance support	No
Multiple board support	Yes

See Also

External Websites

www.naii.com

NAII 76CS1 S/D

NAII 76CS1 S/D block

Library

Simulink Real-Time Library for NAII

Scaling Input to Output

Hardware Input	Block Output Data Type	Scaling
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 arbitrary 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

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 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 do nothing.

  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. The 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 (`-1` means sample time is inherited).

PCI slot

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

External Websites

www.naii.com

NAII 76CS1 D/S

NAII 76CS1 D/S block

Library

Simulink Real-Time Library for NAII

Scaling Input to Output

Hardware Input	Block Output Data Type	Scaling
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 arbitrary order. The maximum allowable channel number can be 0, 2, 4, or 6 depending on the board type.

Reset vector

The reset vector controls the behavior of the channel at model termination. Enter a scalar or a vector that is the same length as the channel vector. If you specify a scalar value, that setting is replicated over the channel vector. If you specify a value of `1`, the corresponding channel is reset to the value specified in the initial value 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 replicated as the initial value over the channel

vector. 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 (`-1` means sample time is inherited).

PCI slot

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

External Websites

www.naii.com

Quanser

This topic describes the Quanser I/O board supported by the Simulink Real-Time product (`www.quanser.com`).

Quanser Q4

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 that provides:

- 4 14-bit singled-ended A/D converters
- 4 12-bit D/A voltage outputs
- 16 digital communication channels (each of which can be used for digital input or digital output)
- 4 24-bit encoder inputs
- 2 32-bit counter/timers

The Simulink Real-Time 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

See Also

External Websites

www.quanser.com

Quanser Q4 Analog Input

Q4 Analog Input block

Library

Simulink Real-Time 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 arbitrary order. For example, to use the first, second and fourth channels, enter

```
[1,2,4]
```

Number the channels beginning with 1 even if this 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

Enter the base sample time or a multiple of the base sample time (`-1` means sample time is inherited).

PCI slot

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

External Websites

www.quanser.com

Quanser Q4 Analog Output

Q4 Analog Output block

Library

Simulink Real-Time 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 arbitrary order. For example, to use the first, second, and fourth channels, enter

```
[1,2,4]
```

Number the channels beginning with 1 even if this 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 replicated over 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

Input Range (V)	Range Code
-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 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 loss of performance.

Reset vector

The reset vector controls the behavior of the channel at model termination. Enter a scalar or a vector that is the same length as the channel vector. If you specify a scalar value, that setting is replicated over the channel vector. If you specify a value of `1`, the corresponding channel is reset to the value specified in the initial value vector. If you specify a value of `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 replicated as the initial value over the channel vector. 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 (`-1` means sample time is inherited).

PCI slot

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

External Websites

www.quanser.com

Quanser Q4 Digital Input

Q4 Digital Input block

Library

Simulink Real-Time 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 arbitrary 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 this 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

Enter the base sample time or a multiple of the base sample time (`-1` means sample time is inherited).

PCI slot

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

External Websites

www.quanser.com

Quanser Q4 Digital Output

Q4 Digital Output block

Library

Simulink Real-Time 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 arbitrary 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 this 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 replicated over the channel vector. If you specify a value of `1`, the corresponding channel is reset to the value specified in the initial value vector. If you specify a value of `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 replicated as the initial value over the channel vector. 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 (`-1` means sample time is inherited).

PCI slot

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

External Websites

www.quanser.com

Quanser Q4 Incremental Encoder

Q4 Incremental Encoder block

Library

Simulink Real-Time 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 arbitrary order. For example, to use the first two odd encoder input channels, enter

```
[1,3]
```

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

Note: If you specify four channels or fewer, 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 replicated as initial count value over the channel vector.

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 replicated over the channel vector. 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 replicated over the channel vector. 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 replicated over the channel vector. The allowable entries are

- 0 — Normal. The counter wraps from 0 to 0xFFFFFF when decremented and from 0xFFFFFF to 0 when incremented.
- 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 replicated over the channel vector. 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 replicated over the channel vector. 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 replicated over the channel vector. 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

Enter the base sample time or a multiple of the base sample time (`-1` means sample time is inherited).

PCI slot

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

External Websites

www.quanser.com

Quanser Q4 Counter

Q4 Counter block

Library

Simulink Real-Time 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 arbitrary 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 replicated over the channel vector. 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 replicated over the channel vector. The allowable entries are

- 0 — Do not display the input arm input. The channel is already 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 replicated over the channel vector. If you specify a value of 1, the corresponding channel is reset to the value specified in the 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 replicated over the channel vector.

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 replicated over the channel vector.

Sample time

Enter the base sample time or a multiple of the base sample time (`-1` means sample time is inherited).

PCI slot

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

External Websites

www.quanser.com

Quanser Q8

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 that provides:

- 8 14-bit singled-ended A/D converters (supported by two A/D chips with four channels per chip)
- 8 12-bit D/A voltage outputs
- 32 digital communication channels (each of which can be used for digital input or digital output)
- 8 24-bit encoder inputs
- 2 32-bit counter/timers

The Simulink Real-Time 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

See Also

External Websites

www.quanser.com

Quanser Q8 Analog Input

Q8 Analog Input block

Library

Simulink Real-Time 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 arbitrary order. For example, to use the first, second and fifth channels, enter

```
[1,2,5]
```

Number the channels beginning with 1 even if this 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

Enter the base sample time or a multiple of the base sample time (`-1` means sample time is inherited).

PCI slot

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

External Websites

www.quanser.com

Quanser Q8 Analog Output

Q8 Analog Output block

Library

Simulink Real-Time 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 arbitrary order. For example, to use the first, second and fifth channels, enter

```
[1,2,5]
```

Number the channels beginning with 1 even if this 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 replicated over 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

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 loss of performance.

Reset vector

The reset vector controls the behavior of the channel at model termination. Enter a scalar or a vector that is the same length as the channel vector. If you specify a scalar value, that setting is replicated over the channel vector. If you specify a value of `1`, the corresponding channel is reset to the value specified in the initial value vector. If you specify a value of `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 replicated as the initial value over the channel vector. 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 (`-1` means sample time is inherited).

PCI slot

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

External Websites

www.quanser.com

Quanser Q8 Digital Input

Q8 Digital Input block

Library

Simulink Real-Time 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 arbitrary 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 this 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

Enter the base sample time or a multiple of the base sample time (`-1` means sample time is inherited).

PCI slot

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

External Websites

www.quanser.com

Quanser Q8 Digital Output

Q8 Digital Output block

Library

Simulink Real-Time 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 arbitrary 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 this 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 replicated over the channel vector. If you specify a value of `1`, the corresponding channel is reset to the value specified in the initial value vector. If you specify a value of `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 replicated as the initial value over the channel vector. 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 (`-1` means sample time is inherited).

PCI slot

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

External Websites

www.quanser.com

Quanser Q8 Incremental Encoder

Q8 Incremental Encoder block

Library

Simulink Real-Time 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 arbitrary order. For example, to use the first three odd encoder input channels, enter

```
[1,3,5]
```

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

Note: If you specify four channels or fewer, 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 replicated as initial count value over the channel vector.

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 replicated over the channel vector. 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 replicated over the channel vector. 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 replicated over the channel vector. The allowable entries are

- 0 — Normal. The counter wraps from 0 to 0xFFFFFF when decremented and from 0xFFFFFF to 0 when incremented.
- 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 replicated over the channel vector. 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 replicated over the channel vector. 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 replicated over the channel vector. 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

Enter the base sample time or a multiple of the base sample time (`-1` means sample time is inherited).

PCI slot

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

External Websites

www.quanser.com

Quanser Q8 Counter

Q8 Counter block

Library

Simulink Real-Time 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 arbitrary 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 replicated over the channel vector. 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 replicated over the channel vector. The allowable entries are

- 0 — Do not display the input arm input. The channel is already 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 replicated over the channel vector. If you specify a value of 1, the corresponding channel is reset to the value specified in the 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 replicated over the channel vector.

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 replicated over the channel vector.

Sample time

Enter the base sample time or a multiple of the base sample time (`-1` means sample time is inherited).

PCI slot

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

External Websites

www.quanser.com

Real Time Devices

This topic describes Real Time Devices I/O boards supported by the Simulink Real-Time product (`www.rtd.com`).

Real Time Devices DM6420

Support for the RTD DM6420 I/O board.

Board

Real Time Devices DM6420

General Description

The DM6420 provides:

- 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
- 2 counter/timers (16-bit)

The Simulink Real-Time 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 Simulink Real-Time 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

See Also

External Websites

www.rtd.com

Real Time Devices DM6420 Analog Input (A/D)

DM6420 Analog Input block

Library

Simulink Real-Time Library for Real Time Devices

Scaling Input to Output

Hardware Input	Block Output Data Type	Scaling
Volts	Double	1

Block Parameters

Channel vector

Enter numbers between 1 and 16. This driver allows the selection of individual A/D channels in arbitrary 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 this 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.

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 ±5, ±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.

Gain	Range (V)
1	-10 to 10
2	-5 to +5

Gain	Range (V)
4	-2.5 to 2.5
8	-1.25 to 1.25

Notice that by increasing the gain code the voltage range is decreased. The gain divides the input voltage range.

For example, if the first channel has a gain code of 1 (10 volt range) and the second and fifth channels have a gain code of 2 (5 volt range), enter

```
[1,2,2]
```

Sample time

Enter the base sample time or a multiple of the base sample time (`-1` means sample time is inherited).

Base address

Enter the base address of the board. This entry must correspond to the DIP switch setting on the board. For example, if the DIP switch setting is 300 (hexadecimal), enter `0x300`.

See Also

External Websites

www.rtd.com

Real Time Devices DM6420 Analog Output (D/A)

DM6420 Analog Output block

Library

Simulink Real-Time 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 arbitrary 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 this board manufacturer starts numbering the channels with 0.

Range vector

Enter a range code for each of the channels entered in the channel vector. The range vector must be the same length as the channel vector. This driver allows a different range for each channel.

The following table is a list of the ranges for this driver and the corresponding range codes.

Input Range (V)	Range Code
-5 to +5	-5
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 base sample time or a multiple of the base sample time (`-1` means sample time is inherited).

Base address

Enter the base address of the board. This entry must correspond to the DIP switch setting on the board. For example, if the DIP switch setting is 300 (hexadecimal), enter `0x300`.

See Also

External Websites

www.rtd.com

Real Time Devices DM6420 Digital Input

DM6420 Digital Input block

Library

Simulink Real-Time Library for Real Time Devices

Note

The DAS1601/16 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	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 arbitrary 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 this 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 the base sample time or a multiple of the base sample time (`-1` means sample time is inherited).

Base address

Enter the base address of the board. This entry must correspond to the DIP switch setting on the board. For example, if the DIP switch setting is 300 (hexadecimal), enter `0x300`.

See Also

External Websites

www.rtd.com

Real Time Devices DM6420 Digital Output

DM6420 Digital Output block

Library

Simulink Real-Time Library for Real Time Devices

Note

The DAS1601/16 has an 8255 chip with two 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 arbitrary 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 this 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.

Sample time

Enter the base sample time or a multiple of the base sample time (`-1` means sample time is inherited).

Base address

Enter the base address of the board. This entry must correspond to the DIP switch setting on the board. For example, if the DIP switch setting is 300 (hexadecimal), enter `0x300`.

See Also

External Websites

www.rtd.com

Real Time Devices DM6430

Support for the RTD DM6430 I/O board.

Board

Real Time Devices DM6430

General Description

The DM6430 is an PC/104 I/O board that provides:

- 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
- 2 counter/timers (16-bit)

The Simulink Real-Time 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 Simulink Real-Time 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

See Also

External Websites

www.rtd.com

Real Time Devices DM6430 Analog Input (A/D)

DM6430 Analog Input block

Library

Simulink Real-Time Library for Real Time Devices

Scaling Input to Output

Hardware Input	Block Output Data Type	Scaling
Volts	Double	1

Block Parameters

Channel vector

Enter numbers between 1 and 16. This driver allows the selection of individual A/D channels in arbitrary 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 this 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.

Gain	Range (V)
1	-10 to 10

Gain	Range (V)
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]
```

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

Enter the base sample time or a multiple of the base sample time (`-1` means sample time is inherited).

Base address

Enter the base address of the board. This entry must correspond to the DIP switch setting on the board. For example, if the DIP switch setting is 300 (hexadecimal), enter `0x300`.

See Also

External Websites

www.rtd.com

Real Time Devices DM6430 Analog Output (D/A)

DM6430 Analog Output block

Library

Simulink Real-Time Library for Real Time Devices

Scaling Input to Output

Hardware Output	Block Input Data Type	Scaling
Volts	Double	1

Block Parameters

This board has two analog outputs (D/A) with a fixed range of -10 volts 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 arbitrary 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 this 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 replicated over the channel vector. If you specify a value of `1`, the corresponding channel is reset to the value specified in the initial value vector. If you specify a value of `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 replicated as the initial value over the channel vector. 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 (`-1` means sample time is inherited).

Base address

Enter the base address of the board. This entry must correspond to the DIP switch setting on the board. For example, if the DIP switch setting is 300 (hexadecimal), enter `0x300`.

See Also

External Websites

www.rtd.com

Real Time Devices DM6430 Digital Input

DM6430 Digital Input block

Library

Simulink Real-Time Library for Real Time Devices

Note

The DM6430 has an 8255 chip with two 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 arbitrary 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 this 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.

Sample time

Enter the base sample time or a multiple of the base sample time (`-1` means sample time is inherited).

Base address

Enter the base address of the board. This entry must correspond to the DIP switch setting on the board. For example, if the DIP switch setting is 300 (hexadecimal), enter `0x300`.

See Also

External Websites

www.rtd.com

Real Time Devices DM6430 Digital Output

DM6430 Digital Output

Library

Simulink Real-Time Library for Real Time Devices

Note

The DM6430 has an 8255 chip with two 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 arbitrary 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 this 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.

Sample time

Enter the base sample time or a multiple of the base sample time (`-1` means sample time is inherited).

Base address

Enter the base address of the board. This entry must correspond to the DIP switch setting on the board. For example, if the DIP switch setting is 300 (hexadecimal), enter `0x300`.

See Also

External Websites

www.rtd.com

Real Time Devices DM6604

Support for the RTD DM6604 I/O board.

Board

Real Time Devices DM6604

General Description

The DM6604 is an PC/104 I/O board that provides:

- 8 analog output (D/A) channels (12-bit)
- 24 digital I/O lines

The Simulink Real-Time 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

Board Characteristics

Board name	DM6604
Manufacturer	Real Time Devices
Bus type	PC/104
Access method	I/O mapped
Multiple block instance support	Yes
Multiple board support	Yes

See Also

External Websites

www.rtd.com

Real Time Devices DM6604 Analog Output (D/A)

DM6604 Analog Output block

Library

Simulink Real-Time 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 arbitrary 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 this 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

Input Range (V)	Range Code
-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

Enter the base sample time or a multiple of the base sample time (`-1` means sample time is inherited).

Base address

Enter the base address of the board. This entry must correspond to the DIP switch setting on the board. For example, if the DIP switch setting is 300 (hexadecimal), enter `0x300`.

See Also

External Websites

www.rtd.com

Real Time Devices DM6604 Digital Input

DM6604 Digital Input block

Library

Simulink Real-Time 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 arbitrary 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 this 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 the base sample time or a multiple of the base sample time (`-1` means sample time is inherited).

Base address

Enter the base address of the board. This entry must correspond to the DIP switch setting on the board. For example, if the DIP switch setting is 300 (hexadecimal), enter `0x300`.

See Also

External Websites

www.rtd.com

Real Time Devices DM6604 Digital Output

DM6604 Digital Output block

Library

Simulink Real-Time 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 arbitrary 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 this 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 replicated over the channel vector. If you specify a value of `1`, the corresponding channel is reset to the value specified in the initial value vector. If you specify a value of `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 replicated as the initial value over the channel vector. 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 (`-1` means sample time is inherited).

Base address

Enter the base address of the board. This entry must correspond to the DIP switch setting on the board. For example, if the DIP switch setting is 300 (hexadecimal), enter `0x300`.

See Also

External Websites

www.rtd.com

Real Time Devices DM6804

Support for the RTD DM6804 I/O board.

Board

Real Time Devices DM6804

General Description

The DM6804 is an PC/104 I/O board that provides:

- 24 digital I/O lines
- 1 8255 chip with 3 digital I/O ports

The Simulink Real-Time block library supports this board with these driver blocks:

- Real Time Devices DM6804 Digital Input
- Real Time Devices DM6804 Digital Output

Board Characteristics

Board name	DM6804
Manufacturer	Real Time Devices
Bus type	PC/104
Access method	I/O mapped
Multiple block instance support	Yes
Multiple board support	Yes

See Also

External Websites

www.rtd.com

Real Time Devices DM6804 Digital Input

DM6804 Digital Input block

Library

Simulink Real-Time Library for Real Time Devices

Note

The DM6804 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.

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 arbitrary 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 this 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 the base sample time or a multiple of the base sample time (`-1` means sample time is inherited).

Base address

Enter the base address of the board. This entry must correspond to the DIP switch setting on the board. For example, if the DIP switch setting is 300 (hexadecimal), enter `0x300`.

See Also

External Websites

www.rtd.com

Real Time Devices DM6804 Digital Output

DM6804 Digital Output block

Library

Simulink Real-Time Library for Real Time Devices

Note

The DM6804 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.

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 arbitrary 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 this 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 replicated over the channel vector. If you specify a value of 1, the corresponding channel is reset to the value specified in the initial value vector. If you specify a value of 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 replicated as the initial value over the channel vector. 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 (-1 means sample time is inherited).

Base address

Enter the base address of the board. This entry must correspond to the DIP switch setting on the board. For example, if the DIP switch setting is 300 (hexadecimal), enter 0x300.

See Also

External Websites

www.rtd.com

Real Time Devices DM6814

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 provides three I/O ports, with each port containing eight digital I/O lines.

The Simulink Real-Time 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 Simulink Real-Time software does not support the 12 digital input lines on this board.

Board Characteristics

Board name	DM6814
Manufacturer	Real Time Devices
Bus type	PC/104
Access method	I/O mapped
Multiple block instance support	Yes
Multiple board support	Yes

See Also

External Websites

www.rtd.com

Real Time Devices DM6814 Incremental Encoder

DM6814 Incremental Encoder block

Library

Simulink Real-Time Library for Real Time Devices

Scaling Input to Output

Hardware Output	Block Input Data Type	Scaling
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

Enter the base sample time or a multiple of the base sample time (`-1` means sample time is inherited).

Base address

Enter the base address of the board. This entry must correspond to the DIP switch setting on the board. For example, if the DIP switch setting is 300 (hexadecimal), enter `0x300`.

See Also

External Websites

www.rtd.com

Real Time Devices DM6814 Digital Input

DM6814 Digital Input block

Library

Simulink Real-Time 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 each block must use a unique set of channels.

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 arbitrary 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 this board manufacturer starts numbering the lines with 0.

Port

Select port `A`, `B`, or `C`.

Sample time

Enter the base sample time or a multiple of the base sample time (`-1` means sample time is inherited).

Base address

Enter the base address of the board. This entry must correspond to the DIP switch setting on the board. For example, if the DIP switch setting is 300 (hexadecimal), enter `0x300`.

See Also

External Websites

www.rtd.com

Real Time Devices DM6814 Digital Output

DM6814 Digital Output block

Library

Simulink Real-Time 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 each block must use a unique set of channels.

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 arbitrary 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 this 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 replicated over the channel vector. If you specify a value of 1, the corresponding channel is reset to the value specified in the initial value vector. If you specify a value of 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 replicated as the initial value over the channel vector. 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 (-1 means sample time is inherited).

Base address

Enter the base address of the board. This entry must correspond to the DIP switch setting on the board. For example, if the DIP switch setting is 300 (hexadecimal), enter 0x300.

See Also

External Websites

www.rtd.com

Real Time Devices DM6816

Support for the RTD DM6816 I/O board.

Board

Real Time Devices DM6816

General Description

The DM6816 is an PC/104 I/O board that provides:

- 9 independent PWM channels with 8-bit resolution
- 3 16-bit timer/counters
- 1 on-board 8 MHz clock

The Simulink Real-Time block library supports this board with this driver block:

- Real Time Devices DM6816 PWM

Board Characteristics

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

See Also

External Websites

www.rtd.com

Real Time Devices DM6816 PWM

DM6816 PWM block

Library

Simulink Real-Time 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`, send `0.06*256 = 15.36` as the input to the channel. The nearest integer is used, so that the duty cycle is actually `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 arbitrary 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 this 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

Enter the base sample time or a multiple of the base sample time (`-1` means sample time is inherited).

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 d000 (hexadecimal), enter `0xd000`.

See Also

External Websites

www.rtd.com

Sensoray, Speedgoat, Texas Instruments

Sensoray

This topic describes the Sensoray I/O boards supported by the Simulink Real-Time product (`www.sensoray.com`).

Sensoray 526

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 Simulink Real-Time 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

Board Characteristics

Board name	526
Manufacturer	Sensoray
Bus type	PC/104
Access method	I/O mapped
Multiple block instance support	No
Multiple board support	Yes

See Also

External Websites

www.sensoray.com

Sensoray526 AD

Sensoray526 AD block

Library

Simulink Real-Time 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 channel numbers. For example, to use the first, second and fifth channels, enter

```
[1, 2, 5]
```

Number the channels beginning with 1 even if this board manufacturer numbers them beginning with 0.

Sample time

Enter the base sample time or a multiple of the base sample time (`-1` means sample time is inherited).

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.

See Also

External Websites

www.sensoray.com

Sensoray526 Dual AD

Sensoray526 Dual AD block

Library

Simulink Real-Time Library for Sensoray

Note

To perform simultaneous conversion with two boards, control both boards simultaneously with the Sensoray526 Dual AD block. 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 channel numbers. For example, to use the first, second and fifth channels, enter

```
[1, 2, 5]
```

Note, for better 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 if this board manufacturer numbers them beginning with 0.

Sample time

Enter the base sample time or a multiple of the base sample time (`-1` means sample time is inherited).

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.

See Also

External Websites

www.sensoray.com

Sensoray526 DA

Sensoray526 DA block

Library

Simulink Real-Time 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 channel numbers. For example, to use the first, second and fourth channels, enter

```
[1, 2, 4]
```

Number the channels beginning with 1 even if this 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 replicated over the channel vector. If you specify a value of `1`, the corresponding channel is reset to the value specified in the initial value vector. If you specify a value of `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, used by the block at initialization. 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 replicated as initial value over the channel vector. 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 (`-1` means sample time is inherited).

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.

See Also

External Websites

www.sensoray.com

Sensoray526 Dual DA

Sensoray526 Dual DA block

Library

Simulink Real-Time Library for Sensoray

Note

To perform simultaneous conversion with two boards, control both boards simultaneously with the Sensoray526 Dual DA block. The block writes the output registers to both boards before writing the conversion start bits to both boards.

Output from the second board changes state approximately 1–2 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 channel numbers. For example, to use the first, second and fourth channels, enter

