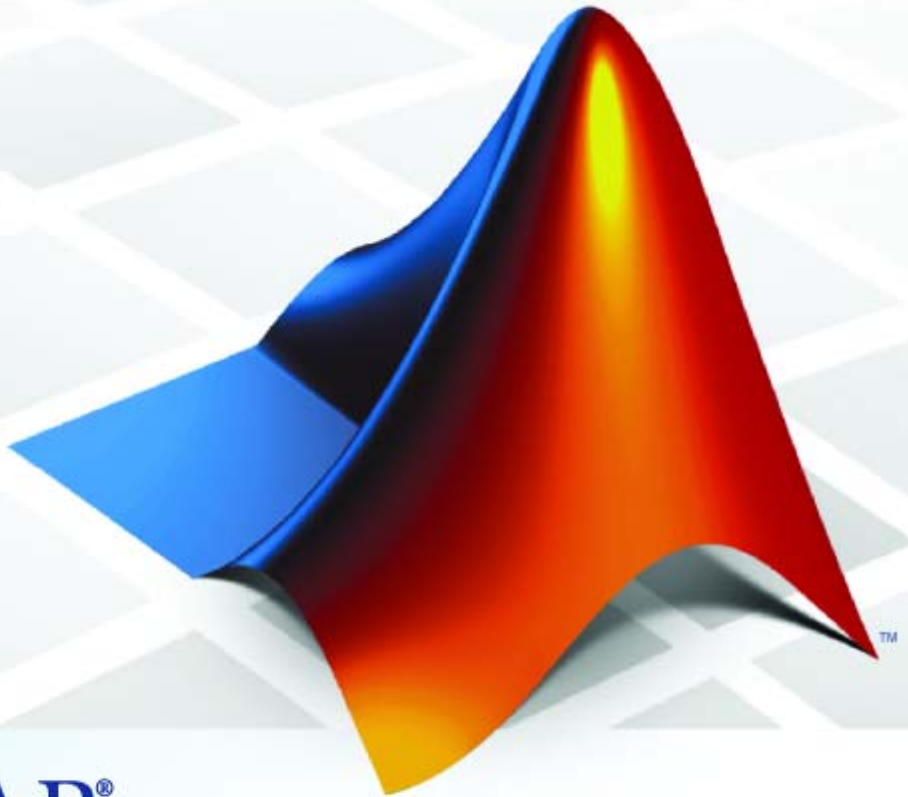


Real-Time Windows Target™ 3

Reference



MATLAB®
& **SIMULINK®**

How to Contact The MathWorks



www.mathworks.com Web
comp.soft-sys.matlab Newsgroup
www.mathworks.com/contact_TS.html Technical Support



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



508-647-7000 (Phone)



508-647-7001 (Fax)



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

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

Real-Time Windows Target™ Reference

© COPYRIGHT 1999–2010 by The MathWorks, Inc.

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

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

Trademarks

MATLAB and Simulink are registered trademarks of The MathWorks, Inc. See www.mathworks.com/trademarks for a list of additional trademarks. Other product or brand names may be trademarks or registered trademarks of their respective holders.

Patents

The MathWorks products are protected by one or more U.S. patents. Please see www.mathworks.com/patents for more information.

Revision History

March 2008	Online only	New for Version 3.1 (Release 2008a)
October 2008	Online only	Revised for Version 3.2 (Release 2008b)
March 2009	Online only	Revised for Version 3.3 (Release 2009a)
September 2009	Online only	Revised for Version 3.4 (Release 2009b)
March 2010	Online only	Revised for Version 3.5 (Release 2010a)

Function Reference

1

Block Reference

2

Input	2-2
Output	2-3

Blocks — Alphabetical List

3

Configuration Parameters

4

Real-Time Workshop Pane: Real-Time Windows	
Target	4-2
Real-Time Windows Target Tab Overview	4-3
Target function library	4-4
Utility function generation	4-6
Compiler optimizations	4-7
Generate assembly listings	4-8
Rebuild all	4-9
External mode	4-10
Transport layer	4-12
MEX-file arguments	4-14
Static memory allocation	4-16
Static memory buffer size	4-18

Function Reference

rtwho

Purpose Display information about Real-Time Windows Target status

Syntax `rtwho`

Description `rtwho` display information about the Real-Time Windows Target™ PC status. For example, it lists:

- All hardware drivers
- Timers
- Kernel performance
- Machine type
- Real-Time Windows Target version

Example This command displays PC information.

```
rtwho
```

```
Real-Time Windows Target version 3.2.0 (C) The MathWorks, Inc. 1994-2008  
Running on Multiprocessor APIC computer.  
MATLAB performance = 100.0%  
Kernel timeslice period = 1 ms
```

Purpose	Attach and activate default Real-Time Windows Target configuration set
Syntax	<pre>rtwinconfigset('model') rtwinconfigset('model', 'ERT') configset=rtwinconfigset() configset=rtwinconfigset('','ERT')</pre>
Arguments	<p><i>model</i> The model to which to attach the default Real-Time Windows Target configuration set.</p> <p>'ERT' The default Real-Time Windows Target configuration set for Real-Time Workshop® Embedded Coder™ software to be attached to <i>model</i>.</p>
Returns	<p><i>configset</i> The default Real-Time Windows Target configuration set, which later can be attached to a model.</p>
Description	<p>The <code>rtwinconfigset</code> function, when called with argument <i>model</i>, attaches the default Real-Time Windows Target configuration set RTWin to the model, then activates the configuration set. This default configuration set specifies various simulation and code-generation parameter values that are useful when working with a Real-Time Windows Target model. In most cases, using <code>rtwinconfigset</code> provides all the configuration parameter values that the model needs.</p> <p>When called with no argument, <code>rtwinconfigset</code> returns the default Real-Time Windows Target configuration set object. Later, you can attach this configuration set to a model. See “Configuring a Model” in the <i>Real-Time Windows Target User’s Guide</i> for more information.</p> <p>The <code>rtwinconfigset('model', 'ERT')</code> function attaches the default Real-Time Windows Target configuration set for Real-Time Workshop Embedded Coder software to the model, then activates the configuration set. This default configuration set specifies various simulation and</p>

rtwinconfigset

code-generation parameter values that are useful when working with a Real-Time Windows Target model in the Real-Time Workshop Embedded Coder environment.

The `rtwinconfigset(' ', 'ERT')` function returns the default Real-Time Windows Target configuration set object for the Real-Time Workshop Embedded Coder software. Later, you can attach this configuration set to a model.

See Also

Configuration Sets
Referencing Configuration Sets
Specifying the Default Configuration Set

Purpose Install and remove Real-Time Windows Target kernel

Syntax

```
rtwintgt -setup  
rtwintgt -install  
rtwintgt -uninstall  
rtwintgt -forceuninstall  
rtwintgt -version
```

Description

`rtwintgt -setup` installs the Real-Time Windows Target kernel on your system. It performs the same operation as `rtwintgt -install`.

`rtwintgt -install` installs the Real-Time Windows Target kernel on your system. It performs the same operation as `rtwintgt -setup`.

`rtwintgt -uninstall` removes the Real-Time Windows Target kernel from your system.

`rtwintgt -forceuninstall` forcibly removes the Real-Time Windows Target kernel from your system. Use this command if `rtwintgt -uninstall` cannot successfully remove the kernel. Use this command only when all other attempts to uninstall the kernel fail. The command leaves the computer in an inconsistent state that cannot be relied on and does not post relevant error messages.

Note Never execute `rtwintgt -forceuninstall` without immediately rebooting, after which you can reinstall the Real-Time Windows Target kernel as described in “Installing the Kernel” in the *Real-Time Windows Target User’s Guide*.

`rtwintgt -version` displays the Real-Time Windows Target currently installed on your system.

Block Reference

Input (p. 2-2)

Input data to a Real-Time Windows
Target application

Output (p. 2-3)

Output data from a Real-Time
Windows Target application

Input

Analog Input	Select and connect analog input channels
Counter Input	Select and connect counter input channels
Digital Input	Select and connect digital input lines or channels
Encoder Input	Select and connect encoder input channels
Other Input	Connect with input sources that other input blocks cannot
Packet Input	Receive unformatted binary data
Stream Input	Receive formatted ASCII data

Output

Analog Output	Select and connect analog output channels
Digital Output	Select and connect digital output lines or channels
Frequency Output	Generate and output a pulse-width-modulated square wave
Other Output	Connect with output sources that other output blocks cannot
Packet Output	Transmit unformatted binary data
Stream Output	Transmit formatted ASCII data