```
[1, 2, 4]
```

Number the channels beginning with 1 even if this 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 replicated over the channel vector. If you specify a value of `1`, the corresponding channel is reset to the value specified in the initial value vector. If you specify a value of `0`, the channel remains at the last value attained while the model was running.

Board A Initial value vector/Board B Initial value vector

The initial value vector contains the initial voltage values for the output channels, used by the block at initialization. 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 replicated as initial value over the channel vector. 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 (`-1` means sample time is inherited).

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.

See Also

External Websites

www.sensoray.com

Sensoray526 DI

Sensoray526 DI block

Library

Simulink Real-Time 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 if this board manufacturer numbers them beginning with 0.

Sample time

Enter the base sample time or a multiple of the base sample time (`-1` means sample time is inherited).

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.

See Also

External Websites

www.sensoray.com

Sensoray526 DO

Sensoray526 DO block

Library

Simulink Real-Time Library for Sensoray

Scaling of Input to Output

Hardware Output	Block Input Data Type	Scaling
TTL (0 or +5) volts	input < `0.5`, output = `0` volts input ≥ `0.5`, output = `5` volts	1

Block Parameters

Channels

Enter a vector of numbers between 1 and 8. These are the channels to which the board writes output. Each group, 1 to 4 and 5 to 8, can be either input or output. Do not combine them. For example, if channels 1 to 4 are the input channels, you cannot have output on channel 2. Instead, you can use channels 5 to 8 to be the output channels.

Number the channels beginning with 1 even if this 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 replicated over the channel vector. If you specify a value of `1`, the corresponding channel is reset to the value specified in the initial value vector. If you specify a value of `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, used by the block at initialization. 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 replicated as initial value over the channel vector. 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 (`-1` means sample time is inherited).

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.

See Also

External Websites

www.sensoray.com

Sensoray526 Encoder Input

Sensoray526 Encoder Input block

Library

Simulink Real-Time 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 +- 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 -> 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

Enter the base sample time or a multiple of the base sample time (`-1` means sample time is inherited).

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.

See Also

External Websites

www.sensoray.com

Speedgoat Support

Speedgoat Real-Time Target Machines

Speedgoat GmbH offers target machines optimized for you to use with Simulink Real-Time software.

If you installed a Speedgoat library, to open the Speedgoat documentation, type `speedgoatdoc` in the Command Window.

If not, you can find Speedgoat target machine configuration documentation here:

`www.speedgoat.com/support/real-time_targets`

You can find Speedgoat target machine product information here:

`www.speedgoat.com/products/real-timetargetmachines`

Speedgoat target machines include:

- Performance — Highest performance, cost-effective system for office or lab.
- Mobile — Small, rugged, fanless, and expandable with up to 14 I/O modules. For mobile and in-vehicle use, and for use in confined areas. Extended temperature support.
- Openframe — Small, rugged, and fanless system mounted on aluminum plate, expandable with up to 14 I/O modules. Extended temperature support.
- Education — Industrial-grade system for use in academic institutions. Attractive price.
- Modular — Expandable system, for large-scale projects and hardware-in-the-loop (HIL) simulators.
- Audio — System optimized for audio applications, such as hearing aids and car acoustics.
- Automation — System designed for machine cabinet installation, for around-the-clock operation.

Speedgoat I/O Modules

Speedgoat GmbH offers I/O modules optimized for you to use with Simulink Real-Time software.

If you installed a Speedgoat library, to open the Speedgoat documentation, type `speedgoatdoc` in the Command Window.

If not, you can find Speedgoat I/O module documentation here:

`www.speedgoat.com/support/iomodules`

You can find Speedgoat I/O module product information here:

`www.speedgoat.com/products/iomodules`

Speedgoat I/O modules include support for:

- Analog I/O: A/D, D/A, sample- or frame-based, with or without isolation, 16–24 bit, both voltage and current
- Digital I/O: LVCMOS, TTL, RS422, LVDS
- Multifunctional, reconfigurable FPGA-based:
  - Interrupt
  - PWM generation and capture, pulse patterns
  - Quadrature Decoding and Encoding (Measurement or Simulation)
  - SSI Master and Slave (Measurement and Simulation)
  - SSI2 (Codechamp encoders)
  - EnDat 2.2 (Heidenhain encoders)
  - BiSS (Renishaw encoders)
  - Hiperface
  - SPI Master and Slave, SPI Sniffer
  - I^2C Master and Slave
  - Cam and Crank simulation
- LVDT/RVDT and Synchro/Resolver measurement and simulation
- Serial:

    - RS232, RS422, RS485
    - SDLC, HDLC
- Camera Link and UVC-compliant USB video cameras (webcams)
- Shared memory
- Thermocouple, RTD, and Strain Gage measurement and simulation
- IEPE/ICP measurement
- Programmable Resistors and Potentiometers
- SPDT, SPST, and DPST Reed Relays
- Fault insertion

Speedgoat Communication Protocols

Speedgoat GmbH offers communication protocol modules optimized for you to use with Simulink Real-Time software.

If you installed a Speedgoat library, to open the Speedgoat documentation, type `speedgoatdoc` in the Command Window.

If not, you can find Speedgoat protocol module documentation here:

`www.speedgoat.com/support/iomodules`

You can find Speedgoat protocol product information here:

`www.speedgoat.com/products/iomodules`

Speedgoat communication protocols include:

- CAN, LIN, SAE J1939, K-Line
- XCP over Ethernet, XCP over CAN
- MIL-STD-1553, ARINC-429
- EtherCAT Master, EtherCAT Slave, Real-Time UDP, Real-Time Raw Ethernet
- EtherNet/IP™ Scanner (Master) and EtherNet/IP Adapter (Slave)
- PROFINET Master and PROFINET Slave
- PROFIBUS, Modbus TCP, Modbus RTU, Ethernet POWERLINK
- Timing and synchronization: PTP (Precision Time Protocol, IEEE 1588), GPS, IRIG

Speedgoat Blocks

Speedgoat IO321

FPGA board providing 64 bidirectional LVCMOS or 32 bidirectional LVDS I/O lines (four are input only)

Board

Speedgoat IO321

General Description

The Speedgoat IO321 is a field-programmable gate array (FPGA) board that provides 64 bidirectional LVCMOS or 32 bidirectional LVDS I/O lines (four are input only). The Speedgoat IO321-5 also provides 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, version 10.1 (`www.xilinx.com`). For more information about design tools, see "Supported Third-Party Tools and Hardware" (HDL Coder).

The Speedgoat IO321 is the base board. The Speedgoat IO321-5 is the Speedgoat IO321 plus the AXM-A30 high-speed A/D port subassembly.

The default clock rate for this board is 66 MHz.

The Simulink Real-Time block library supports this board with these driver blocks:

- Speedgoat IO321 PCI Setup
- Speedgoat IO321-5 A/D Calibration
- Speedgoat IO321 PCI Read
- Speedgoat IO321 PCI Write

To use these blocks, open HDL Coder™ HDL Workflow Advisor and use it to generate a Simulink Real-Time interface subsystem. See "FPGA Programming and Configuration".

The subsystem mask controls the block parameters. Do not edit the parameters directly. The FPGA I/O board block descriptions are for informational purposes only.

The Speedgoat IO321 provides up to 64 channels of 2.5 volt LVCMOS and up to 32 channels of 2.5 volt LVDS. You can use both LVCMOS and LVDS channels in a single

system. However, channel selection must uniquely map to connector pins. For example, you cannot concurrently use the same connector pins for both LVCMOS and LVDS. For more information, see the Speedgoat IO321 68 Pin Connector Map (LVCMOS) and the Speedgoat IO321 68 Pin Connector Map (LVDS).

The Speedgoat IO321-5 configuration adds two analog input ports to the input ports provided by the Speedgoat IO321. For more information, see "Speedgoat IO321-5 A/D Input Ports" on page 48-8.

When defining...	Use Table
LVCMOS channels	Speedgoat IO321 68 Pin Connector Map (LVCMOS)
LVDS channels	Speedgoat IO321 68 Pin Connector Map (LVDS)

Speedgoat IO321 68 Pin Connector Map (LVCMOS)

Connector Pin	I/O Channel	Direction	Transceiver
1	0	Input/Output	LVCMOS
35	1	Input/Output	LVCMOS
2	2	Input/Output	LVCMOS
36	3	Input/Output	LVCMOS
3	4	Input/Output	LVCMOS
37	5	Input/Output	LVCMOS
4	6	Input/Output	LVCMOS
38	7	Input/Output	LVCMOS
5	8	Input/Output	LVCMOS
39	9	Input/Output	LVCMOS
6	10	Input/Output	LVCMOS
40	11	Input/Output	LVCMOS
7	12	Input/Output	LVCMOS
41	13	Input/Output	LVCMOS
8	14	Input/Output	LVCMOS
42	15	Input/Output	LVCMOS
9	16	Input/Output	LVCMOS

Connector Pin	I/O Channel	Direction	Transceiver
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
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

Connector Pin	I/O Channel	Direction	Transceiver
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
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 IO321 68 Pin Connector Map (LVDS)

Connector Pin	I/O Channel	Direction	Transceiver
1	0(+)	Input	LVDS
35	0(-)		

Connector Pin	I/O Channel	Direction	Transceiver
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	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(-)		

Connector Pin	I/O Channel	Direction	Transceiver
16	15(+)	Input	LVDS
50	15(-)		
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(-)		
28	27(+)	Input/Output	LVDS
62	27(-)		
29	28(+)	Input/Output	LVDS
63	28(-)		

Connector Pin	I/O Channel	Direction	Transceiver
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	IO321
Clock rate	32–100 MHz, default 66 MHz
Manufacturer	Speedgoat
Bus type	PCI
Multiple block instance support	Yes
Multiple board support	Yes

Speedgoat IO321-5 A/D Input Ports

The Speedgoat IO321-5 high-speed A/D port subassembly supports two analog input ports that connect to a 16-bit A/D converter. It transmits their measured voltage to the FPGA as signed 16-bit fixed-point integers.

- `+32767` represents `+1.6` V
- `0` represents `0` V
- `-32767` represents `-1.6` V

Voltages scale linearly.

When you assign an input port to the A/D in HDL Coder HDL Workflow Advisor, the software adds calibration parameters to the interface block. For more information, see "FPGA Programming and Configuration" (HDL Coder).

Calibration Procedure

Calibration requires a precision voltage source. This source must be able to generate the required calibration voltages into a 50 Ω load to the accuracy and precision that you want.

To calibrate the Speedgoat IO321-5 A/D port assembly for **`Ch 1`** and **`Ch 2`**.

1. In the subsystem interface block mask, initialize parameter **`Gain`**$_{Ch}$ to `1.0`. For information on the **`Gain`** and **`Offset`** data entry format, see Speedgoat IO321-5 A/D Calibration.
2. Connect the voltage source to channel **`Ch`**.
3. Apply `0` V to **`Ch`**. Read the measured value from the A/D several times, recording your measurements. (The measured value is noisy, with a typical uncertainty of `±30` units.)
4. Choose an average value.
5. Open the subsystem interface block mask. Into the **`Offset`** parameter for channel **`Ch`**, enter the negative of the average value, leaving **`Gain`**$_{Ch}$ 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 **`Offset`**$_{Ch}$ to the accuracy and precision that you want.
8. Choose a calibration voltage. Calculate its expected value depending upon the scaling that you want. For example, if you want `1.6` V to produce `32767` units and the calibration voltage is `1.0` V:

   $$1.6 / 32767 = 1.0 / ExpectedValue$$
   $$ExpectedValue = 1.0 / 1.6 * 32767$$
   $$ExpectedValue = 20479$$

9. Apply the calibration voltage (1 V) to the channel. From the A/D, read the measured value. Calculate **`Gain`**$_{Ch}$ using the following equation:

$$Gain = ExpectedValue / MeasuredValue$$

10 Open the subsystem interface block mask. 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 **Ch 1** calibrates as follows:

Expected Value	Measured Value	Offset	Gain
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 Ω load
Operating Temperature	0–70 °C
Signal-to-noise ratio	69 dB (25 °C) typical

Signal-to-noise and distortion	67 dB (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 IO321 A/D Input Port Connector Map

SMA Connector	Connector Description
1	Analog Input Ch1
2	External Clock (not used)
3	External Trigger (not used)
4	Analog Input Ch2

See Also

Topics

"FPGA Programming and Configuration"
"Supported Third-Party Tools and Hardware" (HDL Coder)
"Digital I/O with Speedgoat FPGA Board"
"PLL-Based Interrupt Generation from FPGA Input"

External Websites

www.speedgoat.com/support/iomodules
www.speedgoat.com

Introduced in R2013b

Speedgoat IO321-5 A/D Calibration

Speedgoat IO321-5 AXM-A30 Calibration block factors

Description

The Speedgoat IO321-5 A/D Calibration block generates A/D gain and offset corrections for the Speedgoat IO321-5 A/D Input Port Assembly.

To use these blocks, open HDL Coder HDL Workflow Advisor and use it to generate a Simulink Real-Time interface subsystem. See “FPGA Programming and Configuration”.

The subsystem mask controls the block parameters. Do not edit the parameters directly. The FPGA I/O board block descriptions are for informational purposes only.

Library

Simulink Real-Time Library for FPGA

Block Parameters

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]`**. **`Ch1Gain`** is the gain correction (`double`) for channel 1. **`Ch2Gain`** is the gain correction (`double`) for channel 2.

A/D Offset Corrections

Enter pairs of the form **`[Ch1Offset,Ch2Offset]`**. **`Ch1Offset`** is the offset correction (`int16`) for channel 1. **`Ch2Offset`** is the offset correction (`int16`) for channel 2.

For more information on how to calculate these factors, see “Speedgoat IO321-5 A/D Input Ports” on page 48-8.

See Also

Topics

"FPGA Programming and Configuration"
"Supported Third-Party Tools and Hardware" (HDL Coder)
"Digital I/O with Speedgoat FPGA Board"
"PLL-Based Interrupt Generation from FPGA Input"

External Websites

www.speedgoat.com/support/iomodules
www.speedgoat.com

Introduced in R2013b

Speedgoat IO321 PCI Read

Speedgoat IO321 PCI Read block

Description

The Speedgoat IO321 PCI Read block reads a 32–bit data value from a PCI port offset address. Output is a `uint32` data type.

To use these blocks, open HDL Coder HDL Workflow Advisor and use it to generate a Simulink Real-Time interface subsystem. See “FPGA Programming and Configuration”.

The subsystem mask controls the block parameters. Do not edit the parameters directly. The FPGA I/O board block descriptions are for informational purposes only.

Library

Simulink Real-Time Library for FPGA

Block Parameters

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.

HDL Workflow Advisor fills in this value based upon the board configuration.

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 being used to latch data from the FPGA into the ports being read. If **Strobe offset** is `0`, the strobe is not used.

HDL Workflow Advisor fills in this value based upon the board configuration.

Sample time

Enter the base sample time or a multiple of the base sample time (`-1` means sample time is inherited).

See Also

Topics

"FPGA Programming and Configuration"
"Supported Third-Party Tools and Hardware" (HDL Coder)
"Digital I/O with Speedgoat FPGA Board"
"PLL-Based Interrupt Generation from FPGA Input"

External Websites

www.speedgoat.com/support/iomodules
www.speedgoat.com

Introduced in R2013b

Speedgoat IO321 PCI Setup

Speedgoat IO321 PCI Setup block

Description

The Speedgoat IO321 PCI Setup block performs the setup and configuration to use the Speedgoat IO321 board.

To use these blocks, open HDL Coder HDL Workflow Advisor and use it to generate a Simulink Real-Time interface subsystem. See “FPGA Programming and Configuration”.

The subsystem mask controls the block parameters. Do not edit the parameters directly. The FPGA I/O board block descriptions are for informational purposes only.

Library

Simulink Real-Time Library for FPGA

Block Parameters

FPGA model/subsystem name

Enter a character vector to identify the model and subsystem containing the Setup block.

Run FPGA only when Simulink Real-Time runs

Select this check box for the FPGA to execute only when the real-time application is running.

Clear this check box for the FPGA to start immediately after the real-time application is loaded.

Reset FPGA states on start

Select this check box for the FPGA to reset the initial state of the FPGA variables and registers to the default state on startup. Takes effect only when you select the check box **Run FPGA only when Simulink Real-Time runs**.

Clear this check box for the FPGA to retain the end state of the previous execution. To return to the initial state, 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 Speedgoat target machine, enter `-1` to locate the board.

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

To determine the bus number and the PCI slot number, type:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

Topics

"FPGA Programming and Configuration"
"Supported Third-Party Tools and Hardware" (HDL Coder)
"Digital I/O with Speedgoat FPGA Board"
"PLL-Based Interrupt Generation from FPGA Input"

External Websites

www.speedgoat.com/support/iomodules
www.speedgoat.com

Introduced in R2013b

Speedgoat IO321 PCI Write

Speedgoat IO321 PCI Write block

Description

The Speedgoat IO321 PCI Write block writes a 32–bit value to a PCI port offset address. Input must be a `uint32` data type.

To use these blocks, open HDL Coder HDL Workflow Advisor and use it to generate a Simulink Real-Time interface subsystem. See “FPGA Programming and Configuration”.

The subsystem mask controls the block parameters. Do not edit the parameters directly. The FPGA I/O board block descriptions are for informational purposes only.

Library

Simulink Real-Time Library for FPGA

Block Parameters

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.

HDL Workflow Advisor fills in this value based upon the board configuration.

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 being used to latch data from the ports into the FPGA. If **Strobe offset** is `0`, the strobe is not used.

HDL Workflow Advisor fills in this value based upon the board configuration.

Sample time

Enter the base sample time or a multiple of the base sample time (`-1` means sample time is inherited).

See Also

Topics

"FPGA Programming and Configuration"
"Supported Third-Party Tools and Hardware" (HDL Coder)
"Digital I/O with Speedgoat FPGA Board"
"PLL-Based Interrupt Generation from FPGA Input"

External Websites

www.speedgoat.com/support/iomodules
www.speedgoat.com

Introduced in R2013b

Speedgoat IO331

FPGA board providing 64 bidirectional LVCMOS or 32 bidirectional LVDS I/O lines

Board

Speedgoat IO331

General Description

The Speedgoat IO331 is a field-programmable gate array (FPGA) board that provides 64 bidirectional LVCMOS or 32 bidirectional LVDS I/O lines. This board is based on a Xilinx Spartan® 6 chip with 147333 logic cells. Design tools supporting this chip include the Xilinx ISE Design Suite, version 14.7 (`www.xilinx.com`). For more information about design tools, see "Supported Third-Party Tools and Hardware" (HDL Coder).

The Speedgoat IO331 is the base board. The Speedgoat IO331-6 is the AXM-A75 A/D converter, an add-on to the Speedgoat IO331.

The default clock rate for this board is 75 MHz. When you use the AXM-A75 A/D or D/A interfaces, the clock rate cannot exceed 75 MHz.

The Simulink Real-Time block library supports this board with these driver blocks:

- Speedgoat IO3xx PCI Setup
- Speedgoat IO3xx PCI Read
- Speedgoat IO3xx PCI Write
- Speedgoat IO3xx PCI Read/Write
- Speedgoat IO331-6 A/D Range

To use these blocks, open HDL Coder HDL Workflow Advisor and use it to generate a Simulink Real-Time interface subsystem. See "FPGA Programming and Configuration".

The subsystem mask controls the block parameters. Do not edit the parameters directly. The FPGA I/O board block descriptions are for informational purposes only.

The Speedgoat IO331 provides up to 64 channels of 2.5 volt LVCMOS and up to 32 channels of 2.5 volt LVDS. You can use both LVCMOS and LVDS channels in a single system. However, channel selection must uniquely map to connector pins. For example,

you cannot concurrently use the same connector pins for both LVCMOS and LVDS. For more information on channel/pin assignment, see the following tables.

The Speedgoat IO331-6 configuration adds 16 differential analog input channels, 8 analog output channels, and 16 digital I/O channels to the channels provided by the Speedgoat IO331. For more information, see “Speedgoat IO331-6 A/D and D/A Converter” on page 48-26.

When defining...	Use Table
LVCMOS channels	Speedgoat IO331 68 Pin Connector Map (LVCMOS)
LVDS channels	Speedgoat IO331 68 Pin Connector Map (LVDS)

Speedgoat IO331 68 Pin Connector Map (LVCMOS)

Connector Pin	I/O Channel	Direction	Transceiver
1	0	Input/Output	LVCMOS
2	1	Input/Output	LVCMOS
35	2	Input/Output	LVCMOS
36	3	Input/Output	LVCMOS
3	4	Input/Output	LVCMOS
4	5	Input/Output	LVCMOS
37	6	Input/Output	LVCMOS
38	7	Input/Output	LVCMOS
5	8	Input/Output	LVCMOS
6	9	Input/Output	LVCMOS
39	10	Input/Output	LVCMOS
40	11	Input/Output	LVCMOS
7	12	Input/Output	LVCMOS
8	13	Input/Output	LVCMOS
41	14	Input/Output	LVCMOS
42	15	Input/Output	LVCMOS
9	16	Input/Output	LVCMOS
10	17	Input/Output	LVCMOS

Connector Pin	I/O Channel	Direction	Transceiver
43	18	Input/Output	LVCMOS
44	19	Input/Output	LVCMOS
11	20	Input/Output	LVCMOS
12	21	Input/Output	LVCMOS
45	22	Input/Output	LVCMOS
46	23	Input/Output	LVCMOS
13	24	Input/Output	LVCMOS
14	25	Input/Output	LVCMOS
47	26	Input/Output	LVCMOS
48	27	Input/Output	LVCMOS
15	28	Input/Output	LVCMOS
16	29	Input/Output	LVCMOS
49	30	Input/Output	LVCMOS
50	31	Input/Output	LVCMOS
17	32	Input/Output	LVCMOS
18	33	Input/Output	LVCMOS
51	34	Input/Output	LVCMOS
52	35	Input/Output	LVCMOS
19	36	Input/Output	LVCMOS
20	37	Input/Output	LVCMOS
53	38	Input/Output	LVCMOS
54	39	Input/Output	LVCMOS
21	40	Input/Output	LVCMOS
22	41	Input/Output	LVCMOS
55	42	Input/Output	LVCMOS
56	43	Input/Output	LVCMOS
23	44	Input/Output	LVCMOS
24	45	Input/Output	LVCMOS

Connector Pin	I/O Channel	Direction	Transceiver
57	46	Input/Output	LVCMOS
58	47	Input/Output	LVCMOS
25	48	Input/Output	LVCMOS
26	49	Input/Output	LVCMOS
59	50	Input/Output	LVCMOS
60	51	Input/Output	LVCMOS
27	52	Input/Output	LVCMOS
28	53	Input/Output	LVCMOS
61	54	Input/Output	LVCMOS
62	55	Input/Output	LVCMOS
29	56	Input/Output	LVCMOS
30	57	Input/Output	LVCMOS
63	58	Input/Output	LVCMOS
64	59	Input/Output	LVCMOS
31	60	Input/Output	LVCMOS
32	61	Input/Output	LVCMOS
65	62	Input/Output	LVCMOS
66	63	Input/Output	LVCMOS
33		Ground	
34		Ground	
67		Ground	
68		Ground	

Speedgoat IO331 68 Pin Connector Map (LVDS)

Connector Pin	I/O Channel	Direction	Transceiver
1	0(+)	Input/Output	LVDS
2	0(-)		
35	1(+)	Input/Output	LVDS

Connector Pin	I/O Channel	Direction	Transceiver
36	1(-)		
3	2(+)	Input/Output	LVDS
4	2(-)		
37	3(+)	Input/Output	LVDS
38	3(-)		
5	4(+)	Input/Output	LVDS
6	4(-)		
39	5(+)	Input/Output	LVDS
40	5(-)		
7	6(+)	Input/Output	LVDS
8	6(-)		
41	7(+)	Input/Output	LVDS
42	7(-)		
9	8(+)	Input/Output	LVDS
10	8(-)		
43	9(+)	Input/Output	LVDS
44	9(-)		
11	10(+)	Input/Output	LVDS
12	10(-)		
45	11(+)	Input/Output	LVDS
46	11(-)		
13	12(+)	Input/Output	LVDS
14	12(-)		
47	13(+)	Input/Output	LVDS
48	13(-)		
15	14(+)	Input/Output	LVDS
16	14(-)		
49	15(+)	Input/Output	LVDS

Connector Pin	I/O Channel	Direction	Transceiver
50	15(-)		
17	16(+)	Input/Output	LVDS
18	16(-)		
51	17(+)	Input/Output	LVDS
52	17(-)		
19	18(+)	Input/Output	LVDS
20	18(-)		
53	19(+)	Input/Output	LVDS
54	19(-)		
21	20(+)	Input/Output	LVDS
22	20(-)		
55	21(+)	Input/Output	LVDS
56	21(-)		
23	22(+)	Input/Output	LVDS
24	22(-)		
57	23(+)	Input/Output	LVDS
58	23(-)		
25	24(+)	Input/Output	LVDS
26	24(-)		
59	25(+)	Input/Output	LVDS
60	25(-)		
27	26(+)	Input/Output	LVDS
28	26(-)		
61	27(+)	Input/Output	LVDS
62	27(-)		
29	28(+)	Input/Output	LVDS
30	28(-)		
63	29(+)	Input/Output	LVDS

Connector Pin	I/O Channel	Direction	Transceiver
64	29(-)		
31	30(+)	Input/Output	LVDS
32	30(-)		
65	31(+)	Input/Output	LVDS
66	31(-)		
33	Ground		
34	Ground		
67	Ground		
68	Ground		

Board Characteristics

Board name	IO331
Clock rate	10–125 MHz, default 75 MHz
Manufacturer	Speedgoat
Bus type	PCI
Multiple block instance support	Yes
Multiple board support	Yes

Speedgoat IO331-6 A/D and D/A Converter

The Speedgoat IO331-6 A/D converter supports 16 differential analog input channels. It transmits their measured voltage to the FPGA as signed 16-bit fixed-point integers. The IO331-6 D/A converter supports 8 analog output channels and receives their voltage setting from the FPGA as signed 16-bit fixed-point integers.

For both input and output channels:

- `+32767` represents the positive end of the input range
- `0` represents `0` V

- `-32767` represents the negative end of the input range

When the board is running at its maximum clock rate of 75 MHz, the A/D sampling rate is approximately 500 kHz. As the board clock rate drops to 10 MHz, the sampling rate decreases linearly to 200 kHz.

The D/A sampling rate is approximately 100 kHz.

When you use the A/D or D/A interfaces, the IO331-6 clock rate cannot exceed 75 MHz.

Port Characteristics

Name	AXM-A75
Manufacturer	Acromag
Analog input configuration	16 differential channels using 16 Analog Devices AD7686 A/D converters
Input range	±10.24, ±5.12, ±2.56, ±1.28 V
A/D resolution	16 bits
Maximum A/D trigger rate	500 kHz
Analog output configuration	8 channels using 8 Analog Devices AD5764RB D/A converters
Output range	±10, ±10.2564, ±10.5263 V
D/A resolution	16 bits
D/A trigger rate (fixed)	100 kHz
Digital I/O	16 TTL channels
Operating temperature	0–70 °C
Maximum clock rate	75 MHz
Connectors	68-pin VHDCI receptacle

In the J1 connector map, the pin names are adapted from the manufacturer data sheet. The pin names, which are zero-based, map to the names used in HDL Coder HDL Workflow Advisor as follows:

- VIN —`A/D Input Channel`
- VOUT — `D/A Output Channel`
- DIO — `TTL I/O (AXM-A75) Channel [0:15]`

Each digital I/O pin can be configured as either input or output, but not as both at the same time.

Speedgoat IO331-6 A/D Converter 68-Pin J1 Connector Map

Pin Description	Number	Pin Description	Number
GND	1	DIO0	35
DIO1	2	DIO2	36
DIO3	3	DIO4	37
DIO5	4	DIO6	38
DIO7	5	GND	39
DIO8	6	DIO9	40
DIO10	7	DIO11	41
DIO12	8	DIO13	42
DIO14	9	DIO15	43
GND	10	VOUT0	44
GND	11	VOUT1	45
GND	12	VOUT2	46
GND	13	VOUT3	47
GND	14	VOUT4	48
GND	15	VOUT5	49
GND	16	VOUT6	50
GND	17	VOUT7	51
GND	18	GND	52
VIN15-	19	VIN15+	53
VIN14-	20	VIN14+	54
VIN13-	21	VIN13+	55
VIN12-	22	VIN12+	56
VIN11-	23	VIN11+	57
VIN10-	24	VIN10+	58

Pin Description	Number	Pin Description	Number
VIN9-	25	VIN9+	59
VIN8-	26	VIN8+	60
VIN7-	27	VIN7+	61
VIN6-	28	VIN6+	62
VIN5-	29	VIN5+	63
VIN4-	30	VIN4+	64
VIN3-	31	VIN3+	65
VIN2-	32	VIN2+	66
VIN1-	33	VIN1+	67
VIN0-	34	VIN0+	68

See Also

Topics

"FPGA Programming and Configuration"
"Supported Third-Party Tools and Hardware" (HDL Coder)
"Digital I/O with Speedgoat FPGA Board"
"PLL-Based Interrupt Generation from FPGA Input"

External Websites

www.speedgoat.com/support/iomodules
www.speedgoat.com

Introduced in R2013a

Speedgoat IO3xx PCI Setup

Speedgoat IO3xx PCI Setup block

Description

The Speedgoat IO3xx PCI Setup block performs the setup and configuration to use the Speedgoat IO3xx boards.

To use these blocks, open HDL Coder HDL Workflow Advisor and use it to generate a Simulink Real-Time interface subsystem. See “FPGA Programming and Configuration”.

The subsystem mask controls the block parameters. Do not edit the parameters directly. The FPGA I/O board block descriptions are for informational purposes only.

Library

Simulink Real-Time Library for FPGA

Block Parameters

FPGA model/subsystem name

Enter a character vector to identify the model and subsystem containing the Setup block.

FPGA board type

Select the type of FPGA being set up. One of 301, 302, 303, 311, 312, 313, 314, 321, 331.

You can use this block for all of the Speedgoat IO3xx FPGA I/O boards.

Run FPGA only when Simulink Real-Time runs

Select this check box for the FPGA to execute only when the real-time application is running.

Clear this check box for the FPGA to start immediately after the real-time application is loaded.

Reset FPGA states on start

Select this check box for the FPGA to reset the initial state of the FPGA variables and registers to the default state on startup. Takes effect only when you select the check box **Run FPGA only when Simulink Real-Time runs**.

Clear this check box for the FPGA to retain the end state of the previous execution. To return to the initial state, 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 Speedgoat target machine, enter `-1` to locate the board.

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

To determine the bus number and the PCI slot number, type:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

Topics

"FPGA Programming and Configuration"
"Supported Third-Party Tools and Hardware" (HDL Coder)
"Digital I/O with Speedgoat FPGA Board"
"PLL-Based Interrupt Generation from FPGA Input"

External Websites

www.speedgoat.com/support/iomodules
www.speedgoat.com

Introduced in R2013a

Speedgoat IO3xx PCI Read

Speedgoat IO3xx PCI Read block

Description

The Speedgoat IO3xx PCI Read block reads a 32-bit data value from a PCI port offset address. Output is a `uint32` data type.

To use these blocks, open HDL Coder HDL Workflow Advisor and use it to generate a Simulink Real-Time interface subsystem. See “FPGA Programming and Configuration”.

The subsystem mask controls the block parameters. Do not edit the parameters directly. The FPGA I/O board block descriptions are for informational purposes only.

Library

Simulink Real-Time Library for FPGA

Block Parameters

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.

HDL Workflow Advisor fills in this value based upon the board configuration.

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 being used to latch data from the FPGA into the ports being read. If **Strobe offset** is `0`, the strobe is not used.

HDL Workflow Advisor fills in this value based upon the board configuration.

Sample time

Enter the base sample time or a multiple of the base sample time (`-1` means sample time is inherited).

See Also

Topics

"FPGA Programming and Configuration"
"Supported Third-Party Tools and Hardware" (HDL Coder)
"Digital I/O with Speedgoat FPGA Board"
"PLL-Based Interrupt Generation from FPGA Input"

External Websites

www.speedgoat.com/support/iomodules
www.speedgoat.com

Introduced in R2013a

Speedgoat IO3xx PCI Write

Speedgoat IO3xx PCI Write block

Description

The Speedgoat IO3xx PCI Write block writes a 32-bit data value to a PCI port offset address. Input must be a `uint32` data type.

To use these blocks, open HDL Coder HDL Workflow Advisor and use it to generate a Simulink Real-Time interface subsystem. See “FPGA Programming and Configuration”.

The subsystem mask controls the block parameters. Do not edit the parameters directly. The FPGA I/O board block descriptions are for informational purposes only.

Library

Simulink Real-Time Library for FPGA

Block Parameters

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.

HDL Workflow Advisor fills in this value based upon the board configuration.

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 being used to latch data from the ports into the FPGA. If **Strobe offset** is `0`, the strobe is not used.

HDL Workflow Advisor fills in this value based upon the board configuration.

Sample time

Enter the base sample time or a multiple of the base sample time (`-1` means sample time is inherited).

See Also

Topics

"FPGA Programming and Configuration"
"Supported Third-Party Tools and Hardware" (HDL Coder)
"Digital I/O with Speedgoat FPGA Board"
"PLL-Based Interrupt Generation from FPGA Input"

External Websites

www.speedgoat.com/support/iomodules
www.speedgoat.com

Introduced in R2013a

Speedgoat IO3xx PCI Read/Write

Speedgoat IO3xx PCI Read/Write block

Description

The Speedgoat IO3xx PCI Read/Write block reads one or more 32-bit data values from a PCI port offset address. The block output is of data type `uint32`. The Read/Write block writes one or more 32-bit data values to a PCI port offset address. The block input must be of data type `uint32`.

The block supports three synchronization modes:

- `Free running`
- `Coprocessing — blocking`
- `Coprocessing — nonblocking with delay`

For more information about these modes, see "FPGA Synchronization Modes".

To use these blocks, open HDL Coder HDL Workflow Advisor and use it to generate a Simulink Real-Time interface subsystem. See "FPGA Programming and Configuration".

The subsystem mask controls the block parameters. Do not edit the parameters directly. The FPGA I/O board block descriptions are for informational purposes only.

Library

Simulink Real-Time Library for FPGA

Block Parameters

Device index

Enter a number between `0` and `7`. The maximum number of allowed boards is 8.

Operating mode

Select one of the following:

- `Free running` (default) — Nonsynchronized. The CPU and the FPGA each run continuously and in parallel. When using the FPGA as an I/O module, select this mode.
- `Coprocessing – blocking` — Synchronized. When the FPGA execution time is short compared to the Speedgoat target machine sample time, select this coprocessor mode.
- `Coprocessing – nonblocking with delay` — Synchronized. When the FPGA execution time is long compared to the Speedgoat target machine sample time, select this coprocessor mode.

Non-streaming write offset

Enter the PCI register offset address for the ports being written with non-streaming data.

HDL Workflow Advisor fills in this value based upon the board configuration.

Non-streaming write types

Enter a cell array containing the data type of each port being written with non-streaming data `uint32` (fixed)). For example, enter `{'uint32'}`.

Non-streaming write lengths

Enter a cell array containing the number of vector elements in the port being written with non-streaming data. For example, enter `[1]`.

If the sum of the values in **Non-streaming write lengths** is greater than 1, you must set **Write strobe offset** to a nonzero value. For example, both `[5]` and `[1 1]` require a strobe.

Streaming write offset

For internal use only.

Streaming write types

For internal use only.

Streaming write length

For internal use only.

Write strobe offset

Enter the PCI register offset address for the signal being used to latch data from the ports into the FPGA. If **Write strobe offset** is `0`, the strobe is not used.

HDL Workflow Advisor fills in this value based upon the board configuration.

Non-streaming read offset

Enter the PCI register offset address for the ports being read as non-streaming data.

HDL Workflow Advisor fills in this value based upon the board configuration.

Non-streaming read types

Enter a cell array containing the data type of each port being read as non-streaming data `uint32` (fixed)). For example, enter `{'uint32'}`.

Non-streaming read lengths

Enter a cell array containing the number of vector elements in the port being read as non-streaming data. For example, enter `[1]`.

If the sum of the values in **Non-streaming read lengths** is greater than 1, you must set **Read strobe offset** to a nonzero value. For example, both `[5]` and `[1 1]` require a strobe.

Streaming read offset

For internal use only.

Streaming read types

For internal use only.

Streaming read length

For internal use only.

Read strobe offset

Enter the PCI register offset address for the signal being used to latch data from the FPGA into the ports being read. If **Read strobe offset** is `0`, the strobe is not used.

HDL Workflow Advisor fills in this value based upon the board configuration.

Sample time

Enter the base sample time or a multiple of the base sample time (`-1` means sample time is inherited).

See Also

Topics

"FPGA Programming and Configuration"
"Supported Third-Party Tools and Hardware" (HDL Coder)

"Digital I/O with Speedgoat FPGA Board"
"PLL-Based Interrupt Generation from FPGA Input"

External Websites

www.speedgoat.com/support/iomodules
www.speedgoat.com

Introduced in R2013a

Speedgoat IO331-6 A/D Range

Speedgoat IO331-6 AXM-A75 A/D range setup block

Description

The Speedgoat IO331-6 A/D Range block specifies the voltage range for the Speedgoat IO331-6 A/D converter.

To use these blocks, open HDL Coder HDL Workflow Advisor and use it to generate a Simulink Real-Time interface subsystem. See “FPGA Programming and Configuration”.

The subsystem mask controls the block parameters. Do not edit the parameters directly. The FPGA I/O board block descriptions are for informational purposes only.

Library

Simulink Real-Time Library for FPGA

Block Parameters

Device index

Enter a number between `0` and `7`. The maximum number of allowed boards is 8.

Range

Select the voltage range for the analog input. Legal values are:

- `+–10.24 volts`
- `+–5.12 volts`
- `+–2.56 volts`
- `+–1.28 volts`

See Also

Topics

"FPGA Programming and Configuration"
"Supported Third-Party Tools and Hardware" (HDL Coder)
"Digital I/O with Speedgoat FPGA Board"
"PLL-Based Interrupt Generation from FPGA Input"

External Websites

www.speedgoat.com/support/iomodules
www.speedgoat.com

Introduced in R2013b

Speedgoat IO333

FPGA board in the IO333-325K-06 configuration

Board

Speedgoat IO333

General Description

The Speedgoat IO333 is a field-programmable gate array (FPGA) board based on a Xilinx Kintex® 7 chip with 325 K logic cells. Design tools supporting this chip include the Xilinx Vivado® Design Suite 2016.2 (`www.xilinx.com`). For more information about design tools, see "Supported Third-Party Tools and Hardware" (HDL Coder).

The Simulink Real-Time block library supports the Speedgoat IO333 with these driver blocks:

- Speedgoat IO333 PCI Setup
- Speedgoat IO333 PCI Read
- Speedgoat IO333 PCI Write
- Speedgoat IO333 PCI Read/Write

To use these blocks, open HDL Coder HDL Workflow Advisor and use it to generate a Simulink Real-Time interface subsystem. See "FPGA Programming and Configuration".

The subsystem mask controls the block parameters. Do not edit the parameters directly. The FPGA I/O board block descriptions are for informational purposes only.

Front plugin boards provide I/O connections for different configurations of the Speedgoat IO333. HDL Coder HDL Workflow Advisor supports the Speedgoat IO333-325K-06 configuration. To use the hardware interface on the `-06` plugin module, first install *Speedgoat HDL Coder Integration Package for the IO333-325K*. For more information, see the documentation for *Speedgoat HDL Coder Integration Package for the IO333-325K* (www.speedgoat.com/support/iomodules).

Board Characteristics

Board name	IO333
Clock rate	10–200 MHz, default 100 MHz
Manufacturer	Speedgoat
Bus type	PCI Express
Multiple block instance support	Yes
Multiple board support	Yes

See Also

Topics

"FPGA Programming and Configuration"
"Supported Third-Party Tools and Hardware" (HDL Coder)
"Digital I/O with Speedgoat FPGA Board"
"PLL-Based Interrupt Generation from FPGA Input"

External Websites

www.speedgoat.com/support/iomodules
www.speedgoat.com

Introduced in R2016b

Speedgoat IO333 PCI Setup

Speedgoat IO333 PCI Setup block

Description

The Speedgoat IO333 PCI Setup block performs the setup and configuration to use the Speedgoat IO333 boards.

To use these blocks, open HDL Coder HDL Workflow Advisor and use it to generate a Simulink Real-Time interface subsystem. See “FPGA Programming and Configuration”.

The subsystem mask controls the block parameters. Do not edit the parameters directly. The FPGA I/O board block descriptions are for informational purposes only.

Library

Simulink Real-Time Library for FPGA

Block Parameters

FPGA model/subsystem name

Enter a character vector to identify the model and subsystem containing the Setup block.

FPGA board type

Select the type of FPGA being set up. Set by default to `IO333-325K`.

Run FPGA only when Simulink Real-Time runs

Select this check box for the FPGA to execute only when the real-time application is running.

Clear this check box for the FPGA to start immediately after the real-time application is loaded.

Reset FPGA states on start

Select this check box for the FPGA to reset the initial state of the FPGA variables and registers to the default state on startup. Takes effect only when you select the check box **Run FPGA only when Simulink Real-Time runs**.

Clear this check box for the FPGA to retain the end state of the previous execution. To return to the initial state, 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 Speedgoat target machine, enter `-1` to locate the board.

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

To determine the bus number and the PCI slot number, type:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

Topics

"FPGA Programming and Configuration"
"Supported Third-Party Tools and Hardware" (HDL Coder)
"Digital I/O with Speedgoat FPGA Board"
"PLL-Based Interrupt Generation from FPGA Input"

External Websites

www.speedgoat.com/support/iomodules
www.speedgoat.com

Introduced in R2016b

Speedgoat IO333 PCI Read

Speedgoat IO333 PCI Read block

Description

The Speedgoat IO333 PCI Read block reads a 32-bit data value from a PCI port offset address. Output is a `uint32` data type.

To use these blocks, open HDL Coder HDL Workflow Advisor and use it to generate a Simulink Real-Time interface subsystem. See “FPGA Programming and Configuration”.

The subsystem mask controls the block parameters. Do not edit the parameters directly. The FPGA I/O board block descriptions are for informational purposes only.

Library

Simulink Real-Time Library for FPGA

Block Parameters

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.

HDL Workflow Advisor fills in this value based upon the board configuration.

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 being used to latch data from the FPGA into the ports being read. If **Strobe offset** is `0`, the strobe is not used.

HDL Workflow Advisor fills in this value based upon the board configuration.

Sample time

Enter the base sample time or a multiple of the base sample time (`-1` means sample time is inherited).

See Also

Topics

"FPGA Programming and Configuration"
"Supported Third-Party Tools and Hardware" (HDL Coder)
"Digital I/O with Speedgoat FPGA Board"
"PLL-Based Interrupt Generation from FPGA Input"

External Websites

www.speedgoat.com/support/iomodules
www.speedgoat.com

Introduced in R2016b

Speedgoat IO333 PCI Write

Speedgoat IO333 PCI Write block

Description

The Speedgoat IO333 PCI Write block writes a 32-bit data value to a PCI port offset address. Input must be a `uint32` data type.

To use these blocks, open HDL Coder HDL Workflow Advisor and use it to generate a Simulink Real-Time interface subsystem. See “FPGA Programming and Configuration”.

The subsystem mask controls the block parameters. Do not edit the parameters directly. The FPGA I/O board block descriptions are for informational purposes only.

Library

Simulink Real-Time Library for FPGA

Block Parameters

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.

HDL Workflow Advisor fills in this value based upon the board configuration.

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 being used to latch data from the ports into the FPGA. If **Strobe offset** is `0`, the strobe is not used.

HDL Workflow Advisor fills in this value based upon the board configuration.

Sample time

Enter the base sample time or a multiple of the base sample time (`-1` means sample time is inherited).

See Also

Topics

"FPGA Programming and Configuration"
"Supported Third-Party Tools and Hardware" (HDL Coder)
"Digital I/O with Speedgoat FPGA Board"
"PLL-Based Interrupt Generation from FPGA Input"

External Websites

www.speedgoat.com/support/iomodules
www.speedgoat.com

Introduced in R2016b

Speedgoat IO333 PCI Read/Write

Speedgoat IO333 PCI Read/Write block

Description

The Speedgoat IO333 PCI Read/Write block reads one or more 32-bit data values from a PCI port offset address. The block output is of data type `uint32`. The Read/Write block writes one or more 32-bit data values to a PCI port offset address. The block input must be of data type `uint32`.

The block supports three synchronization modes:

- `Free running`
- `Coprocessing — blocking`
- `Coprocessing — nonblocking with delay`

For more information about these modes, see “FPGA Synchronization Modes”.

To use these blocks, open HDL Coder HDL Workflow Advisor and use it to generate a Simulink Real-Time interface subsystem. See “FPGA Programming and Configuration”.

The subsystem mask controls the block parameters. Do not edit the parameters directly. The FPGA I/O board block descriptions are for informational purposes only.

Library

Simulink Real-Time Library for FPGA

Block Parameters

Device index

Enter a number between `0` and `7`. The maximum number of allowed boards is 8.

Operating mode

Select one of the following:

- `Free running` (default) — Nonsynchronized. The CPU and the FPGA each run continuously and in parallel. When using the FPGA as an I/O module, select this mode.
- `Coprocessing – blocking` — Synchronized. When the FPGA execution time is short compared to the Speedgoat target machine sample time, select this coprocessor mode.
- `Coprocessing – nonblocking with delay` — Synchronized. When the FPGA execution time is long compared to the Speedgoat target machine sample time, select this coprocessor mode.

Non-streaming write offset

Enter the PCI register offset address for the ports being written with non-streaming data.

HDL Workflow Advisor fills in this value based upon the board configuration.

Non-streaming write types

Enter a cell array containing the data type of each port being written with non-streaming data `uint32` (fixed)). For example, enter `{'uint32'}`.

Non-streaming write lengths

Enter a cell array containing the number of vector elements in the port being written with non-streaming data. For example, enter `[1]`.

If the sum of the values in **Non-streaming write lengths** is greater than 1, you must set **Write strobe offset** to a nonzero value. For example, both `[5]` and `[1 1]` require a strobe.

Streaming write offset

For internal use only.

Streaming write types

For internal use only.

Streaming write length

For internal use only.

Write strobe offset

Enter the PCI register offset address for the signal being used to latch data from the ports into the FPGA. If **Write strobe offset** is `0`, the strobe is not used.

HDL Workflow Advisor fills in this value based upon the board configuration.

Non-streaming read offset

Enter the PCI register offset address for the ports being read as non-streaming data.

HDL Workflow Advisor fills in this value based upon the board configuration.

Non-streaming read types

Enter a cell array containing the data type of each port being read as non-streaming data `uint32` (fixed)). For example, enter `{'uint32'}`.

Non-streaming read lengths

Enter a cell array containing the number of vector elements in the port being read as non-streaming data. For example, enter `[1]`.

If the sum of the values in **Non-streaming read lengths** is greater than 1, you must set **Read strobe offset** to a nonzero value. For example, both `[5]` and `[1 1]` require a strobe.

Streaming read offset

For internal use only.

Streaming read types

For internal use only.

Streaming read length

For internal use only.

Read strobe offset

Enter the PCI register offset address for the signal being used to latch data from the FPGA into the ports being read. If **Read strobe offset** is `0`, the strobe is not used.

HDL Workflow Advisor fills in this value based upon the board configuration.

Sample time

Enter the base sample time or a multiple of the base sample time (`-1` means sample time is inherited).

See Also

Topics

"FPGA Programming and Configuration"
"Supported Third-Party Tools and Hardware" (HDL Coder)

"Digital I/O with Speedgoat FPGA Board"
"PLL-Based Interrupt Generation from FPGA Input"

External Websites

www.speedgoat.com/support/iomodules
www.speedgoat.com

Introduced in R2016b

UEI, Asynchronous Events

United Electronic Industries (UEI)

This topic describes how to use the UEI groups of boards supported by the Simulink Real-Time product (`www.ueidaq.com`).

A/D Frame Acquisition

In A/D frame acquisition, also called hardware-timed acquisition, the board acquires a set of data points equally spaced in time. This set of data points is called a frame. To process a frame, the hardware raises an interrupt, which triggers the model to process the frame. Because the board acquires the entire frame before raising the interrupt, the frame does not have data gaps at the boundaries.

You can model high-throughput environmental sensors and audio processing using A/D frame blocks.

More About

- "Specifying UEI Boards" on page 49-3

Specifying UEI Boards

The United Electronic Industries (UEI) PowerDAQ board series contains many 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** —- Letters that represent ranges denote gain. 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 providing:

- Sixteen 12-bit analog input channels
- A speed of 1 million samples per second
- Available gains of 1, 10, 100, or 1000

The board displayed on the block is the current board type. To specify a different board, double-click the block. In the **Board Type** list, choose a different board.

Analog Input Frame Driver Blocks

In this section...

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 Simulink Real-Time software. The Simulink Real-Time 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. In the model Configuration Parameters dialog box, configure the **I/O board generating the interrupt** and **Real-Time interrupt source** settings with those boards and those IRQ values.

Note: To use the UEI PowerDAQ board series with the Simulink Real-Time Analog Input frame driver blocks, upgrade the UEI PowerDAQ boards. Contact your UEI representative and request the Simulink Real-Time 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:

- Set the slowest board as the master. To use a faster board as the master, derive a value for the master board **Acquisition frequency** parameter that enables

the slowest board in the configuration to work. Set the **Acquisition frequency** parameter of the master board driver block to this value. 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 box, enter a value in the **Priority** property. A lower number indicates a higher priority. For example, if you have multiple slave blocks, enter values of `2` for the slave blocks and `3` for the master block.
- The number of channels in use on the master does not need to be the same as the number in use on the slave boards.
- Connect the master and its slaves with the J6 connectors on the boards and cables. This connection links the acquisition clock and scan clock from the master board to the 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 target computer configuration.

This topic includes

- "Single UEI Board" on page 49-6 — How to use `SimulinkRealTime.target.getPCIInfo` to get information about the PCI devices installed on the target computer. This method works if you have one UEI board in your system. The target computer BIOS assigns the IRQ number.
- "Guidelines for Multiple UEI Boards" on page 49-6 — How to get information about the PCI devices when you have two or more UEI boards configured in a master/slave configuration. This method requires you to restart the target computer 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 Simulink Real-Time kernel is not a PnP operating system; you must disable this feature or PCI devices do not work on the Simulink Real-Time software.

Single UEI Board

If you have one UEI board for your target computer, that board is the master.

1. Insert the UEI board in a slot of your target computer.
2. Restart the target computer.
3. On the development 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 **Simulink Real-Time 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 setting specifies that the UEI MFx board generates the interrupt.
7. For the **PCI slot (-1: autosearch)** parameter, enter the same PCI address as for the UEI MFx Frame block **PCI slot** parameter. Enter `-1` (for autodetection) if the UEI MFx board 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 computer.
2. Restart the target computer.
3. On the development 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 **Simulink Real-Time 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 setting specifies that the UEI MFx board generates the interrupt.
7. From a MATLAB Command Window running on the development computer, type

   ```
   tg = slrt;
   getPCIInfo(tg, '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)** 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 setting specifies that the UEI MFx board generates the interrupt.