Blocks — Alphabetical List

Analog Input

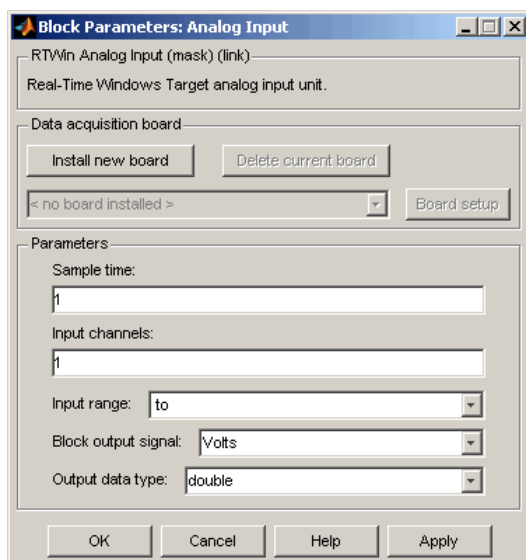
Purpose Select and connect analog input channels

Library Real-Time Windows Target

Description The Analog Input block allows you to select and connect specific analog input channels to your Simulink® model. After you add an Analog Input block to your model, you can enter the parameters for its I/O driver. The following procedure uses the Humusoft® AD512 I/O board as an example:

- 1 Double-click the Analog Input block.

The Block Parameters: Analog Input dialog box opens:



- 2 In the **Sample time** box, enter the same value you entered in the **Fixed step size** box from the Configuration Parameters dialog box, or an integer multiple of that value.

- 3** In the **Input channels** box, enter a channel vector that selects the analog input channels you are using on this board. The vector can be any valid MATLAB® vector form. For example, to select all eight analog input channels on the AD512 board, enter

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

If you want to use the first three analog input channels, enter

[1,2,3]

- 4** From the **Input range** list, choose the input range for all of the analog input channels you entered in the **Input channels** box. For example, with the AD512 board, choose **-5 to 5 V**.

If you want the input range to be different for different analog channels, you need to add an I/O block for each different input range.

- 5** From the **Block output signal** list, choose from the following options:

- **Volts** — Returns a value equal to the analog voltage.
- **Normalized bipolar** — Returns a full range value of -1 to +1 regardless of the input voltage range.
- **Normalized unipolar** — Returns a full range value of 0 to +1 regardless of the input voltage range. For example, an analog input range of 0 to +5 volts and -5 to +5 volts would both be converted to 0 to +1.
- **Raw** — Returns a value of 0 to $2^n - 1$. For example, a 12-bit A/D converter would return values of 0 to $2^{12} - 1$ (0 to 4095). The advantage of this method is the returned value is always an integer with no roundoff errors.

- 6** Set **Output data type** to specify the type of data that the block will output to the model.

- 7** Click **OK** or **Apply**.

Analog Output

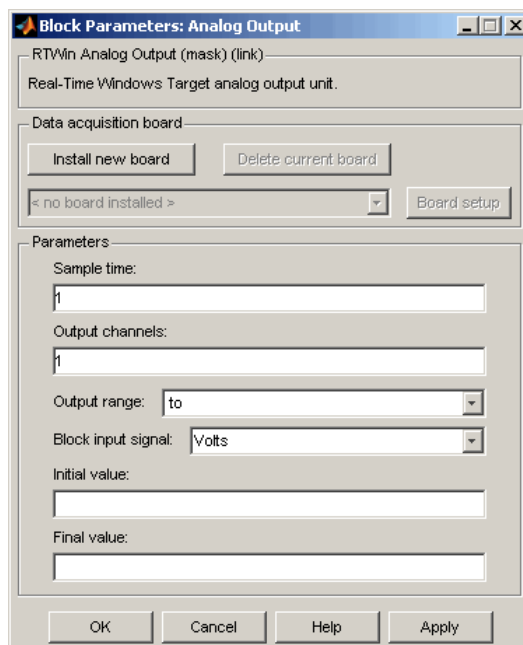
Purpose Select and connect analog output channels

Library Real-Time Windows Target

Description The Analog Output block allows you to select and connect specific analog output channels to your Simulink model. After you add an Analog Output block to your model, you can enter the parameters for its I/O driver. The block outputs The following procedure uses the Humusoft AD512 I/O board as an example.