4 For the **PCI slot (-1: autosearch)** parameter, enter the same PCI address as for the UEI MFx Frame block **PCI slot (-1: autosearch)** parameter.

   - Only enter `-1` (for autodetection) if the UEI MFx board is the only UEI board in the target system.
   - If more than one UEI board is 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` folder. To operate these models, set the required interrupt source in the simulation parameters dialog box.

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 a Simulink Real-Time scope. The model runs each time the board processes 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. When the rest of the model executes, sometimes a new frame of output data has not reached in the output signal. However, if you process the data in the Function-Call Subsystem, this latency is not an issue.

  Note: Even if the frame time of the board is the same as the model sample time, the times derive from different sources. Therefore, the times can 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, the model cannot execute at a faster rate than frame completion. The For Iterator block executes once per frame completion and loops through the data.

United Electronic Industries (UEI): Boards and Blocks

This topic describes the UEI groups of boards supported by the Simulink Real-Time product (`www.ueidaq.com`).

UEI PD2-MF 12-Bit Series
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
UEI PD2-MF 14-Bit Series
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
UEI PD2-MF 16-Bit Series
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
UEI PD2-MFS 12-Bit Series
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
UEI PD2-MFS 14-Bit Series
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
UEI PD2-MFS 16-Bit Series
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
UEI PDXI-MF 12-Bit Series
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
UEI PDXI-MF 14-Bit Series
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
UEI PDXI-MF 16-Bit Series
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
UEI PDXI-MFS 12-Bit Series
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
UEI PDXI-MFS 14-Bit Series
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
UEI PDXI-MFS 16-Bit Series

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
UEI PD2-AO Series
UEI PD2-AO Analog Output (D/A)
UEI PD2-AO Digital Input
UEI PD2-AO Digital Output
UEI PD2-DIO-64
UEI PD2-DIO-64 Digital Input
UEI PD2-DIO-64 Digital Output
UEI PD2-DIO-128
UEI PD2-DIO-128 Digital Input
UEI PD2-DIO-128 Digital Output
UEI PD2-DIO-128i
UEI PD2-DIO-128i Digital Input
UEI PD2-DIO-128i Digital Output
UEI PDL-DIO-64
UEI PDL-DIO-64 Digital Input
UEI PDL-DIO-64 Digital Output
UEI PDXI-DIO-64
UEI PDXI-DIO-64 Digital Input
UEI PDXI-DIO-64 Digital Output
UEI PDXI-AO Series
UEI PDXI-AO Analog Output (D/A)
UEI PDXI-AO Digital Input
UEI PDXI-AO Digital Output

UEI PD2-MF 12-Bit Series

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 boards provide:

- 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
- Gain ranges of 1–8 or 1–1000
- 2 analog output (D/A) channels (12-bit)
- 16 digital input channels
- 16 digital output channels

The Simulink Real-Time software does not support the counter/timers on these boards.

The Simulink Real-Time 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

Board Characteristics

Board type	PD2-MF-16-1M/12L PD2-MF-16-1M/12H

	PD2-MF-64-1M/12L PD2-MF-64-1M/12H PD2–MF-16–3M/12L PD2–MF-16–3M/12H PD2–MF-64–3M/12L PD2–MF-64–3M/12H
Manufacturer	United Electronic Industries (UEI)
Bus type	PCI
Access method	Memory mapped
Multiple block instance support	No
Multiple board support	Yes

See Also

Topics

"Specifying UEI Boards" on page 49-3

External Websites

www.ueidaq.com

UEI PD2-MF 12-Bit Series Analog Input (A/D)

PD2-MF 12-Bit Series Analog Input block

Library

Simulink Real-Time Library for UEI

Scaling Input to Output

I/O Module Input	Block Output 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 input channels. For example, use the first and third analog input (A/D) channels, enter

```
[1 3]
```

The channel numbers can occur in arbitrary order. Number the channels beginning with 1 even if this 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 vector (slow bit)

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.

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.

Sample time

Enter the base sample time or a multiple of the base sample time (`-1` means sample time is inherited).

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

Topics

"Specifying UEI Boards" on page 49-3

External Websites

www.ueidaq.com

UEI PD2-MF 12-Bit Series Frame Analog Input

PD2-MF 12-Bit Series Frame Analog Input block

Library

Simulink Real-Time Library for UEI

Scaling Input to Output

I/O Module Input	Block Output 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 that make up one scan. For example, to read the first and fifth channels, enter

```
[1 5]
```

The channel numbers can occur in arbitrary order. Number the channels beginning with 1 even if this 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 replicate the gain value across the channels in the scan. Available gains vary depending on board type. Allowable values depend on the board type.

- L (Low level) boards — Specify the gain as one of 1, 10, 100, or 1000.
- H (High level) boards — Specify the gain as one of 1, 2, 4, or 8.

- DG-option boards — Specify the gain as one of 1, 2, 5, or 10.

Mux settling time vector (slow bit)

Enter a vector that is the same length as the channel vector. Each vector element must be a `0` or `1`. If you specify `1`, the corresponding 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 Simulink Real-Time 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 Simulink Real-Time 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 acquisition delays are `3` microseconds, and two are `12` microseconds. With a **Frame time** value of `0.001` second, if *N* is greater than `38`, the data will not be acceptable. If you set the 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`

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 physical constraints in the I/O module, 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

```
total number of samples = FrameSize x 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` — Use `Frame` if you connect the output from this block to the input of a block that requires a frame, such as a Signal Processing block.
- `Vector` — Use `Vector` if you connect the output from this block to the input of a block that requires vector input, such as a real-time Scope block.

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. Rebuild the model when you change this value.

The **Frame size**, **Scan time**, and **Frame time** parameters are not independent, but are related by

Frametime = *Framesize* × *Scantime*.

After you specify two of the parameters, specify `-1` for the third parameter and the equation determines the third parameter.

Note: Because the Simulink Real-Time software uses this value to set 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. Better 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 computer when DMA is attempted. In these cases, reduce the **DMA burst size** value to 8, rebuild and rerun the real-time application. If the DMA works at 8, select 16 and try again.

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

Topics

“Specifying UEI Boards” on page 49-3

External Websites

www.ueidaq.com

UEI PD2-MF 12-Bit Series Analog Output (D/A)

PD2-MF 12-Bit Series Analog Output block

Library

Simulink Real-Time Library for UEI

Scaling Input to Output

I/O Module 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 arbitrary order. Number the channels beginning with 1 even if this 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.

Sample time

Enter the base sample time or a multiple of the base sample time (`-1` means sample time is inherited).

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

Topics

“Specifying UEI Boards” on page 49-3

External Websites

www.ueidaq.com

UEI PD2-MF 12-Bit Series Digital Input

PD2-MF 12-Bit Series Digital Input block

Library

Simulink Real-Time Library for UEI

Scaling Input to Output

I/O Module 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 arbitrary order. Number the channels beginning with 1 even if this board manufacturer numbers them beginning with 0. The maximum allowable channel number is 16 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 (`-1` means sample time is inherited).

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

Topics

"Specifying UEI Boards" on page 49-3

External Websites

www.ueidaq.com

UEI PD2-MF 12-Bit Series Digital Output

PD2-MF 12-Bit Series Digital Output block

Library

Simulink Real-Time Library for UEI

Scaling Input to Output

I/O Module 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 arbitrary order. Number the channels beginning with 1 even if this board manufacturer numbers them beginning with 0. The maximum allowable channel number is 16, 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 (`-1` means sample time is inherited).

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

Topics

"Specifying UEI Boards" on page 49-3

External Websites

www.ueidaq.com

UEI PD2-MF 14-Bit Series

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 boards provide:

- 16 or 64 single or 8 or 32 differential analog input (A/D) channels (14-bit)
- Gain ranges of 1–8 or 1–1000
- 2 analog output (D/A) channels (12-bit)
- 16 digital input channels
- 16 digital output channels

The Simulink Real-Time software does not support the counter/timers on these boards.

The Simulink Real-Time 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

Board Characteristics

Board type	PD2-MF-16-400/14L PD2-MF-16-400/14H PD2-MF-64-400/14L

	PD2-MF-64-400/14H PD2-MF-16-2M/14H PD2-MF-64-2M/14H
Manufacturer	United Electronic Industries (UEI)
Bus type	PCI
Access method	Memory mapped
Multiple block instance support	No
Multiple board support	Yes

See Also

Topics

"Specifying UEI Boards" on page 49-3

External Websites

www.ueidaq.com

UEI PD2-MF 14-Bit Series Analog Input (A/D)

PD2-MF 14-Bit Series Analog Input block

Library

Simulink Real-Time Library for UEI

Scaling Input to Output

I/O Module Input	Block Output 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 input channels. For example, use the first and third analog input (A/D) channels, enter

```
[1 3]
```

The channel numbers can occur in arbitrary order. Number the channels beginning with 1 even if this 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 vector (slow bit)

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.

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.

Sample time

Enter the base sample time or a multiple of the base sample time (`-1` means sample time is inherited).

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

Topics

"Specifying UEI Boards" on page 49-3

External Websites

www.ueidaq.com

UEI PD2-MF 14-Bit Series Frame Analog Input

PD2-MF 14-Bit Series Frame Analog Input block

Library

Simulink Real-Time Library for UEI

Scaling Input to Output

I/O Module Input	Block Output 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 that make up one scan. For example, to read the first and fifth channels, enter

```
[1 5]
```

The channel numbers can occur in arbitrary order. Number the channels beginning with 1 even if this 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 replicate the gain value across the channels in the scan. Available gains vary depending on board type. Allowable values depend on the board type.

- L (Low level) boards — Specify the gain as one of 1, 10, 100, or 1000.
- H (High level) boards — Specify the gain as one of 1, 2, 4, or 8.

- DG-option boards — Specify the gain as one of 1, 2, 5, or 10.

Mux settling time vector (slow bit)

Enter a vector that is the same length as the channel vector. Each vector element must be a 0 or 1. If you specify 1, the corresponding 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 Simulink Real-Time 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 Simulink Real-Time 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 acquisition delays are 3 microseconds, and two are 12 microseconds. With a **Frame time** value of `0.001` second, if *N* is greater than 38, the data will not be acceptable. If you set the 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`

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 physical constraints in the I/O module, 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

```
total number of samples = FrameSize x 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` — Use `Frame` if you connect the output from this block to the input of a block that requires a frame, such as a Signal Processing block.
- `Vector` — Use `Vector` if you connect the output from this block to the input of a block that requires vector input, such as a real-time Scope block.

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. Rebuild the model when you change this value.

The **Frame size**, **Scan time**, and **Frame time** parameters are not independent, but are related by

Frametime = *Framesize* × *Scantime*.

After you specify two of the parameters, specify `-1` for the third parameter and the equation determines the third parameter.

Note: Because the Simulink Real-Time software uses this value to set 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. Better 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 computer when DMA is attempted. In these cases, reduce the **DMA burst size** value to 8, rebuild and rerun the real-time application. If the DMA works at 8, select 16 and try again.

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

Topics

“Specifying UEI Boards” on page 49-3

External Websites

www.ueidaq.com

UEI PD2-MF 14-Bit Series Analog Output (D/A)

PD2-MF 14-Bit Series Analog Output block

Library

Simulink Real-Time Library for UEI

Scaling Input to Output

I/O Module 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 arbitrary order. Number the channels beginning with 1 even if this 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.

Sample time

Enter the base sample time or a multiple of the base sample time (`-1` means sample time is inherited).

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

Topics

"Specifying UEI Boards" on page 49-3

External Websites

www.ueidaq.com

UEI PD2-MF 14-Bit Series Digital Input

PD2-MF 14-Bit Series Digital Input block

Library

Simulink Real-Time Library for UEI

Scaling Input to Output

I/O Module 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 arbitrary order. Number the channels beginning with 1 even if this board manufacturer numbers them beginning with 0. The maximum allowable channel number is 16 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 (`-1` means sample time is inherited).

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

Topics

"Specifying UEI Boards" on page 49-3

External Websites

www.ueidaq.com

UEI PD2-MF 14-Bit Series Digital Output

PD2-MF 14-Bit Series Digital Output block

Library

Simulink Real-Time Library for UEI

Scaling Input to Output

I/O Module 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 arbitrary order. Number the channels beginning with 1 even if this board manufacturer numbers them beginning with 0. The maximum allowable channel number is 16, 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 (`-1` means sample time is inherited).

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

Topics

"Specifying UEI Boards" on page 49-3

External Websites

www.ueidaq.com

UEI PD2-MF 16-Bit Series

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 boards provide:

- 16 or 64 single or 8 or 32 differential analog input (A/D) channels (16-bit)
- Gain ranges of 1–8 or 1–1000
- 2 analog output (D/A) channels (12-bit)
- 16 digital input channels
- 16 digital output channels

The Simulink Real-Time software does not support the counter/timers on these boards.

The Simulink Real-Time 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

Board Characteristics

Board type	PD2-MF-16-50/16H PD2-MF-16-333/16L PD2-MF-16-333/16H

	PD2-MF-64-333/16L PD2-MF-64-333/16H PD2-MF-16-500/16L PD2-MF-16-500/16H PD2-MF-64-500/16L PD2-MF-64-500/16H
Manufacturer	United Electronic Industries (UEI)
Bus type	PCI
Access method	Memory mapped
Multiple block instance support	No
Multiple board support	Yes

See Also

Topics

"Specifying UEI Boards" on page 49-3

External Websites

www.ueidaq.com

UEI PD2-MF 16-Bit Series Analog Input (A/D)

PD2-MF 16-Bit Series Analog Input block

Library

Simulink Real-Time Library for UEI

Scaling Input to Output

I/O Module Input	Block Output 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 input channels. For example, use the first and third analog input (A/D) channels, enter

```
[1 3]
```

The channel numbers can occur in arbitrary order. Number the channels beginning with 1 even if this 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 vector (slow bit)

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.

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.

Sample time

Enter the base sample time or a multiple of the base sample time (`-1` means sample time is inherited).

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

Topics

"Specifying UEI Boards" on page 49-3

External Websites

www.ueidaq.com

UEI PD2-MF 16-Bit Series Frame Analog Input

PD2-MF 16-Bit Series Frame Analog Input block

Library

Simulink Real-Time Library for UEI

Scaling Input to Output

I/O Module Input	Block Output 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 that make up one scan. For example, to read the first and fifth channels, enter

```
[1 5]
```

The channel numbers can occur in arbitrary order. Number the channels beginning with 1 even if this 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 replicate the gain value across the channels in the scan. Available gains vary depending on board type. Allowable values depend on the board type.

- L (Low level) boards — Specify the gain as one of 1, 10, 100, or 1000.
- H (High level) boards — Specify the gain as one of 1, 2, 4, or 8.

- DG-option boards — Specify the gain as one of 1, 2, 5, or 10.

Mux settling time vector (slow bit)

Enter a vector that is the same length as the channel vector. Each vector element must be a `0` or `1`. If you specify `1`, the corresponding 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 Simulink Real-Time 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 Simulink Real-Time 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 acquisition delays are `3` microseconds, and two are `12` microseconds. With a **Frame time** value of `0.001` second, if *N* is greater than `38`, the data will not be acceptable. If you set the 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`

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 physical constraints in the I/O module, 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

```
total number of samples = FrameSize x 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` — Use `Frame` if you connect the output from this block to the input of a block that requires a frame, such as a Signal Processing block.
- `Vector` — Use `Vector` if you connect the output from this block to the input of a block that requires vector input, such as a real-time Scope block.

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. Rebuild the model when you change this value.

The **Frame size**, **Scan time**, and **Frame time** parameters are not independent, but are related by

Frametime = *Framesize* × *Scantime*.

After you specify two of the parameters, specify `-1` for the third parameter and the equation determines the third parameter.

Note: Because the Simulink Real-Time software uses this value to set 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. Better 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 computer when DMA is attempted. In these cases, reduce the **DMA burst size** value to 8, rebuild and rerun the real-time application. If the DMA works at 8, select 16 and try again.

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

Topics

“Specifying UEI Boards” on page 49-3

External Websites

www.ueidaq.com

UEI PD2-MF 16-Bit Series Analog Output (D/A)

PD2-MF 16-Bit Series Analog Output block

Library

Simulink Real-Time Library for UEI

Scaling Input to Output

I/O Module 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 arbitrary order. Number the channels beginning with 1 even if this 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.

Sample time

Enter the base sample time or a multiple of the base sample time (`-1` means sample time is inherited).

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

Topics

"Specifying UEI Boards" on page 49-3

External Websites

www.ueidaq.com

UEI PD2-MF 16-Bit Series Digital Input

PD2-MF 16-Bit Series Digital Input block

Library

Simulink Real-Time Library for UEI

Scaling Input to Output

I/O Module 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 arbitrary order. Number the channels beginning with 1 even if this board manufacturer numbers them beginning with 0. The maximum allowable channel number is 16 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 (`-1` means sample time is inherited).

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

Topics

"Specifying UEI Boards" on page 49-3

External Websites

www.ueidaq.com

UEI PD2-MF 16-Bit Series Digital Output

PD2-MF 16-Bit Series Digital Output block

Library

Simulink Real-Time Library for UEI

Scaling Input to Output

I/O Module 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 arbitrary order. Number the channels beginning with 1 even if this board manufacturer numbers them beginning with 0. The maximum allowable channel number is 16, 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 (`-1` means sample time is inherited).

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

Topics

"Specifying UEI Boards" on page 49-3

External Websites

www.ueidaq.com

UEI PD2-MFS 12-Bit Series

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 boards provide:

- 4 or 8 single analog input (A/D) channels (12-bit)
- 2 analog output (D/A) channels (12-bit)
- 16 digital input channels
- 16 digital output channels

The Simulink Real-Time software does not support the counter/timers on these boards.

The Simulink Real-Time 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

Board Characteristics

Board type	PD2-MFS-4-1M/12 PD2-MFS-4-1M/12DG PD2-MFS-8-1M/12 PD2-MFS-8-1M/12DG

Manufacturer	United Electronic Industries (UEI)
Bus type	PCI
Access method	Memory mapped
Multiple block instance support	No
Multiple board support	Yes

See Also

Topics

"Specifying UEI Boards" on page 49-3

External Websites

www.ueidaq.com

UEI PD2-MFS 12-Bit Series Analog Input (A/D)

PD2-MFS 12-Bit Series Analog Input block

Library

Simulink Real-Time Library for UEI

Scaling Input to Output

I/O Module Input	Block Output 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 input channels. For example, use the first and third analog input (A/D) channels, enter

```
[1 3]
```

The channel numbers can occur in arbitrary order. Number the channels beginning with 1 even if this 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 vector (slow bit)

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.

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.

Sample time

Enter the base sample time or a multiple of the base sample time (`-1` means sample time is inherited).

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

Topics

"Specifying UEI Boards" on page 49-3

External Websites

www.ueidaq.com

UEI PD2-MFS 12-Bit Series Frame Analog Input

PD2-MFS 12-Bit Series Frame Analog Input block

Library

Simulink Real-Time Library for UEI

Scaling Input to Output

I/O Module Input	Block Output 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 that make up one scan. For example, to read the first and fifth channels, enter

```
[1 5]
```

The channel numbers can occur in arbitrary order. Number the channels beginning with 1 even if this 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 replicate the gain value across the channels in the scan. Available gains vary depending on board type. Allowable values depend on the board type.

- L (Low level) boards — Specify the gain as one of 1, 10, 100, or 1000.
- H (High level) boards — Specify the gain as one of 1, 2, 4, or 8.

- DG-option boards — Specify the gain as one of 1, 2, 5, or 10.

Mux settling time vector (slow bit)

Enter a vector that is the same length as the channel vector. Each vector element must be a 0 or 1. If you specify 1, the corresponding 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 Simulink Real-Time 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 Simulink Real-Time 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 acquisition delays are 3 microseconds, and two are 12 microseconds. With a **Frame time** value of `0.001` second, if *N* is greater than 38, the data will not be acceptable. If you set the 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`

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 physical constraints in the I/O module, 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

```
total number of samples = FrameSize x 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` — Use `Frame` if you connect the output from this block to the input of a block that requires a frame, such as a Signal Processing block.
- `Vector` — Use `Vector` if you connect the output from this block to the input of a block that requires vector input, such as a real-time Scope block.

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. Rebuild the model when you change this value.

The **Frame size**, **Scan time**, and **Frame time** parameters are not independent, but are related by

Frametime = *Framesize* × *Scantime*.

After you specify two of the parameters, specify `-1` for the third parameter and the equation determines the third parameter.

Note: Because the Simulink Real-Time software uses this value to set 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. Better 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 computer when DMA is attempted. In these cases, reduce the **DMA burst size** value to 8, rebuild and rerun the real-time application. If the DMA works at 8, select 16 and try again.

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

Topics

"Specifying UEI Boards" on page 49-3

External Websites

www.ueidaq.com

UEI PD2-MFS 12-Bit Series Analog Output (D/A)

PD2-MFS 12-Bit Series Analog Output block

Library

Simulink Real-Time Library for UEI

Scaling Input to Output

I/O Module 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 arbitrary order. Number the channels beginning with 1 even if this 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.

Sample time

Enter the base sample time or a multiple of the base sample time (`-1` means sample time is inherited).

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

Topics

"Specifying UEI Boards" on page 49-3

External Websites

www.ueidaq.com

UEI PD2-MFS 12-Bit Series Digital Input

PD2-MFS 12-Bit Series Digital Input block

Library

Simulink Real-Time Library for UEI

Scaling Input to Output

I/O Module 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 arbitrary order. Number the channels beginning with 1 even if this board manufacturer numbers them beginning with 0. The maximum allowable channel number is 16 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 (`-1` means sample time is inherited).

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

Topics

"Specifying UEI Boards" on page 49-3

External Websites

www.ueidaq.com

UEI PD2-MFS 12-Bit Series Digital Output

PD2-MFS 12-Bit Series Digital Output block

Library

Simulink Real-Time Library for UEI

Scaling Input to Output

I/O Module 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 arbitrary order. Number the channels beginning with 1 even if this board manufacturer numbers them beginning with 0. The maximum allowable channel number is 16, 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 (`-1` means sample time is inherited).

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

Topics

"Specifying UEI Boards" on page 49-3

External Websites

www.ueidaq.com

UEI PD2-MFS 14-Bit Series

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 boards provide:

- 4 or 8 single analog input (A/D) channels (14-bit)
- 2 analog output (D/A) channels (12-bit)
- 16 digital input channels
- 16 digital output channels

The Simulink Real-Time software does not support the counter/timers on these boards.

The Simulink Real-Time 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)
Bus type	PCI
Access method	Memory mapped
Multiple block instance support	No
Multiple board support	Yes

See Also

Topics

"Specifying UEI Boards" on page 49-3

External Websites

www.ueidaq.com

UEI PD2-MFS 14-Bit Series Analog Input (A/D)

PD2-MFS 14-Bit Series Analog Input block

Library

Simulink Real-Time Library for UEI

Scaling Input to Output

I/O Module Input	Block Output 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 input channels. For example, use the first and third analog input (A/D) channels, enter

```
[1 3]
```

The channel numbers can occur in arbitrary order. Number the channels beginning with 1 even if this 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 vector (slow bit)

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.

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.

Sample time

Enter the base sample time or a multiple of the base sample time (`-1` means sample time is inherited).

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

Topics

"Specifying UEI Boards" on page 49-3

External Websites

www.ueidaq.com

UEI PD2-MFS 14-Bit Series Frame Analog Input

PD2-MFS 14-Bit Series Frame Analog Input block

Library

Simulink Real-Time Library for UEI

Scaling Input to Output

I/O Module Input	Block Output 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 that make up one scan. For example, to read the first and fifth channels, enter

```
[1 5]
```

The channel numbers can occur in arbitrary order. Number the channels beginning with 1 even if this 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 replicate the gain value across the channels in the scan. Available gains vary depending on board type. Allowable values depend on the board type.

- L (Low level) boards — Specify the gain as one of 1, 10, 100, or 1000.
- H (High level) boards — Specify the gain as one of 1, 2, 4, or 8.

- DG-option boards — Specify the gain as one of 1, 2, 5, or 10.

Mux settling time vector (slow bit)

Enter a vector that is the same length as the channel vector. Each vector element must be a `0` or `1`. If you specify `1`, the corresponding 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 Simulink Real-Time 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 Simulink Real-Time 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 acquisition delays are `3` microseconds, and two are `12` microseconds. With a **Frame time** value of `0.001` second, if *N* is greater than `38`, the data will not be acceptable. If you set the 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`

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 physical constraints in the I/O module, 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

```
total number of samples = FrameSize x 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` — Use `Frame` if you connect the output from this block to the input of a block that requires a frame, such as a Signal Processing block.
- `Vector` — Use `Vector` if you connect the output from this block to the input of a block that requires vector input, such as a real-time Scope block.

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. Rebuild the model when you change this value.

The **Frame size**, **Scan time**, and **Frame time** parameters are not independent, but are related by

Frametime = *Framesize* × *Scantime*.

After you specify two of the parameters, specify `-1` for the third parameter and the equation determines the third parameter.

Note: Because the Simulink Real-Time software uses this value to set 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. Better 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 computer when DMA is attempted. In these cases, reduce the **DMA burst size** value to 8, rebuild and rerun the real-time application. If the DMA works at 8, select 16 and try again.

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

Topics

“Specifying UEI Boards” on page 49-3

External Websites

www.ueidaq.com

UEI PD2-MFS 14-Bit Series Analog Output (D/A)

PD2-MFS 14-Bit Series Analog Output block

Library

Simulink Real-Time Library for UEI

Scaling Input to Output

I/O Module 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 arbitrary order. Number the channels beginning with 1 even if this 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.

Sample time

Enter the base sample time or a multiple of the base sample time (`-1` means sample time is inherited).

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

Topics

“Specifying UEI Boards” on page 49-3

External Websites

www.ueidaq.com

UEI PD2-MFS 14-Bit Series Digital Input

PD2-MFS 14-Bit Series Digital Input block

Library

Simulink Real-Time Library for UEI

Scaling Input to Output

I/O Module 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 arbitrary order. Number the channels beginning with 1 even if this board manufacturer numbers them beginning with 0. The maximum allowable channel number is 16 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 (`-1` means sample time is inherited).

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

Topics

"Specifying UEI Boards" on page 49-3

External Websites

www.ueidaq.com

UEI PD2-MFS 14-Bit Series Digital Output

PD2-MFS 14-Bit Series Digital Output block

Library

Simulink Real-Time Library for UEI

Scaling Input to Output

I/O Module 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 arbitrary order. Number the channels beginning with 1 even if this board manufacturer numbers them beginning with 0. The maximum allowable channel number is 16, 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 (`-1` means sample time is inherited).

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

Topics

"Specifying UEI Boards" on page 49-3

External Websites

www.ueidaq.com

UEI PD2-MFS 16-Bit Series

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 boards provide:

- 4 or 8 single analog input (A/D) channels (16-bit)
- 2 analog output (D/A) channels (12-bit)
- 16 digital input channels
- 16 digital output channels

The Simulink Real-Time software does not support the counter/timers on these boards.

The Simulink Real-Time 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
Access method	Memory mapped
Multiple block instance support	No
Multiple board support	Yes

See Also

Topics

"Specifying UEI Boards" on page 49-3

External Websites

www.ueidaq.com

UEI PD2-MFS 16-Bit Series Analog Input (A/D)

PD2-MFS 16-Bit Series Analog Input block

Library

Simulink Real-Time Library for UEI

Scaling Input to Output

I/O Module Input	Block Output 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 input channels. For example, use the first and third analog input (A/D) channels, enter

```
[1 3]
```

The channel numbers can occur in arbitrary order. Number the channels beginning with 1 even if this 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 vector (slow bit)

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.

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.

Sample time

Enter the base sample time or a multiple of the base sample time (`-1` means sample time is inherited).

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

Topics

"Specifying UEI Boards" on page 49-3

External Websites

www.ueidaq.com

UEI PD2-MFS 16-Bit Series Frame Analog Input

PD2-MFS 16-Bit Series Frame Analog Input block

Library

Simulink Real-Time Library for UEI

Scaling Input to Output

I/O Module Input	Block Output 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 that make up one scan. For example, to read the first and fifth channels, enter

```
[1 5]
```

The channel numbers can occur in arbitrary order. Number the channels beginning with 1 even if this 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 replicate the gain value across the channels in the scan. Available gains vary depending on board type. Allowable values depend on the board type.

- L (Low level) boards — Specify the gain as one of 1, 10, 100, or 1000.
- H (High level) boards — Specify the gain as one of 1, 2, 4, or 8.

- DG-option boards — Specify the gain as one of 1, 2, 5, or 10.

Mux settling time vector (slow bit)

Enter a vector that is the same length as the channel vector. Each vector element must be a 0 or 1. If you specify 1, the corresponding 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 Simulink Real-Time 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 Simulink Real-Time 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 acquisition delays are 3 microseconds, and two are 12 microseconds. With a **Frame time** value of `0.001` second, if *N* is greater than 38, the data will not be acceptable. If you set the 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`

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 physical constraints in the I/O module, 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

```
total number of samples = FrameSize x 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` — Use `Frame` if you connect the output from this block to the input of a block that requires a frame, such as a Signal Processing block.
- `Vector` — Use `Vector` if you connect the output from this block to the input of a block that requires vector input, such as a real-time Scope block.

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. Rebuild the model when you change this value.

The **Frame size**, **Scan time**, and **Frame time** parameters are not independent, but are related by

Frametime = *Framesize* × *Scantime*.

After you specify two of the parameters, specify `-1` for the third parameter and the equation determines the third parameter.

Note: Because the Simulink Real-Time software uses this value to set 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. Better 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 computer when DMA is attempted. In these cases, reduce the **DMA burst size** value to 8, rebuild and rerun the real-time application. If the DMA works at 8, select 16 and try again.

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

Topics

"Specifying UEI Boards" on page 49-3

External Websites

www.ueidaq.com

UEI PD2-MFS 16-Bit Series Analog Output (D/A)

PD2-MFS 16-Bit Series Analog Output block

Library

Simulink Real-Time Library for UEI

Scaling Input to Output

I/O Module 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 arbitrary order. Number the channels beginning with 1 even if this 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.

Sample time

Enter the base sample time or a multiple of the base sample time (`-1` means sample time is inherited).

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

Topics

"Specifying UEI Boards" on page 49-3

External Websites

www.ueidaq.com

UEI PD2-MFS 16-Bit Series Digital Input

PD2-MFS 16-Bit Series Digital Input block

Library

Simulink Real-Time Library for UEI

Scaling Input to Output

I/O Module 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 arbitrary order. Number the channels beginning with 1 even if this board manufacturer numbers them beginning with 0. The maximum allowable channel number is 16 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 (`-1` means sample time is inherited).

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

Topics

"Specifying UEI Boards" on page 49-3

External Websites

www.ueidaq.com

UEI PD2-MFS 16-Bit Series Digital Output

PD2-MFS 16-Bit Series Digital Output

Library

Simulink Real-Time Library for UEI

Scaling Input to Output

I/O Module 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 arbitrary order. Number the channels beginning with 1 even if this board manufacturer numbers them beginning with 0. The maximum allowable channel number is 16, 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 (`-1` means sample time is inherited).

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

Topics

"Specifying UEI Boards" on page 49-3

External Websites

www.ueidaq.com

UEI PDXI-MF 12-Bit Series

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 boards provide:

- 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
- Gain ranges of 1–8 or 1–1000
- 2 analog output (D/A) channels (12-bit)
- 16 digital input channels
- 16 digital output channels

The Simulink Real-Time software does not support the counter/timers on these boards.

The Simulink Real-Time 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)
Bus type	CompactPCI, PXI
Access method	Memory mapped
Multiple block instance support	No
Multiple board support	Yes

See Also

Topics

"Specifying UEI Boards" on page 49-3

External Websites

www.ueidaq.com

UEI PDXI-MF 12-Bit Series Analog Input (A/D)

PDXI-MF 12-Bit Series Analog Input block

Library

Simulink Real-Time Library for UEI

Scaling Input to Output

I/O Module Input	Block Output 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 input channels. For example, use the first and third analog input (A/D) channels, enter

```
[1 3]
```

The channel numbers can occur in arbitrary order. Number the channels beginning with 1 even if this 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 vector (slow bit)

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.

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.

Sample time

Enter the base sample time or a multiple of the base sample time (`-1` means sample time is inherited).

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

Topics

"Specifying UEI Boards" on page 49-3

External Websites

www.ueidaq.com

UEI PDXI-MF 12-Bit Series Frame Analog Input

PDXI-MF 12-Bit Series Frame Analog Input block

Library

Simulink Real-Time Library for UEI

Scaling Input to Output

I/O Module Input	Block Output 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 that make up one scan. For example, to read the first and fifth channels, enter

```
[1 5]
```

The channel numbers can occur in arbitrary order. Number the channels beginning with 1 even if this 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 replicate the gain value across the channels in the scan. Available gains vary depending on board type. Allowable values depend on the board type.

- L (Low level) boards — Specify the gain as one of 1, 10, 100, or 1000.
- H (High level) boards — Specify the gain as one of 1, 2, 4, or 8.

- DG-option boards — Specify the gain as one of 1, 2, 5, or 10.

Mux settling time vector (slow bit)

Enter a vector that is the same length as the channel vector. Each vector element must be a `0` or `1`. If you specify `1`, the corresponding 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 Simulink Real-Time 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 Simulink Real-Time 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 acquisition delays are `3` microseconds, and two are `12` microseconds. With a **Frame time** value of `0.001` second, if *N* is greater than `38`, the data will not be acceptable. If you set the 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`

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 physical constraints in the I/O module, 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

```
total number of samples = FrameSize x 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` — Use `Frame` if you connect the output from this block to the input of a block that requires a frame, such as a Signal Processing block.
- `Vector` — Use `Vector` if you connect the output from this block to the input of a block that requires vector input, such as a real-time Scope block.

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. Rebuild the model when you change this value.

The **Frame size**, **Scan time**, and **Frame time** parameters are not independent, but are related by

Frametime = *Framesize* × *Scantime*.

After you specify two of the parameters, specify `-1` for the third parameter and the equation determines the third parameter.

Note: Because the Simulink Real-Time software uses this value to set 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. Better 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 computer when DMA is attempted. In these cases, reduce the **DMA burst size** value to 8, rebuild and rerun the real-time application. If the DMA works at 8, select 16 and try again.

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

Topics

"Specifying UEI Boards" on page 49-3

External Websites

www.ueidaq.com

UEI PDXI-MF 12-Bit Series Analog Output (D/A)

PDXI-MF 12-Bit Series Analog Output block

Library

Simulink Real-Time Library for UEI

Scaling Input to Output

I/O Module 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 arbitrary order. Number the channels beginning with 1 even if this 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.

Sample time

Enter the base sample time or a multiple of the base sample time (`-1` means sample time is inherited).

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

Topics

"Specifying UEI Boards" on page 49-3

External Websites

www.ueidaq.com

UEI PDXI-MF 12-Bit Series Digital Input

PDXI-MF 12-Bit Series Digital Input block

Library

Simulink Real-Time Library for UEI

Scaling Input to Output

I/O Module 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 arbitrary order. Number the channels beginning with 1 even if this board manufacturer numbers them beginning with 0. The maximum allowable channel number is 16 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 (`-1` means sample time is inherited).

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

Topics

"Specifying UEI Boards" on page 49-3

External Websites

www.ueidaq.com

UEI PDXI-MF 12-Bit Series Digital Output

PDXI-MF 12-Bit Series Digital Output block

Library

Simulink Real-Time Library for UEI

Scaling Input to Output

I/O Module 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 arbitrary order. Number the channels beginning with 1 even if this board manufacturer numbers them beginning with 0. The maximum allowable channel number is 16, 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 (`-1` means sample time is inherited).

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

Topics

"Specifying UEI Boards" on page 49-3

External Websites

www.ueidaq.com

UEI PDXI-MF 14-Bit Series

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 boards provide:

- 16 or 64 single or 8 or 32 differential analog input (A/D) channels (14-bit)
- 2 analog output (D/A) channels (12-bit)
- 16 digital input channels
- 16 digital output channels

The Simulink Real-Time software does not support the counter/timers on these boards.

The Simulink Real-Time 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
Multiple block instance support	No
Multiple board support	Yes

See Also

Topics

"Specifying UEI Boards" on page 49-3

External Websites

www.ueidaq.com

UEI PDXI-MF 14-Bit Series Analog Input (A/D)

PDXI-MF 14-Bit Series Analog Input block

Library

Simulink Real-Time Library for UEI

Scaling Input to Output

I/O Module Input	Block Output 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 input channels. For example, use the first and third analog input (A/D) channels, enter

```
[1 3]
```

The channel numbers can occur in arbitrary order. Number the channels beginning with 1 even if this 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 vector (slow bit)

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.

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.

Sample time

Enter the base sample time or a multiple of the base sample time (`-1` means sample time is inherited).

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

Topics

"Specifying UEI Boards" on page 49-3

External Websites

www.ueidaq.com

UEI PDXI-MF 14-Bit Series Frame Analog Input

PDXI-MF 14-Bit Series Frame Analog Input block

Library

Simulink Real-Time Library for UEI

Scaling Input to Output

I/O Module Input	Block Output 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 that make up one scan. For example, to read the first and fifth channels, enter

```
[1 5]
```

The channel numbers can occur in arbitrary order. Number the channels beginning with 1 even if this 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 replicate the gain value across the channels in the scan. Available gains vary depending on board type. Allowable values depend on the board type.

- L (Low level) boards — Specify the gain as one of 1, 10, 100, or 1000.
- H (High level) boards — Specify the gain as one of 1, 2, 4, or 8.

- DG-option boards — Specify the gain as one of 1, 2, 5, or 10.

Mux settling time vector (slow bit)

Enter a vector that is the same length as the channel vector. Each vector element must be a 0 or 1. If you specify 1, the corresponding 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 Simulink Real-Time 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 Simulink Real-Time 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 acquisition delays are 3 microseconds, and two are 12 microseconds. With a **Frame time** value of `0.001` second, if *N* is greater than 38, the data will not be acceptable. If you set the 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`

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 physical constraints in the I/O module, 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

```
total number of samples = FrameSize x 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` — Use `Frame` if you connect the output from this block to the input of a block that requires a frame, such as a Signal Processing block.
- `Vector` — Use `Vector` if you connect the output from this block to the input of a block that requires vector input, such as a real-time Scope block.

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. Rebuild the model when you change this value.

The **Frame size**, **Scan time**, and **Frame time** parameters are not independent, but are related by

Frametime = *Framesize* × *Scantime*.