- 1 Double-click the Analog Output block.

The Block Parameters: Analog Output dialog box opens:



- 2** In the **Sample time** box, enter the same value you entered in the **Fixed step size** box from the Configuration Parameters dialog box, or an integer multiple of that value.

0.001

- 3** In the **Output channels** box, enter a channel vector that selects the analog output channels you are using on this board. The vector can be any valid MATLAB vector form. For example, to select both analog output channels on the AD512 board, enter

[1,2] or [1:2]

- 4** From the **Output range** list, choose the input range for all of the analog input channels you entered in the **Input channels** box. For example, with the AD512 board, choose -5 to 5 V.

If you want the input range to be different for different analog channels, you need to add an I/O block for each different input range.

- 5** From the **Block input signal** list, choose from the following options:

- **Volts** — Expects a value equal to the analog output voltage.
- **Normalized bipolar** — Expects a value between -1 and +1 that is converted to the full range of the output voltage regardless of the output voltage range.
- **Normalized unipolar** — Expects a value between 0 and +1 that is converted to the full range of the output voltage regardless of the output voltage range. For example, an analog output range of 0 to +5 volts and -5 to +5 volts would both be converted from values between 0 and +1.
- **Raw** — Expects a value of 0 to $2^n - 1$. For example, a 12-bit A/D converter would expect a value between 0 and $2^{12} - 1$ (0 to 4095). The advantage of this method is the expected value is always an integer with no roundoff errors.

Analog Output

- 6** Enter the initial value for each analog output channel you entered in the **Output channels** box. For example, if you entered [1,2] in the **Output channels** box, and you want an initial value of 0 volts, enter [0,0].
- 7** Enter a final value for each analog channel you entered in the **Output channels** box. For example, if you entered [1,2] in the **Output channels** box, and you want final values of 0 volts, enter [0,0].
- 8** Click **OK** or **Apply**.

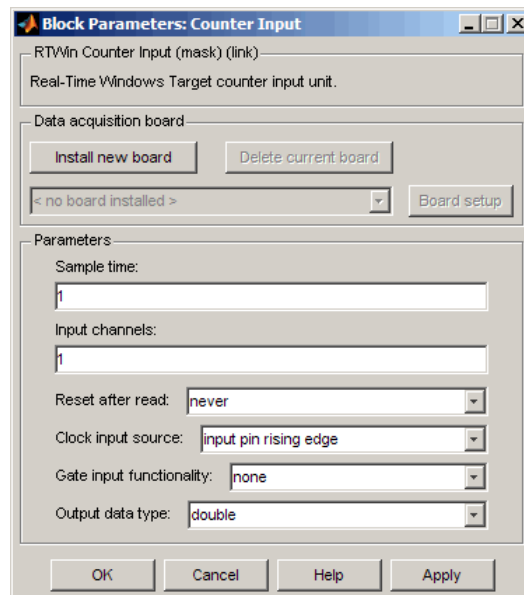
Purpose Select and connect counter input channels

Library Real-Time Windows Target

Description The Counter Input block allows you to select and connect specific counter input channels to your Simulink model. After you have added a Counter Input block to your model, you can enter the parameters for its I/O driver. The following procedure uses the Humusoft MF604 I/O board as an example.

- 1 Double-click the Counter Input block.

The Block Parameters: Counter Input dialog box opens:



- 2 In the **Sample time** box, enter the same value you entered in the **Fixed step size** box from the Configuration Parameters dialog box, or an integer multiple of that value.

Counter Input

3 In the **Input channels** box, enter a channel vector that selects the counter input channels you are using on this board. The vector can be any valid MATLAB vector form. For example, to select all four counter input channels on the MF604 board, enter

[1,2,3,4] or [1:4]

4 From **Reset after read**, which determines if the counter should be reset to zero after its value has been read, choose one of the following options:

- **never** — Do not reset after reading.
- **always** — Always reset after reading.
- **level** — Reset after reading if block input is nonzero. This will add an input to the Counter Input block.
- **rising edge** — Reset after reading if block input changes from zero to nonzero between the last two successive readings. This will add an input to the Counter Input block.
- **falling edge** — Reset after reading if the block input changes from nonzero to zero between last two successive readings. This will add an input to the Counter Input block.
- **either edge** — Reset after reading if the block input changes either from zero to nonzero or from nonzero to zero between the last two successive readings. This will add an input to the Counter Input block.