After you specify two of the parameters, specify `-1` for the third parameter and the equation determines the third parameter.

Note: Because the Simulink Real-Time software uses this value to set 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. Better 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 computer when DMA is attempted. In these cases, reduce the **DMA burst size** value to 8, rebuild and rerun the real-time application. If the DMA works at 8, select 16 and try again.

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

Topics

"Specifying UEI Boards" on page 49-3

External Websites

www.ueidaq.com

UEI PDXI-MF 14-Bit Series Analog Output (D/A)

PDXI-MF 14-Bit Series Analog Output block

Library

Simulink Real-Time Library for UEI

Scaling Input to Output

I/O Module 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 arbitrary order. Number the channels beginning with 1 even if this 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.

Sample time

Enter the base sample time or a multiple of the base sample time (`-1` means sample time is inherited).

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

Topics

"Specifying UEI Boards" on page 49-3

External Websites

www.ueidaq.com

UEI PDXI-MF 14-Bit Series Digital Input

PDXI-MF 14-Bit Series Digital Input block

Library

Simulink Real-Time Library for UEI

Scaling Input to Output

I/O Module 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 arbitrary order. Number the channels beginning with 1 even if this board manufacturer numbers them beginning with 0. The maximum allowable channel number is 16 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 (`-1` means sample time is inherited).

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

Topics

"Specifying UEI Boards" on page 49-3

External Websites

www.ueidaq.com

UEI PDXI-MF 14-Bit Series Digital Output

PDXI-MF 14-Bit Series Digital Output block

Library

Simulink Real-Time Library for UEI

Scaling Input to Output

I/O Module 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 arbitrary order. Number the channels beginning with 1 even if this board manufacturer numbers them beginning with 0. The maximum allowable channel number is 16, 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 (`-1` means sample time is inherited).

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

Topics

"Specifying UEI Boards" on page 49-3

External Websites

www.ueidaq.com

UEI PDXI-MF 16-Bit Series

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 boards provide:

- 16 or 64 single or 8 or 32 differential analog input (A/D) channels (16-bit)
- 2 analog output (D/A) channels (12-bit)
- 16 digital input channels
- 16 digital output channels

The Simulink Real-Time software does not support the counter/timers on these boards.

The Simulink Real-Time 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
Multiple block instance support	No
Multiple board support	Yes

See Also

Topics

"Specifying UEI Boards" on page 49-3

External Websites

www.ueidaq.com

UEI PDXI-MF 16-Bit Series Analog Input (A/D)

PDXI-MF 16-Bit Series Analog Input block

Library

Simulink Real-Time Library for UEI

Scaling Input to Output

I/O Module Input	Block Output 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 input channels. For example, use the first and third analog input (A/D) channels, enter

```
[1 3]
```

The channel numbers can occur in arbitrary order. Number the channels beginning with 1 even if this 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 vector (slow bit)

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.

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.

Sample time

Enter the base sample time or a multiple of the base sample time (`-1` means sample time is inherited).

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

Topics

"Specifying UEI Boards" on page 49-3

External Websites

www.ueidaq.com

UEI PDXI-MF 16-Bit Series Frame Analog Input

PDXI-MF 16-Bit Series Frame Analog Input block

Library

Simulink Real-Time Library for UEI

Scaling Input to Output

I/O Module Input	Block Output 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 that make up one scan. For example, to read the first and fifth channels, enter

```
[1 5]
```

The channel numbers can occur in arbitrary order. Number the channels beginning with 1 even if this 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 replicate the gain value across the channels in the scan. Available gains vary depending on board type. Allowable values depend on the board type.

- L (Low level) boards — Specify the gain as one of 1, 10, 100, or 1000.
- H (High level) boards — Specify the gain as one of 1, 2, 4, or 8.

- DG-option boards — Specify the gain as one of 1, 2, 5, or 10.

Mux settling time vector (slow bit)

Enter a vector that is the same length as the channel vector. Each vector element must be a `0` or `1`. If you specify `1`, the corresponding 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 Simulink Real-Time 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 Simulink Real-Time 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 acquisition delays are `3` microseconds, and two are `12` microseconds. With a **Frame time** value of `0.001` second, if *N* is greater than `38`, the data will not be acceptable. If you set the 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`

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 physical constraints in the I/O module, 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

```
total number of samples = FrameSize x 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` — Use `Frame` if you connect the output from this block to the input of a block that requires a frame, such as a Signal Processing block.
- `Vector` — Use `Vector` if you connect the output from this block to the input of a block that requires vector input, such as a real-time Scope block.

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. Rebuild the model when you change this value.

The **Frame size**, **Scan time**, and **Frame time** parameters are not independent, but are related by

Frametime = *Framesize* × *Scantime*.

After you specify two of the parameters, specify `-1` for the third parameter and the equation determines the third parameter.

Note: Because the Simulink Real-Time software uses this value to set 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. Better 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 computer when DMA is attempted. In these cases, reduce the **DMA burst size** value to 8, rebuild and rerun the real-time application. If the DMA works at 8, select 16 and try again.

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

Topics

"Specifying UEI Boards" on page 49-3

External Websites

www.ueidaq.com

UEI PDXI-MF 16-Bit Series Analog Output (D/A)

PDXI-MF 16-Bit Series Analog Output block

Library

Simulink Real-Time Library for UEI

Scaling Input to Output

I/O Module 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 arbitrary order. Number the channels beginning with 1 even if this 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.

Sample time

Enter the base sample time or a multiple of the base sample time (`-1` means sample time is inherited).

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

Topics

"Specifying UEI Boards" on page 49-3

External Websites

www.ueidaq.com

UEI PDXI-MF 16-Bit Series Digital Input

PDXI-MF 16-Bit Series Digital Input block

Library

Simulink Real-Time Library for UEI

Scaling Input to Output

I/O Module 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 arbitrary order. Number the channels beginning with 1 even if this board manufacturer numbers them beginning with 0. The maximum allowable channel number is 16 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 (`-1` means sample time is inherited).

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

Topics

"Specifying UEI Boards" on page 49-3

External Websites

www.ueidaq.com

UEI PDXI-MF 16-Bit Series Digital Output

PDXI-MF 16-Bit Series Digital Output block

Library

Simulink Real-Time Library for UEI

Scaling Input to Output

I/O Module 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 arbitrary order. Number the channels beginning with 1 even if this board manufacturer numbers them beginning with 0. The maximum allowable channel number is 16, 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 (`-1` means sample time is inherited).

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

Topics

"Specifying UEI Boards" on page 49-3

External Websites

www.ueidaq.com

UEI PDXI-MFS 12-Bit Series

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 that provide:

- 4 or 8 single analog input (A/D) channels (12-bit)
- 2 analog output (D/A) channels (12-bit)
- 16 digital input lines
- 16 digital output lines

The Simulink Real-Time software does not support the counter/timers on these boards.

The Simulink Real-Time 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

See Also

Topics

"Specifying UEI Boards" on page 49-3

External Websites

www.ueidaq.com

UEI PDXI-MFS 12-Bit Series Analog Input (A/D)

PDXI-MFS 12-Bit Series Analog Input block

Library

Simulink Real-Time Library for UEI

Scaling Input to Output

I/O Module Input	Block Output 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 input channels. For example, use the first and third analog input (A/D) channels, enter

```
[1 3]
```

The channel numbers can occur in arbitrary order. Number the channels beginning with 1 even if this 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 vector (slow bit)

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.

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.

Sample time

Enter the base sample time or a multiple of the base sample time (`-1` means sample time is inherited).

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

Topics

"Specifying UEI Boards" on page 49-3

External Websites

www.ueidaq.com

UEI PDXI-MFS 12-Bit Series Frame Analog Input

PDXI-MFS 12-Bit Series Frame Analog Input block

Library

Simulink Real-Time Library for UEI

Scaling Input to Output

I/O Module Input	Block Output 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 that make up one scan. For example, to read the first and fifth channels, enter

```
[1 5]
```

The channel numbers can occur in arbitrary order. Number the channels beginning with 1 even if this 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 replicate the gain value across the channels in the scan. Available gains vary depending on board type. Allowable values depend on the board type.

- L (Low level) boards — Specify the gain as one of 1, 10, 100, or 1000.
- H (High level) boards — Specify the gain as one of 1, 2, 4, or 8.

- DG-option boards — Specify the gain as one of 1, 2, 5, or 10.

Mux settling time vector (slow bit)

Enter a vector that is the same length as the channel vector. Each vector element must be a 0 or 1. If you specify 1, the corresponding 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 Simulink Real-Time 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 Simulink Real-Time 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 acquisition delays are 3 microseconds, and two are 12 microseconds. With a **Frame time** value of `0.001` second, if *N* is greater than 38, the data will not be acceptable. If you set the 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`

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 physical constraints in the I/O module, 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

```
total number of samples = FrameSize x 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` — Use `Frame` if you connect the output from this block to the input of a block that requires a frame, such as a Signal Processing block.
- `Vector` — Use `Vector` if you connect the output from this block to the input of a block that requires vector input, such as a real-time Scope block.

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. Rebuild the model when you change this value.

The **Frame size**, **Scan time**, and **Frame time** parameters are not independent, but are related by

Frametime = *Framesize* × *Scantime*.

After you specify two of the parameters, specify `-1` for the third parameter and the equation determines the third parameter.

Note: Because the Simulink Real-Time software uses this value to set 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. Better 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 computer when DMA is attempted. In these cases, reduce the **DMA burst size** value to 8, rebuild and rerun the real-time application. If the DMA works at 8, select 16 and try again.

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

Topics

“Specifying UEI Boards” on page 49-3

External Websites

www.ueidaq.com

UEI PDXI-MFS 12-Bit Series Analog Output (D/A)

PDXI-MFS 12-Bit Series Analog Output block

Library

Simulink Real-Time Library for UEI

Scaling Input to Output

I/O Module 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 arbitrary order. Number the channels beginning with 1 even if this 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.

Sample time

Enter the base sample time or a multiple of the base sample time (`-1` means sample time is inherited).

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

Topics

"Specifying UEI Boards" on page 49-3

External Websites

www.ueidaq.com

UEI PDXI-MFS 12-Bit Series Digital Input

PDXI-MFS 12-Bit Series Digital Input block

Library

Simulink Real-Time Library for UEI

Scaling Input to Output

I/O Module 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 arbitrary order. Number the channels beginning with 1 even if this board manufacturer numbers them beginning with 0. The maximum allowable channel number is 16 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 (`-1` means sample time is inherited).

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

Topics

"Specifying UEI Boards" on page 49-3

External Websites

www.ueidaq.com

UEI PDXI-MFS 12-Bit Series Digital Output

PDXI-MFS 12-Bit Series Digital Output block

Library

Simulink Real-Time Library for UEI

Scaling Input to Output

I/O Module 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 arbitrary order. Number the channels beginning with 1 even if this board manufacturer numbers them beginning with 0. The maximum allowable channel number is 16, 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 (`-1` means sample time is inherited).

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

Topics

"Specifying UEI Boards" on page 49-3

External Websites

www.ueidaq.com

UEI PDXI-MFS 14-Bit Series

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 boards provide:

- 4 or 8 single analog input (A/D) channels (14-bit)
- 2 analog output (D/A) channels (12-bit)
- 16 digital input channels
- 16 digital output channels

The Simulink Real-Time software does not support the counter/timers on these boards.

The Simulink Real-Time 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 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

See Also

Topics

"Specifying UEI Boards" on page 49-3

External Websites

www.ueidaq.com

UEI PDXI-MFS 14-Bit Series Analog Input (A/D)

PDXI-MFS 14-Bit Series Analog Input block

Library

Simulink Real-Time Library for UEI

Scaling Input to Output

I/O Module Input	Block Output 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 input channels. For example, use the first and third analog input (A/D) channels, enter

```
[1 3]
```

The channel numbers can occur in arbitrary order. Number the channels beginning with 1 even if this 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 vector (slow bit)

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.

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.

Sample time

Enter the base sample time or a multiple of the base sample time (`-1` means sample time is inherited).

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

Topics

"Specifying UEI Boards" on page 49-3

External Websites

www.ueidaq.com

UEI PDXI-MFS 14-Bit Series Frame Analog Input

PDXI-MFS 14-Bit Series Frame Analog Input block

Library

Simulink Real-Time Library for UEI

Scaling Input to Output

I/O Module Input	Block Output 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 that make up one scan. For example, to read the first and fifth channels, enter

```
[1 5]
```

The channel numbers can occur in arbitrary order. Number the channels beginning with 1 even if this 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 replicate the gain value across the channels in the scan. Available gains vary depending on board type. Allowable values depend on the board type.

- L (Low level) boards — Specify the gain as one of 1, 10, 100, or 1000.
- H (High level) boards — Specify the gain as one of 1, 2, 4, or 8.

- DG-option boards — Specify the gain as one of 1, 2, 5, or 10.

Mux settling time vector (slow bit)

Enter a vector that is the same length as the channel vector. Each vector element must be a `0` or `1`. If you specify `1`, the corresponding 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 Simulink Real-Time 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 Simulink Real-Time 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 acquisition delays are `3` microseconds, and two are `12` microseconds. With a **Frame time** value of `0.001` second, if *N* is greater than `38`, the data will not be acceptable. If you set the 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`

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 physical constraints in the I/O module, 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

```
total number of samples = FrameSize x 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` — Use `Frame` if you connect the output from this block to the input of a block that requires a frame, such as a Signal Processing block.
- `Vector` — Use `Vector` if you connect the output from this block to the input of a block that requires vector input, such as a real-time Scope block.

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. Rebuild the model when you change this value.

The **Frame size**, **Scan time**, and **Frame time** parameters are not independent, but are related by

Frametime = *Framesize* × *Scantime*.

After you specify two of the parameters, specify `-1` for the third parameter and the equation determines the third parameter.

Note: Because the Simulink Real-Time software uses this value to set 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. Better 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 computer when DMA is attempted. In these cases, reduce the **DMA burst size** value to 8, rebuild and rerun the real-time application. If the DMA works at 8, select 16 and try again.

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

Topics

“Specifying UEI Boards” on page 49-3

External Websites

www.ueidaq.com

UEI PDXI-MFS 14-Bit Series Analog Output (D/A)

PDXI-MFS 14-Bit Series Analog Output block

Library

Simulink Real-Time Library for UEI

Scaling Input to Output

I/O Module 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 arbitrary order. Number the channels beginning with 1 even if this 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.

Sample time

Enter the base sample time or a multiple of the base sample time (`-1` means sample time is inherited).

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

Topics

“Specifying UEI Boards” on page 49-3

External Websites

www.ueidaq.com

UEI PDXI-MFS 14-Bit Series Digital Input

PDXI-MFS 14-Bit Series Digital Input block

Library

Simulink Real-Time Library for UEI

Scaling Input to Output

I/O Module 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 arbitrary order. Number the channels beginning with 1 even if this board manufacturer numbers them beginning with 0. The maximum allowable channel number is 16 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 (`-1` means sample time is inherited).

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

Topics

"Specifying UEI Boards" on page 49-3

External Websites

www.ueidaq.com

UEI PDXI-MFS 14-Bit Series Digital Output

PDXI-MFS 14-Bit Series Digital Output block

Library

Simulink Real-Time Library for UEI

Scaling Input to Output

I/O Module 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 arbitrary order. Number the channels beginning with 1 even if this board manufacturer numbers them beginning with 0. The maximum allowable channel number is 16, 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 (`-1` means sample time is inherited).

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

Topics

"Specifying UEI Boards" on page 49-3

External Websites

www.ueidaq.com

UEI PDXI-MFS 16-Bit Series

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 boards provide:

- 4 or 8 single analog input (A/D) channels (16-bit)
- 2 analog output (D/A) channels (12-bit)
- 16 digital input channels
- 16 digital output channels

The Simulink Real-Time software does not support the counter/timers on these boards.

The Simulink Real-Time 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

Board Characteristics

Board type	PDXI-MFS-4-300/16 PDXI-MFS-4-300/16DG PDXI-MFS-8-300/16 PDXI-MFS-8-300/16DG

	PDXI-MFS-4-500/16 PDXI-MFS-4-500/16DG PDXI-MFS-8-500/16 PDXI-MFS-8-500/16DG
Manufacturer	United Electronic Industries (UEI)
Bus type	CompactPCI, PXI
Access method	Memory mapped
Multiple block instance support	No
Multiple board support	Yes

See Also

Topics

"Specifying UEI Boards" on page 49-3

External Websites

www.ueidaq.com

UEI PDXI-MFS 16-Bit Series Analog Input (A/D)

PDXI-MFS 16-Bit Series Analog Input block

Library

Simulink Real-Time Library for UEI

Scaling Input to Output

I/O Module Input	Block Output 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 input channels. For example, use the first and third analog input (A/D) channels, enter

```
[1 3]
```

The channel numbers can occur in arbitrary order. Number the channels beginning with 1 even if this 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 vector (slow bit)

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.

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.

Sample time

Enter the base sample time or a multiple of the base sample time (`-1` means sample time is inherited).

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

Topics

"Specifying UEI Boards" on page 49-3

External Websites

www.ueidaq.com

UEI PDXI-MFS 16-Bit Series Frame Analog Input

PDXI-MFS 16-Bit Series Frame Analog Input block

Library

Simulink Real-Time Library for UEI

Scaling Input to Output

I/O Module Input	Block Output 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 that make up one scan. For example, to read the first and fifth channels, enter

```
[1 5]
```

The channel numbers can occur in arbitrary order. Number the channels beginning with 1 even if this 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 replicate the gain value across the channels in the scan. Available gains vary depending on board type. Allowable values depend on the board type.

- L (Low level) boards — Specify the gain as one of 1, 10, 100, or 1000.
- H (High level) boards — Specify the gain as one of 1, 2, 4, or 8.

- DG-option boards — Specify the gain as one of 1, 2, 5, or 10.

Mux settling time vector (slow bit)

Enter a vector that is the same length as the channel vector. Each vector element must be a 0 or 1. If you specify 1, the corresponding 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 Simulink Real-Time 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 Simulink Real-Time 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 acquisition delays are 3 microseconds, and two are 12 microseconds. With a **Frame time** value of `0.001` second, if *N* is greater than 38, the data will not be acceptable. If you set the 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`

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 physical constraints in the I/O module, 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

```
total number of samples = FrameSize x 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` — Use `Frame` if you connect the output from this block to the input of a block that requires a frame, such as a Signal Processing block.
- `Vector` — Use `Vector` if you connect the output from this block to the input of a block that requires vector input, such as a real-time Scope block.

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. Rebuild the model when you change this value.

The **Frame size**, **Scan time**, and **Frame time** parameters are not independent, but are related by

Frametime = *Framesize* × *Scantime*.

After you specify two of the parameters, specify `-1` for the third parameter and the equation determines the third parameter.

Note: Because the Simulink Real-Time software uses this value to set 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. Better 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 computer when DMA is attempted. In these cases, reduce the **DMA burst size** value to 8, rebuild and rerun the real-time application. If the DMA works at 8, select 16 and try again.

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

Topics

“Specifying UEI Boards” on page 49-3

External Websites

www.ueidaq.com

UEI PDXI-MFS 16-Bit Series Analog Output (D/A)

PDXI-MFS 16-Bit Series Analog Output block

Library

Simulink Real-Time Library for UEI

Scaling Input to Output

I/O Module 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 arbitrary order. Number the channels beginning with 1 even if this 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.

Sample time

Enter the base sample time or a multiple of the base sample time (`-1` means sample time is inherited).

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

Topics

"Specifying UEI Boards" on page 49-3

External Websites

www.ueidaq.com

UEI PDXI-MFS 16-Bit Series Digital Input

PDXI-MFS 16-Bit Series Digital Input block

Library

Simulink Real-Time Library for UEI

Scaling Input to Output

I/O Module 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 arbitrary order. Number the channels beginning with 1 even if this board manufacturer numbers them beginning with 0. The maximum allowable channel number is 16 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 (`-1` means sample time is inherited).

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

Topics

"Specifying UEI Boards" on page 49-3

External Websites

www.ueidaq.com

UEI PDXI-MFS 16-Bit Series Digital Output

PDXI-MFS 16-Bit Series Digital Output block

Library

Simulink Real-Time Library for UEI

Scaling Input to Output

I/O Module 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 arbitrary order. Number the channels beginning with 1 even if this board manufacturer numbers them beginning with 0. The maximum allowable channel number is 16, 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 (`-1` means sample time is inherited).

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

Topics

"Specifying UEI Boards" on page 49-3

External Websites

www.ueidaq.com

UEI PD2-AO Series

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 provide:

- 8, 16, 32, or 96 analog output (D/A) channels (16-bit) with a maximum sample rate of 100 kHz per channel
- 8 digital input channels
- 8 digital output channels

The Simulink Real-Time 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

Board Characteristics

Board type	PD2-AO-8/16 PD2-AO-16/16 PD2-AO-32/16 PD2-AO-96/16
Manufacturer	United Electronic Industries (UEI)
Bus type	PCI
Access method	Memory mapped

Multiple block instance support	No
Multiple board support	Yes

See Also

Topics

"Specifying UEI Boards" on page 49-3

External Websites

www.ueidaq.com

UEI PD2-AO Analog Output (D/A)

PD2-AO Analog Output block

Library

Simulink Real-Time Library for UEI

Scaling Input to Output

I/O Module 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 arbitrary order. Number the channels beginning with 1 even if this 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, the board does not produce output.

As a workaround, enter a contiguous specification like:

```
[1 2 3 4 5]
```

to use channels 1, 3, and 5. Connect a Simulink ground to channels 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 is the 96 channel version, the board reads only the first value in the vector and resets 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 (`-1` means sample time is inherited).

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

Topics

"Specifying UEI Boards" on page 49-3

External Websites

www.ueidaq.com

UEI PD2-AO Digital Input

PD2-AO Digital Input block

Library

Simulink Real-Time Library for UEI

Scaling Input to Output

I/O Module 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 arbitrary order. Number the channels beginning with 1 even if this board manufacturer numbers them beginning with 0. The maximum allowable channel number is 16 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 (`-1` means sample time is inherited).

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

Topics

"Specifying UEI Boards" on page 49-3

External Websites

www.ueidaq.com

UEI PD2-AO Digital Output

PD2-AO Digital Output block

Library

Simulink Real-Time Library for UEI

Scaling Input to Output

I/O Module 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 arbitrary order. Number the channels beginning with 1 even if this board manufacturer numbers them beginning with 0. The maximum allowable channel number is 16, 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 (`-1` means sample time is inherited).

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

Topics

"Specifying UEI Boards" on page 49-3

External Websites

www.ueidaq.com

UEI PD2-DIO-64

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 provide:

- 64 digital I/O channels in 16-bit ports
- 3 counters/timers
- 2 enhanced synchronous serial interface (ESSI) ports
- 4 high-speed (100 ns) IRQ lines

The Simulink Real-Time block library supports this series of boards with these driver blocks:

- UEI PD2-DIO-64 Digital Input
- UEI PD2-DIO-64 Digital Output

The Simulink Real-Time 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

See Also

Topics

"Specifying UEI Boards" on page 49-3

External Websites

www.ueidaq.com

UEI PD2-DIO-64 Digital Input

PD2-DIO-64 Digital Input block

Library

Simulink Real-Time Library for UEI

Scaling Input to Output

I/O Module Input	Block Output Data Type	Scaling
TTL	Double	TTL low = 0.0 TTL high = 1.0

Note

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 arbitrary order. Number the channels beginning with 0, which is the same as the numbering on the I/O module label.

Sample time

Enter the base sample time or a multiple of the base sample time (`-1` means sample time is inherited).

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

Topics

"Specifying UEI Boards" on page 49-3

External Websites

www.ueidaq.com

UEI PD2-DIO-64 Digital Output

PD2-DIO-64 Digital Output block

Library

Simulink Real-Time Library for UEI

Scaling Input to Output

I/O Module 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 arbitrary order. Number the channels beginning with `0`, which is the same as the numbering on the I/O module 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 replicated over the channel vector. If you specify a value of `1`,

the corresponding channel is reset to the value specified in the initial value vector. If you specify a value of `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 replicated as the initial value over the channel vector. 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 (`-1` means sample time is inherited).

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

Topics

"Specifying UEI Boards" on page 49-3

External Websites

www.ueidaq.com

UEI PD2-DIO-128

Support for the UEI PD2-DIO-128 Series I/O boards

Board Series

UEI PD2-DIO-128

General Board Series Description

The PD2-DIO-128 series PCI boards provide:

- 128 digital I/O channels in 16-bit ports
- 3 counters/timers
- 2 enhanced synchronous serial interface (ESSI) ports
- 4 high-speed (100 ns) IRQ lines

The Simulink Real-Time block library supports this series of boards with these driver blocks:

- UEI PD2-DIO-128 Digital Input
- UEI PD2-DIO-128 Digital Output

The Simulink Real-Time 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-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

See Also

Topics

"Specifying UEI Boards" on page 49-3

External Websites

www.ueidaq.com

UEI PD2-DIO-128 Digital Input

PD2-DIO-128 Digital Input block

Library

Simulink Real-Time Library for UEI

Scaling Input to Output

I/O Module Input	Block Output Data Type	Scaling
TTL	Double	TTL low = 0.0 TTL high = 1.0

Note

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 arbitrary order. Number the channels beginning with 0, which is the same as the I/O module numbering.

Sample time

Enter the base sample time or a multiple of the base sample time (`-1` means sample time is inherited).

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

Topics

"Specifying UEI Boards" on page 49-3

External Websites

www.ueidaq.com

UEI PD2-DIO-128 Digital Output

PD2-DIO-128 Digital Output block

Library

Simulink Real-Time Library for UEI

Scaling Input to Output

I/O Module 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 arbitrary order. Number the channels beginning with 0, which is the same as the numbering on the I/O module 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 replicated over the channel vector. If you specify a value of 1,

the corresponding channel is reset to the value specified in the initial value vector. If you specify a value of `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 replicated as the initial value over the channel vector. 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 (`-1` means sample time is inherited).

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

Topics

"Specifying UEI Boards" on page 49-3

External Websites

www.ueidaq.com

UEI PD2-DIO-128i

Support for the UEI PD2-DIO-128i Series I/O boards

Board Series

UEI PD2-DIO-128i

General Board Series Description

The PD2-DIO-128i series PCI boards provide as 16-bit ports:

- 64 opto-isolated digital input ports
- 64 opto-isolated digital output ports

The Simulink Real-Time block library supports this series of boards with these driver blocks:

- UEI PD2-DIO-128i Digital Input
- UEI PD2-DIO-128i Digital Output

Board Characteristics

Board type	PD2-DIO-128i
Manufacturer	United Electronic Industries (UEI)
Bus type	PCI
Access method	Memory mapped
Multiple block instance support	No
Multiple board support	Yes

See Also

Topics

“Specifying UEI Boards” on page 49-3

External Websites

www.ueidaq.com

UEI PD2-DIO-128i Digital Input

PD2-DIO-128i Digital Input block

Library

Simulink Real-Time Library for UEI

Scaling Input to Output

I/O Module 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.

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. You can 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 arbitrary order. Number the channels beginning with 0, which is the same as the numbering on the I/O module label.

Sample time

Enter the base sample time or a multiple of the base sample time (`-1` means sample time is inherited).

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

Topics

"Specifying UEI Boards" on page 49-3

External Websites

www.ueidaq.com

UEI PD2-DIO-128i Digital Output

PD2-DIO-128i Digital Output block

Library

Simulink Real-Time Library for UEI

Scaling Input to Output

I/O Module 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.

The input and output ports on the PD2-DIO-128i are completely separate. You can 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 arbitrary order. Number the channels beginning with 0, which is the same as the numbering on the I/O module 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 replicated over the channel vector. If you specify a value of `1`, the corresponding channel is reset to the value specified in the initial value vector. If you specify a value of `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 replicated as the initial value over the channel vector. 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 (`-1` means sample time is inherited).

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

Topics

"Specifying UEI Boards" on page 49-3

External Websites

www.ueidaq.com

UEI PDL-DIO-64

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 provide:

- 64 digital I/O channels in 16-bit ports
- 3 counters/timers
- 2 enhanced synchronous serial interface (ESSI) ports
- 4 high-speed (100 ns) IRQ lines

The Simulink Real-Time block library supports this series of boards with these driver blocks:

- UEI PDL-DIO-64 Digital Input
- UEI PDL-DIO-64 Digital Output

The Simulink Real-Time software only supports the digital I/O channels for this board series. It does not support the additional functionality.

Board Characteristics

Board type	PDL-DIO-64 PDL-DIO-64ST PDL-DIO-64CT PDL-DIO-64TS
Manufacturer	United Electronic Industries (UEI)
Bus type	PCI

Access method	Memory mapped
Multiple block instance support	No
Multiple board support	Yes

See Also

Topics

"Specifying UEI Boards" on page 49-3

External Websites

www.ueidaq.com

UEI PDL-DIO-64 Digital Input

PDL-DIO-64 Digital Input block

Library

Simulink Real-Time Library for UEI

Scaling Input to Output

I/O Module Input	Block Output Data Type	Scaling
TTL	Double	TTL low = 0.0 TTL high = 1.0

Note

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 arbitrary order. Number the channels beginning with 0, which is the same as the numbering on the I/O module label.

Sample time

Enter the base sample time or a multiple of the base sample time (`-1` means sample time is inherited).

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

Topics

"Specifying UEI Boards" on page 49-3

External Websites

www.ueidaq.com

UEI PDL-DIO-64 Digital Output

PDL-DIO-64 Digital Output block

Library

Simulink Real-Time Library for UEI

Scaling Input to Output

I/O Module 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 arbitrary order. Number the channels beginning with 0, which is the same as the numbering on the I/O module 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 replicated over the channel vector. If you specify a value of 1,

the corresponding channel is reset to the value specified in the initial value vector. If you specify a value of `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 replicated as the initial value over the channel vector. 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 (`-1` means sample time is inherited).

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

Topics

"Specifying UEI Boards" on page 49-3

External Websites

www.ueidaq.com

UEI PDXI-DIO-64

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 provide:

- 64 digital I/O channels in 16-bit ports
- 3 counters/timers
- 2 enhanced synchronous serial interface (ESSI) ports
- 4 high-speed (100 ns) IRQ lines

The Simulink Real-Time block library supports this series of boards with these driver blocks:

- UEI PDXI-DIO-64 Digital Input
- UEI PDXI-DIO-64 Digital Output

The Simulink Real-Time software only supports the digital I/O channels for this board series. It does not support the additional functionality.

Board Characteristics

Board type	PDXI-DIO-64 PDXI-DIO-64ST PDXI-DIO-64CT PDXI-DIO-64TS
Manufacturer	United Electronic Industries (UEI)
Bus type	CompactPCI, PXI

Access method	Memory mapped
Multiple block instance support	No
Multiple board support	Yes

See Also

Topics

"Specifying UEI Boards" on page 49-3

External Websites

www.ueidaq.com

UEI PDXI-DIO-64 Digital Input

PDXI-DIO-64 Digital Input block

Library

Simulink Real-Time Library for UEI

Scaling Input to Output

I/O Module Input	Block Output Data Type	Scaling
TTL	Double	TTL low = 0.0 TTL high = 1.0

Note

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 arbitrary order. Number the channels beginning with 0, which is the same as the numbering on the I/O module label.

Sample time

Enter the base sample time or a multiple of the base sample time (`-1` means sample time is inherited).

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

Topics

"Specifying UEI Boards" on page 49-3

External Websites

www.ueidaq.com

UEI PDXI-DIO-64 Digital Output

PDXI-DIO-64 Digital Output block

Library

Simulink Real-Time Library for UEI

Scaling Input to Output

I/O Module 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 arbitrary order. Number the channels beginning with 0, which is the same as the numbering on the I/O module 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 replicated over the channel vector. If you specify a value of 1,

the corresponding channel is reset to the value specified in the initial value vector. If you specify a value of `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 replicated as the initial value over the channel vector. 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 (`-1` means sample time is inherited).

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

Topics

"Specifying UEI Boards" on page 49-3

External Websites

www.ueidaq.com

UEI PDXI-AO Series

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 provide:

- 8, 16, or 32 analog output (D/A) channels (16 bit) with a maximum sampling rate of 100 kHz per channel
- 8 digital input channel
- 8 digital output channel

The Simulink Real-Time 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

Board Characteristics

Board type	PDXI-AO-8/16 PDXI-AO-16/16 PDXI-AO-32/16 PDXI-AO-96/16
Manufacturer	United Electronic Industries (UEI)
Bus type	CompactPCI, PXI
Access method	Memory mapped

Multiple block instance support	No
Multiple board support	Yes

See Also

Topics

"Specifying UEI Boards" on page 49-3

External Websites

www.ueidaq.com

UEI PDXI-AO Analog Output (D/A)

PDXI-AO Analog Output block

Library

Simulink Real-Time Library for UEI

Scaling Input to Output

I/O Module 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 arbitrary order. Number the channels beginning with 1 even if this 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 other board 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 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 (`-1` means sample time is inherited).

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

Topics

"Specifying UEI Boards" on page 49-3

External Websites

www.ueidaq.com

UEI PDXI-AO Digital Input

PDXI-AO Digital Input block

Library

Simulink Real-Time Library for UEI

Scaling Input to Output

I/O Module 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 arbitrary order. Number the channels beginning with 1 even if this board manufacturer numbers them beginning with 0. The maximum allowable channel number is 16 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 (`-1` means sample time is inherited).

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

Topics

"Specifying UEI Boards" on page 49-3

External Websites

www.ueidaq.com

UEI PDXI-AO Digital Output

PDXI-AO Digital Output block

Library

Simulink Real-Time Library for UEI

Scaling Input to Output

I/O Module 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 arbitrary order. Number the channels beginning with 1 even if this board manufacturer numbers them beginning with 0. The maximum allowable channel number is 16, 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 (`-1` means sample time is inherited).

PCI slot (-1:autosearch)

If only one board of this type is in the target computer, enter `-1` to 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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

See Also

Topics

"Specifying UEI Boards" on page 49-3

External Websites

www.ueidaq.com

Asynchronous Events

Asynchronous Event Support

In this section...

Adding an Asynchronous Event

The Simulink Real-Time software includes support for asynchronous events in response to an interrupt from I/O boards. In response to these interrupts, the CPU can suspend normal execution and jump to another section of code called an Interrupt Service routine (ISR).

When developing an Simulink Real-Time model, you can model an Interrupt Server Routine (ISR) by using a Function-Call Subsystem. Also, 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.

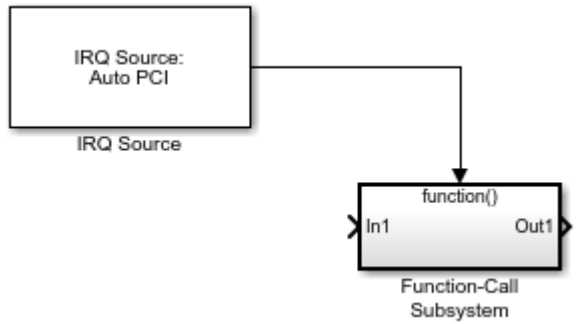
After you install an I/O board with interrupt support into your target computer, you can add Simulink Real-Time asynchronous blocks to your Simulink model.

1. In the MATLAB Command Window, type

   ```
   slrtlib
   ```

   The Simulink Real-Time Library opens.
2. Double-click the Asynchronous Event group block.

   The Library: slrtlib/Asynchronous Event window opens.
3. Drag the Simulink Real-Time IRQ block into 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 To determine and use the IRQ that the BIOS assigned to the board, from the **IRQ line number** list, choose `Auto (PCI only)`.

Alternatively, choose one of the values `3–15` for this number. To determine the available IRQ line numbers on the target computer, use the function `SimulinkRealTime.target.getPCIInfo`.

6 From the **I/O board generating the interrupt** drop-down list, select an interrupt board.

7 In the **PCI slot (-1: autosearch) or ISA base address** field, enter the PCI slot number or enter `-1` to let the Simulink Real-Time software determine the number.

8 Click **OK**.

For more information about the IRQ Source block, see Async IRQ Source.

To transfer data from your ISR, add an Async Transition block or Async Read/Write block to your Simulink model. See Async Rate Transition, Async Buffer Write and Read, and “Asynchronous Interrupt Examples” on page 51-4.

If you are using a CAN field bus with interrupts, see “Asynchronous Interrupt Examples” on page 51-4.

Asynchronous Interrupt Examples

The Simulink Real-Time software provides several example models. If you installed the MATLAB software in the default location, these models are located here:

```
C:\MATLAB\toolbox\rtw\targets\xpc\xpcdemos
```

To access these models, in the MATLAB Command Window, type the name of the model. Each model contains annotations documenting its purpose, and serves as an example of how to use these blocks.

- `xpcasyncbuffer` — 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.

Asynchronous Event: Blocks

This topic describes the Target Management library and Displays and Logging library blocks:

Async Buffer Write and Read

Async Buffer Write and Read blocks

Library

Simulink Real-Time Library for Asynchronous Event

Description

These blocks provide double buffering of data between the ISR and the model which executes rate-monotonically in real time. 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.

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.

Block Parameters

Sample time

Enter the base sample time or a multiple of the base sample time (`-1` means sample time is inherited).

See Also

“Asynchronous Event Support” on page 51-2

Introduced before R2006a

Async IRQ Source

Async IRQ Source block

Library

Simulink Real-Time Library for Asynchronous Event

Description

The IRQ Source block configures the Simulink and Simulink Real-Time software to treat a particular Function-Call Subsystem as an Interrupt Service Routine (ISR). This block is actually a virtual block and does not exist at model execution time. However, the model initialization code sets up the CPU to execute the ISR when the specified interrupt occurs.

Block Parameters

IRQ line number

Select `Auto (PCI only)` to enable the Simulink Real-Time software to automatically determine the IRQ that the BIOS assigned to the board and use it.

Alternatively, select the IRQ line number you are using for this block. This depends on the characteristics of your I/O module. You may need to query the PCI bus in the target computer to find what IRQ the PCI bus assigned to your I/O module. Use the function `SimulinkRealTime.target.getPCIInfo`.

Valid IRQ numbers are between 3 and 15.

I/O board generating the interrupt

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.

PCI slot (-1: autosearch) or ISA base address

- If PCI:

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:

```
tg = slrt;
getPCIInfo(tg, 'installed')
```

- If ISA, enter the base address.

See Also

"Asynchronous Event Support" on page 51-2

Introduced before R2006a

Async Rate Transition

Async Rate Transition block

Library

Simulink Real-Time Library for Asynchronous Event

Description

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.

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.

If a second interrupt occurs while the buffer is being copied, data is corrupted. The CPU copies part of the data from the first buffer. When the second interrupt occurs, it writes over the entire first buffer. When the CPU returns from the second interrupt, it continues the copy operation from the first buffer, which contains data written during the second interrupt.

To prevent data corruption, use `Async Buffer Write and Read` blocks.

Block Parameters

Sample time

Enter the base sample time or a multiple of the base sample time (`-1` means sample time is inherited).

See Also

"Asynchronous Event Support" on page 51-2

Introduced before R2006a

Utility Drivers, Target Management, Displays and Logging

Utility Blocks

The Simulink Real-Time utility blocks support utility functions. Some of these blocks exist in the Utilities library, available at the top level of the Simulink Real-Time 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.

Bit Packing

Construct data frames

Library

Simulink Real-Time Library for Utilities

Description

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 arbitrary 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 dialog box.

Block Parameters

Bit Patterns

Specify bit patterns. The data type entered in the control must be a MATLAB cell array vector. The number of elements in the cell array define the number of input ports shown by this block instance. The cell array elements must be of type double array and define the position of each bit of the incoming value (data typed input port) in the outgoing double value (data frame). From a data type perspective (input ports), the block behaves like a Simulink Sink block, and therefore the data types of the input ports are inherited from the driving blocks.

Output port (packed) data type

From the list, select an output port (packed) data type.

- `double`
- `single`
- `int8`
- `uint8`
- `int16`

- `uint16`
- `int32`
- `uint32`
- `boolean`

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.

Introduced in R2006a

Bit Unpacking

Deconstruct data frames

Library

Simulink Real-Time 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 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`

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 of arbitrary 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.

Introduced in R2006a

Byte Packing

Construct data frames
Library: Utilities

Description

The Byte Packing block converts one or more signals of user-selectable data types to a single vector of varying data types. The output of this block typically connects to an input port of a Send block.

For example, suppose that you are packing three signals into a vector of `uint8`. The signals have the following attributes:

Dimension	Size	Type
Scalar	1	`single`
Vector	3	`uint8`
Vector	3	`uint8`

1. Set the packed output port data type to `uint8`.
2. Set the input port data type to a cell array encoding the data types:

   ```
   {'single', ['uint8'], ['uint8']}
   ```

   Use square brackets to represent vectors.
3. Set the byte alignment value to 1.
4. Connect the signals to the Byte Packing block.

Ports

Input

`Port_1` — First of *N* input ports

scalar | vector

The block has from 1 to *N* input ports. Specify the number of input ports and their types by entering them as a cell array in the parameter **Input port (unpacked) data types (cell array)**.

Data Types: `single` | `double` | `int8` | `int16` | `int32` | `uint8` | `uint16` | `uint32` | `Boolean`

Output

`Port_1` — Output port containing packed data

vector

The block displays one output port that transmits a vector of packed data. You determine the data type of the packed data by setting **Output port (packed) data type**.

Data Types: `single` | `double` | `int8` | `uint8` | `int16` | `uint16` | `int32` | `uint32` | `Boolean`

Parameters

`Output port (packed) data type` — Data type for the packed output signal

`uint8` (default) | `double` | `single` | `int8` | `int16` | `uint16` | `int32` | `uint32` | `boolean`

From the list, select a data type for the output port.

`Input port (unpacked) data types (cell array)` — Data types for the unpacked input signals

`{'uint8'}` (default) | `double` | `single` | `int8` | `int16` | `uint16` | `int32` | `uint32` | `boolean`

Specify as a cell array the data types of the input ports (unpacked) for the different input signals. The number of elements in the cell array determines the number of input ports

shown by this block instance. To represent vector elements, use square brackets in the cell array.

`Byte Alignment` — Alignment of the input signal data types after packing
1 (default) | 2 | 4 | 8

Each element in the input signals list starts at a multiple of the alignment value, specified from the start of the vector. If the alignment value is larger than the size of the data type in bytes, the block fills the space with pad bytes of value `0`.

For example, if the alignment value is `4`:

- `uint32` receives no padding
- `uint16` receives `2` bytes of padding
- `uint8` receives `3` bytes of padding

If the model accesses the data items frequently, consider selecting an alignment value equal to the largest data type that you want to access. If the model transfers data items frequently as a group, consider an alignment value of `1`, which packs the data into as small a space as possible.

Model Examples

See Also

See Also
`Byte Unpacking`

Introduced in R2006a

Byte Reversal/Change Endianess

Reverse little-endian data for big-endian processor

Library

Simulink Real-Time Library for Utilities

Description

You use the Byte Reversal/Change Endianess block for communication between a Simulink Real-Time system and a system running with a processor that is *big-endian*. Processors compatible with the Intel 80x86 family are *little-endian*. For this situation, insert a Byte Reversal/Change Endianess block before the Pack block and another just after the Unpack block. The following is the Change Endianess block.

Block Parameters for Change Endianess

Number of input ports

The number of input ports adjusts automatically to follow this parameter, and the number of outputs is equal to the number of inputs.

Machine word length

Select one of the following machine word lengths to which to convert the data:

- Byte
- Word
- Double Word

The following is the Byte Reversal block.

Byte Reversal Block Parameters

Number of inputs

The number of input ports adjusts automatically to follow this parameter, and the number of outputs is equal to the number of inputs.

Introduced in R2006a

Byte Unpacking

Deconstruct data frames
Library: Utilities

Description

This block converts a vector of varying data types into one or more signals of user-selectable data types. The input of this block typically connects to an output port of a Receive block.

For example, suppose that you are unpacking a `uint8` vector signal into three signals. The signals have the following attributes:

Dimension	Size	Type
Scalar	1	`single`
Vector	3	`uint8`
Vector	3	`uint8`

1. Set the output port data type to:

   ```
   {'single', ['uint8'], ['uint8']}
   ```

   Use square brackets to represent vectors.
2. Set the output port dimension to:

   ```
   {[1],[3],[3]}
   ```
3. Set the alignment value to `1`.
4. Connect the output signals to the Byte Unpacking block.

Ports

Input

`Port_1` — Input port containing packed data

vector

The block displays one input port that receives a vector of packed data. The source of the packed data determines by inheritance the data type of the packed data.

Data Types: `single` | `double` | `int8` | `uint8` | `int16` | `uint16` | `int32` | `uint32` | `Boolean`

Output

`Port_1` — First of *N* output ports

scalar | vector

The block displays from `1` to *N* output ports, as specified by elements of the cell array in the parameter **Output port (unpacked) data types (cell array)**.