5 From **Clock input source**, which determines the clock input source to increment the counter, select

- **input pin rising edge** — Clock edge low to high transitions
- **input pin falling edge** — Clock edge high to low transitions
- **internal clock** — Internal timebase

If you set the **Gate input functionality** parameter to enable when high, latch & reset on edge or enable when low,

latch & reset on edge, you can measure positive or negative pulse lengths in units of the internal timebase. You can use this combination with the National Instruments® PCI/PXI-60xx and National Instruments PCI/PXI-62xx drivers for pulse width measurement.

Not all counter chips support selecting the input edge. In this case, only supported options appear in the pull-down menu.

- 6** From **Gate input functionality**, which defines the action of the counter's gate pin, select
- none — Gate is disabled.
 - enable when high — Counting is disabled when the gate is low and enabled when the gate is high.
 - enable when low — Counting is disabled when the gate is high and enabled when the gate is low.
 - start on rising edge — Counting is disabled until low to high transition of the gate occurs.
 - start on falling edge — Counting is disabled until high to low transition of the gate occurs.
 - reset on rising edge — Counter is reset when low to high transition of the gate occurs.
 - reset on falling edge — Counter is reset when high to low transition of the gate occurs.
 - latch on rising edge — The count of the counter is remembered when low to high transition of the gate occurs.
 - latch on falling edge — The count of the counter is remembered when high to low transition of the gate occurs.
 - latch & reset on rising edge — The count of the counter is remembered and then the counter is reset when low to high transition of the gate occurs.

Counter Input

- **latch & reset on falling edge** — The count of the counter is remembered and then the counter is reset when high to low transition of the gate occurs.
- **enable when high, latch & reset on edge** — Enables pulse counting when the gate input goes high, counts the clock while it is high, and remembers (latches) the pulse count. Then resets the counter to prepare it for the next pulse in the gate input.

If you set the **Clock input source** parameter to **internal clock**, you can measure positive pulse lengths in units of the internal timebase. You can use this combination with the National Instruments PCI/PXI-60xx and National Instruments PCI/PXI-62xx drivers for pulse width measurement.

- **enable when low, latch & reset on edge** — Enables pulse counting when the gate input goes high, counts the clock while it is low, and remembers (latches) the pulse count. Then resets the counter to prepare it for the next pulse in the gate input.

If you set the **Clock input source** parameter to **internal clock**, you can measure negative pulse lengths in units of the internal timebase. You can use this combination with the National Instruments PCI/PXI-60xx and National Instruments PCI/PXI-62xx drivers for pulse width measurement.

Not all counter chips support all gate modes. Only supported gate modes appear in the pull-down menu.

7 Set **Output data type** to specify the type of data that the block will output to the model.

8 Click **OK** or **Apply**.

Counter Input Block Demo

The Real-Time Windows Target software includes a demo that shows the operation of the Counter Input block. To see this demo, type `rtcounter` in the MATLAB Command Window, or start MATLAB Help

and choose **Real-Time Windows Target > Demos > Frequency Measurement**.

Digital Input

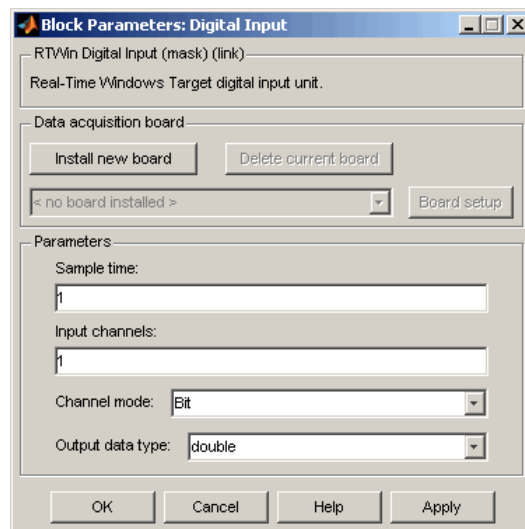
Purpose Select and connect digital input lines or channels

Library Real-Time Windows Target

Description The Digital Input block allows you to select and connect specific digital lines or channels to your Simulink model. After you have added a Digital Input block to your model, you can enter the parameters for its I/O driver. The following procedure uses the Humusoft AD512 I/O board as an example.