Data Types: `single` | `double` | `int8` | `int16` | `int32` | `uint8` | `uint16` | `uint32` | `Boolean`

Parameters

`Output port (unpacked) data types (cell array)` — Data types for the unpacked output signals

`uint8` (default) | `double` | `single` | `int8` | `int16` | `uint16` | `int32` | `uint32` | `boolean`

Specify as a cell array the data types of the output ports (unpacked) for the different output signals. The number of elements in the cell array determines the number of output ports shown by this block instance. To represent vector elements, use square brackets in the cell array.

`Output port (unpacked) dimensions (cell array)` — Dimensions of each output port (unpacked)

`{[1]}` (default) | `{[N], [M], ...}`

Specify the dimensions of the output ports as a cell array of vectors.

`Byte Alignment` — Alignment of the output signal data types before unpacking
1 (default) | 2 | 4 | 8

Each element in the output signals list starts at a multiple of the alignment value, specified from the start of the input vector. If the alignment value is larger than the size of the data type in bytes, the vector contains pad bytes of value `0`.

For example, if the alignment value is `4`:

- `uint32` receives no padding
- `uint16` receives `2` bytes of padding
- `uint8` receives `3` bytes of padding

Model Examples

See Also

See Also

`Byte Packing`

Introduced in R2006a

Extended Counter

Extended Counter sample block

Library

Simulink Real-Time Library for Rollover

Description

This block outputs a count not limited to the number of bits on the board encoder counter register. It receives as input the output of an encoder block.

Block Parameters

Rollover Counts

Enter the number of rollover counts. This is the number at which the rollover occurs. This value should normally be `2^n`, where `n` is the number of bits in the encoder counter register. Each rollover equals this number.

Encoder Starting Value

Enter the starting value with which the encoder block begins. This value should match the value in the encoder mask.

Rollover Threshold

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

- Larger than the worst-case difference between successive outputs (to prevent declaring a rollover when it was just a large step)
- Less than the maximum count value minus the worst-case difference (to prevent missing a rollover)

  Ideally, enter the value `2^(n-1)`, where `n` is the number of bits in the encoder counter register. The default is `2^15`.

See Also

Rollover Counter

Rollover Counter

Rollover Counter sample block

Library

Simulink Real-Time Library for Rollover

Description

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.

Block Parameters

Encoder Starting Value

Enter the starting value you want the encoder block to begin with. Match this value to the value in the encoder mask.

Rollover Threshold

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.

See Also

Extended Counter

Shared Memory Pack

Shared memory pack

Library

Simulink Real-Time Library for Shared Memory

Description

This block packs the specified partition structure into an unstructured double word array vector. It converts 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; pass the output of each structure segment as input to the Shared Memory Pack block.

Memory partitions consist of groups of Simulink signals, which are combined into blocks (packets) of 32-bit words. Before you begin to configure this block, be sure that you have a predefined shared memory partition structure as required by the shared memory manufacturer.

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

Introduced in R2006a

Shared Memory Unpack

Shared memory unpacking

Library

Simulink Real-Time Library for Shared Memory

Description

This block unpacks an unstructured double word array vector (from the Shared Memory Pack block) into the specified partition structure.

Before you begin to configure this block, be sure that you have a predefined shared memory partition structure as required by the shared memory manufacturer.

This block ignores the **Address** field of the partition structure.

Block Parameters

Partition struct

Enter the name of the predefined shared memory partition structure. The block unpacks the double word array vector into this structure.

See Also

Shared Memory Pack

Introduced in R2006a

54

Target Management, Display, and Logging Blocks

CPU Temperature

Return current CPU temperature in Celsius
Library: Target Management / Target Information

Description

This block outputs the CPU temperature. You can monitor this value and halt real-time execution when the temperature reaches a value that depends on the temperature range of the target computer.

Ports

Output

`Port_1` — CPU temperature
scalar

Outputs the CPU temperature in Celsius with granularity of 1 °C.

Data Types: `double`

Parameters

`Sample time (-1 for inherited)` — Sample time of block
`-1` (default) | numeric

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

Model Examples

See Also

Introduced in R2017a

From File

Read data from file on target computer

Library

Simulink Real-Time Library for Target Management

Description

The From File block reads data from a file on the target computer hard disk and outputs that data in chunks every sample time. As the Simulink Real-Time 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 and propagates it to the block outputs for use by the real-time application. For example, use the From File block to drive a model with externally acquired data (data from a file).

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 that 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:

```
28 bytes (1 * sizeof('single') + 3 * sizeof('double'))
```

This data can then be converted into signal values by the Byte Unpacking block.

See the following topics:

- “File Format” on page 54-5 — Describes the target computer source file format
- “Block Parameters” on page 54-6 — Describes the block parameters for the From File block

File Format

Before you use a target computer file as the source for the From File block, 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.

For example, assume that your file contains the data from 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 `SimulinkRealTime.utils.bytes2file` function to create the file on the development computer. This function has the following syntax:

```
SimulinkRealTime.utils.bytes2file(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.

You can then use `SimulinkRealTime.copyFileToTarget` to download the file to the target computer.

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 Simulink Real-Time 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 time the disk read size

  Increasing this parameter value helps prevent the real-time application from emptying the buffer faster than the background task can fill it. This can happen if you have multitasking models or conditionally executed subsystems, which can cause temporary increases in task execution time and leave less time for the background task to fill the buffer.

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 buffer decreases by eight. Each time the number of free bytes in the buffer goes to 512 or higher, the Simulink Real-Time kernel attempts to read 512 bytes from the Simulink Real-Time 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 real-time 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 the base sample time or a multiple of the base sample time (`-1` means sample time is inherited).

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.

Introduced before R2006a

From Target

Read data from target computer

Library

Simulink Real-Time Library for Displays and Logging

Description

This block behaves like a source. Its output is connected to the input of a display device.

You can use the function `SimulinkRealTime.utils.createInstrumentationModel` to create an instrumentation model using the From Target and To Target blocks.

The From Target block runs as a non-real-time Simulink block on the development computer. It is asynchronous to the real-time application running on the target computer. If the real-time application has a sample time slower than the non-real-time model, the From Target block can query the target computer more than once. The target computer returns the same value in this case. Conversely, it is possible for the From Target block to miss some sample times between two successively returned values.

Note: The use of From Target blocks requires a connection between the development and target computers. Without a connection, operations such as opening a model or copying these blocks take longer than normal.

Some notes on the From Target block behavior:

- To highlight a signal line that a From Target block refers to, double-click the From Target block.
- If the From Target block has not yet been configured, double-clicking the From Target block has no effect.
- To edit the From Target block parameters, right-click the block and select **Mask Parameters**.

Block Parameters

Target application name

The function `SimulinkRealTime.utils.createInstrumentationModel` automatically enters a name entry for this parameter. It is the same name as the Simulink model that the Simulink Real-Time software uses to build the real-time application.

Signal name (block name)

The function `SimulinkRealTime.utils.createInstrumentationModel` automatically enters a name entry for this parameter. For multiple blocks, the function creates a From 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 `SimulinkRealTime.utils.createInstrumentationModel` 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 real-time application to the default target computer. This assumes that you configured a default target computer through the Simulink Real-Time Explorer (see "PCI Bus Ethernet Setup" or "USB-to-Ethernet 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"

Introduced in R2014a

Time Stamp Counter

Pentium timestamp counter

Library

Simulink Real-Time Library for Target Information

Description

This block outputs a `uint32` vector of length 2 that contains the timestamp 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 can result in abnormal behavior, including a halt in execution. This block takes an arbitrary input used only to sequence execution of the block.

Introduced in R2014a

Time Stamp Delta

Timestamp delta

Library

Simulink Real-Time Library for Target Information

Description

This block takes as input two `uint32` vectors of length `2`. Two separate Time Stamp Counter blocks can supply each vector. The output is a scalar double that contains the delta between the two timestamps.

Introduced in R2006a

Scope

Real-time Scope block

Library

Simulink Real-Time Library for Displays and Logging

Description

Real-time 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 54-12
- "Host Scope Block Parameters" on page 54-16
- "File Scope Block Parameters" on page 54-18

Note:

- To monitor the output signal from a Constant block using a real-time Scope block, add a test point for the Constant block output signal.
- If your model connects the output of a Mux block to the input of a real-time Scope block, the signal is not observable. To observe the signal, add a unity gain block (a Gain block with a gain of 1) between the Mux block and the Scope block.
- The real-time application can generate data faster than the kernel can process it. Previous data can be overwritten, causing gaps. If gaps occur in the data, consider increasing the value of the **Decimation** property of the scope.

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 Simulink Real-Time Scope block. Typically, you do not need to edit this value.

This number identifies the Simulink Real-Time Scope block and the scope display on the development 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 you download and start the real-time application. 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 scrolling (refreshing the entire plot).
- `Graphical rolling` — Displays running data continuously scrolling from left to right across the scope (similar behavior to oscilloscopes).
- `Graphical sliding` — The value `'sliding'` will be removed in a future release. It behaves like value `rolling`.

If you have a scope type of `Target` and a scope mode of `Numerical`, the scope block dialog adds a **Numerical format** box to the dialog. You can define the display format for the data. If you choose not to complete the **Numerical format** box, the Simulink Real-Time software displays the signal using the default format of `%15.6f`. This is a floating point format without a label.

Numerical format

If you have a scope type of `Target` and a scope mode of `Numerical`, the scope block dialog adds a **Numerical format** box to the dialog. Enter a label and associated numeric format type in which to display signals. By default, the entry format is the floating point `%15.6f`. The **Numerical format** box takes entries of the format:

```
'[LabelN] [%width.precisiontype] [LabelX]'
```

where

`LabelN` is the label for the signal. You can use a different label for each signal or the same label for each signal. This argument is optional.

`width` is the minimum number of characters to offset from the left of the screen or label. This argument is optional.

`precision` is the maximum number of decimal points for the signal value. This argument is optional.

`type` is the data type for the signal format. You can use one or more of the following types:

Type	Description
`%e` or `%E`	Exponential format using `e` or `E`
`%f`	Floating point
`%g`	Signed value printed in `f` or `e` format depending on which is smaller
`%G`	Signed value printed in `f` or `E` format depending on which is smaller

`LabelX` is a second label for the signal. You can use a different label for each signal or the same label for each signal. This argument is optional.

Enclose the contents of the **Numerical format** box 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 character vector 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 Simulink Real-Time 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` 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` and `Graphical rolling`.

Number of samples

Enter the number of values to be acquired in a data package. The minimum number is `3` samples.

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 **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 one of `FreeRun`, `Software triggering`, `Signal triggering`, or `Scope triggering`.

- `FreeRun` or `Software triggering` — You do not need to enter additional parameters.

- `Signal triggering` — Enter the following additional parameters, as required:
  - In the **Trigger signal** box, enter the index of a signal previously added to the scope.

    This parameter does not apply if the **Add signal port to connect a signal trigger source** check box is selected.
  - (Alternatively) Click the **Add signal port to connect a signal trigger source** check box, then connect an arbitrary trigger signal to the port Trigger signal.
  - In the **Trigger level** box, enter a value for the signal to cross before triggering.
  - From the **Trigger slope** list, select one of `Either`, `Rising`, or `Falling`.
- `Scope triggering` — Enter the following additional parameters, as required:
  - In the **Trigger scope number** box, enter the scope number of a Scope block. If you use this trigger mode, you must also add a second Scope block to your Simulink model.
  - If you want the scope to trigger on a specific sample of the other scope, enter a value in the text box **Sample to trigger on (-1 for end of acquisition)**. The default value is `0`, and indicates that the triggering scope and the triggered (current) scope start simultaneously.

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 Simulink Real-Time Scope block. Typically, you do not need to edit this value.

This number identifies the Simulink Real-Time Scope block and the scope display on the development 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 you download and start the real-time application. With a target scope, the scope window opens automatically. With a host scope, you can open a host scope viewer window from Simulink Real-Time Explorer.

Number of samples

Enter the number of values to be acquired in a data package.

Number of pre/post samples

Enter the number of samples to save or skip. Specify a value less than `0` to save this number of samples before a trigger event. Specify a value greater than `0` to skip this number of samples after the trigger event before data acquisition begins.

Decimation

Enter a value to collect data at each sample time (`1`) or to collect data at less than every sample time (`2` or greater).

Trigger mode

From the list, select one of `FreeRun`, `Software triggering`, `Signal triggering`, or `Scope triggering`.

- `FreeRun` or `Software triggering` — You do not need to enter additional parameters.
- `Signal triggering` — Enter the following additional parameters, as required:
  - In the **Trigger signal** box, enter the index of a signal previously added to the scope.

    This parameter does not apply if the **Add signal port to connect a signal trigger source** check box is selected.
  - (Alternatively) Click the **Add signal port to connect a signal trigger source** check box, then connect an arbitrary trigger signal to the port Trigger signal.
  - In the **Trigger level** box, enter a value for the signal to cross before triggering.
  - From the **Trigger slope** list, select one of `Either`, `Rising`, or `Falling`.
- `Scope triggering` — Enter the following additional parameters, as required:
  - In the **Trigger scope number** box, enter the scope number of a Scope block. If you use this trigger mode, you must also add a second Scope block to your Simulink model.

- If you want the scope to trigger on a specific sample of the other scope, enter a value in the text box **Sample to trigger on (-1 for end of acquisition)**. The default value is `0`, and indicates that the triggering scope and the triggered (current) scope start simultaneously.

File 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 Simulink Real-Time Scope block. Typically, you do not need to edit this value.

This number identifies the Simulink Real-Time Scope block and the scope display on the development 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 you download and start the real-time application. The scope window opens automatically.

Number of samples

Enter the number of values to be acquired in a data package.

The **Number of samples** parameter works with the **AutoRestart** setting.

- Autorestart is on — When the scope triggers, the scope starts collecting data into a memory buffer. A background task examines the buffer and writes data to the disk continuously, appending new data to the end of the file. When the scope reaches the number of samples that you specified, it starts collecting data again, overwriting the memory buffer. If the background task cannot keep pace with data collection, data can be lost.
- Autorestart is off — When the scope triggers, the scope starts collecting data into a memory buffer. It stops when it has collected the number of samples that you specified. A background task examines the buffer and writes data to the disk continuously, appending the new data to the end of the file.

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 one of `FreeRun`, `Software triggering`, `Signal triggering`, or `Scope triggering`.

- `FreeRun` or `Software triggering` — You do not need to enter additional parameters.
- `Signal triggering` — Enter the following additional parameters, as required:
  - In the **Trigger signal** box, enter the index of a signal previously added to the scope.

    This parameter does not apply if the **Add signal port to connect a signal trigger source** check box is selected.
  - (Alternatively) Click the **Add signal port to connect a signal trigger source** check box, then connect an arbitrary trigger signal to the port Trigger signal.
  - In the **Trigger level** box, enter a value for the signal to cross before triggering.
  - From the **Trigger slope** list, select one of `Either`, `Rising`, or `Falling`.
- `Scope triggering` — Enter the following additional parameters, as required:
  - In the **Trigger scope number** box, enter the scope number of a Scope block. If you use this trigger mode, you must also add a second Scope block to your Simulink model.
  - If you want the scope to trigger on a specific sample of the other scope, enter a value in the text box **Sample to trigger on (-1 for end of acquisition)**. The default value is `0`, and indicates that the triggering scope and the triggered (current) scope start simultaneously.

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.

A fully qualified file name can have a maximum of 260 characters: The file part can have at most 12 characters: eight for the file name, one for the period, and at most three for the file extension. A file name longer than eight characters is truncated to six characters followed by `'~1'`.

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 maintains 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).

You cannot read a file that was written during real-time execution until execution has completed.

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 better performance.

AutoRestart

Select this check box to have the file scope collect data into a memory buffer. A background task examines the buffer and writes data to the disk continuously,

appending new data to the end of the file. When the scope has collected the required number of samples, it starts collecting data over again, overwriting the memory buffer. If the background task cannot keep up with data collection, data can be lost.

Clear the **AutoRestart** check box to have the scope collect data into a memory buffer up to the specified number of samples and stop. As when autorestart is on, a background task examines the buffer and writes data to the disk continuously, appending the new data to the end of the file.

Setting this check box enables the following parameters: **Dynamic file name enabled** and **Max file size in bytes (multiple of WriteSize)**.

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 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 additional log files, it overwrites the first log file.

Introduced in R2014a

Software Reboot

Restart target computer software

Library

Simulink Real-Time Library for Target Information

Description

You can use the Software Reboot block to restart the target computer when your simulation reaches a certain state. For example, you can restart the target computer when your control system becomes unstable.

This block has one input. The input is the restart signal and accepts the following values:

- `1` — Restarts the target computer.
- `0` — If you use a Software Reboot block and you do not want the target computer to restart, the input value to this block must be `0`.

Note: Some systems do not support restarting from software. To test whether your system supports a software restart, from the MATLAB Command Window, enter `slrttest`. For more information, see "Run Confidence Test on Configuration".

Block Parameters

Sample time

Enter the base sample time or a multiple of the base sample time (`-1` means sample time is inherited).

Introduced before R2006a

To Target

Send data to target computer

Library

Simulink Real-Time Library for Displays and Logging

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 real-time application.

You can use the function `SimulinkRealTime.utils.createInstrumentationModel` to create an instrumentation model using the To Target and From Target blocks.

The To Target block runs as a non-real-time Simulink block on the development computer. It is asynchronous to the real-time model running on the target computer. If the To Target block receives continuously changing input, two parameter updates can be sent to the target computer before the next sample time of the real-time application. In this case, the real-time application only uses the last parameter value received. Conversely, it is also possible for one or more sample times to elapse on the target computer before the To Target block sends the next parameter value.

In either case, the To Target block only sends parameter values to the target computer when there is a change (for example, the input of the To Target block changes).

Note: The use of To Target blocks requires a connection between the development and target computers. Without a connection, operations such as opening a model or copying these blocks take longer than normal.

Some notes on the To Target block behavior:

- To highlight the Simulink model block referenced by a To Target block, double-click the block.
- If the To Target block has not yet been configured, double-clicking the block has no effect.

- To edit the To Target block parameters, right-click the block and select **Mask Parameters**.

Block Parameters

Target application name

The function `SimulinkRealTime.utils.createInstrumentationModel` automatically enters a name entry for this parameter. It is the same name as the Simulink model that Simulink Real-Time uses to build the real-time application.

Path to block in target application

The function `SimulinkRealTime.utils.createInstrumentationModel` automatically enters an entry for this parameter and uses it to access the block identifier.

Parameter name

The function `SimulinkRealTime.utils.createInstrumentationModel` 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 real-time application to the default target computer. This assumes that you configured a default target computer through the Simulink Real-Time Explorer (see "PCI Bus Ethernet Setup" or "USB-to-Ethernet 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"

Introduced in R2014a

Current Available Stack Size

Current Available Stack Size returns free stack available

Library

Simulink Real-Time Library for Execution Parameters

Description

This block outputs the number of bytes of stack memory currently available to the real-time application thread.

See Also

`Minimum Available Stack Size`

Introduced in R2014a

Minimum Available Stack Size

Get the smallest amount of free stack available

Library

Simulink Real-Time Library for Execution Parameters

Description

This block outputs the number of bytes that have not been used in the stack since the thread was created.

Note: The block traverses the entire stack to find unused bytes. Use this block only for diagnostic purposes.

See Also

`Current Available Stack Size`

Introduced in R2014a

Get Overload Counter

Get Overload Counter returns number of CPU overloads

Library

Simulink Real-Time Library for Execution Parameters

Description

This block returns the CPU overload count. To display the value, connect the block output to a real-time `Scope` block. To achieve your required refresh rate, adjust the Simulink Real-Time Scope block parameter **Number of samples** to a small number, such as `10`.

For multirate models in multitasking mode, Get Overload Counter returns the number of overloads of the base rate task.

Block Parameters

Sample time

Enter the base sample time or a multiple of the base sample time (`-1` means sample time is inherited).

See Also

`Set Overload Counter`

Introduced in R2014a

Set Overload Counter

Set current CPU overload count

Library

Simulink Real-Time Library for Execution Parameters

Description

This block enables you to adjust the CPU overload count.

Block Parameters

Sample time

Enter the base sample time or a multiple of the base sample time (`-1` means sample time is inherited).

See Also

`Get Overload Counter`

Introduced in R2014a

Task Execution Time

Task execution time (TET), in seconds

Library

Simulink Real-Time Library for Execution Parameters

Description

This block outputs the task execution time (TET) in seconds.

To visualize the TET while your real-time application is running, connect the output of this block to a Simulink Real-Time Scope block.

Task execution time (TET) measures how long it takes the kernel to run for one base-rate time step. For a multirate model, use the profiler to find out what the execution time is for each rate.

Block Parameters

Sample time

Enter the base sample time or a multiple of the base sample time (`-1` means sample time is inherited).

Introduced in R2014a

Session Time

Time kernel has been executing

Library

Simulink Real-Time Library for Target Information

Description

This block outputs the time in clock ticks that the kernel has been executing. This value is not the same as the execution time of the real-time application. It is the time since you last started the target computer.

A use for this block is to determine the execution time of subsystems in your Simulink model. Add a Session Time block before and after the subsystem, and then calculate the difference in time.

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 1.193×10^6.

Block Parameters

Sample time

Enter the base sample time or a multiple of the base sample time (`-1` means sample time is inherited).

Introduced in R2014a

Drivers No Longer Recommended for Use

Simulink Real-Time Drivers No Longer Recommended for Use

The following sets of drivers are no longer recommended for use with the Simulink Real-Time product.

Library of Drivers No Longer Recommended for Use

Type `xpcobsoletelib` to access driver blocks that are no longer recommended or that have been updated.

Drivers No Longer Recommended

These boards and blocks will be removed in a future release. Do not use them in new models. Instead, use I/O modules and driver blocks available from www.speedgoat.com.

“Utility Blocks No Longer Recommended for Use” on page 55-3

Utility Blocks No Longer Recommended for Use

These utility blocks were used for digital I/O. They are no longer recommended for use.

Digital I/O Bit-Packing	Construct data frames (not recommended)
Digital I/O Bit-Unpacking	Extract data frames (not recommended)

Digital I/O Bit-Packing

Construct data frames (not recommended)

Library

Simulink Real-Time Library of Drivers No Longer Recommended for Use.

To open this library, type `xpcobsoletelib` in the Command Window.

Description

This block constructs data frames. Its output port is typically connected to an input port of a digital I/O block. The block has one output port of data type `uint32` (a scalar). This port 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 dialog box. See `CAN Bit-Packing` for an example of how to use a bit-packing 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 uint32 value (data frame).

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.

This block inherits the sample time of the driving blocks.

See Also

Digital I/O Bit-Unpacking

Digital I/O Bit-Unpacking

Extract data frames (not recommended)

Library

Simulink Real-Time Library of Drivers No Longer Recommended for Use.

To open this library, type `xpcobsoletelib` in the Command Window.

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). This port 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 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