- 1 Double-click the Digital Input block.

The Block Parameters: Digital Input dialog box opens:



- 2 In the **Sample time** box, enter the same value you entered in the **Fixed step size** box from the Configuration Parameters dialog box, or an integer multiple of that value.

3 In the **Input channels** box, enter a channel vector that selects the digital input channels you are using on this board. The vector can be any valid MATLAB vector form. For example, to select all eight digital input channels on the AD512 board, enter

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

If you want to use the first four digital input lines, enter

[1,2,3,4]

If you have one 8-bit digital channel, enter [1]. If you have two 8-bit digital channels, enter [1 9], and from the Channel mode list, choose **Byte**.

4 From the Channel mode list, choose one of the following options:

- **Bit** — Returns a value of 0 or 1.
- **Byte** — Groups eight digital lines into one digital channel and returns a value of 0 to 255.

5 Set **Output data type** to specify the type of data that the block will output to the model.

6 Click **OK** or **Apply**.

Digital Output

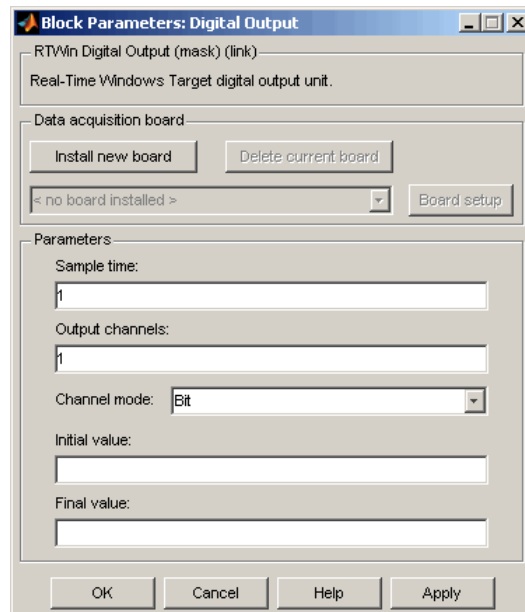
Purpose Select and connect digital output lines or channels

Library Real-Time Windows Target

Description The Digital Output block allows you to select and connect specific digital lines or channels to your Simulink model. After you have added a Digital Output block to your model, you can enter the parameters for its I/O driver. The following procedure uses the Humusoft AD512 I/O board as an example.

- 1 Double-click the Digital Output block.

The Block Parameters: Digital Output dialog box opens:



- 2 In the **Sample time** box, enter the same value you entered in the **Fixed step size** box from the Configuration Parameters dialog box, or an integer multiple of that value.

- 3** In the **Output channels** box, enter a channel vector that selects the digital output channels you are using on this board. The vector can be any valid MATLAB vector form. For example, to select all eight digital output channels on the AD512 board, enter

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

If you want to use the first four digital output lines, enter

```
[1,2,3,4]
```

If you have one 8-bit digital channel, enter [1]. If you have two 8-bit digital channels, enter [1 9], and from the Channel mode list, choose Byte.

- 4** From the Channel mode list, choose from one of the following:

- Bit — Expects a value of 0 or 1.
- Byte — Expects a value of 0 to 255 that is converted to one digital channel of eight digital lines.

- 5** Enter the initial values for each digital output line or channel you entered in the **Output channels** box. For example, if you entered [1,2,3,4] in the **Output channels** box, and you want initial values of 0 and 1, enter

```
[0,0,1,1]
```

If you choose Byte from the Channel mode list, enter a value between 0 and 255 for each digital output channel. For example, for one byte (8 digital lines) with an initial value of 25, enter [25]. For two bytes (16 digital lines) with initial values of 25 and 50, enter [25 50].

- 6** Enter a final value for each digital output channel you entered in the **Output channels** box. For example, if you entered [1,2,3,4] in the **Output channels** box, and you want final values of 0, enter

```
[0,0,0,0]
```

Digital Output

If you choose Byte from the Channel mode list, enter a value between 0 and 255 for each digital output channel.

7 Click **OK** or **Apply**.

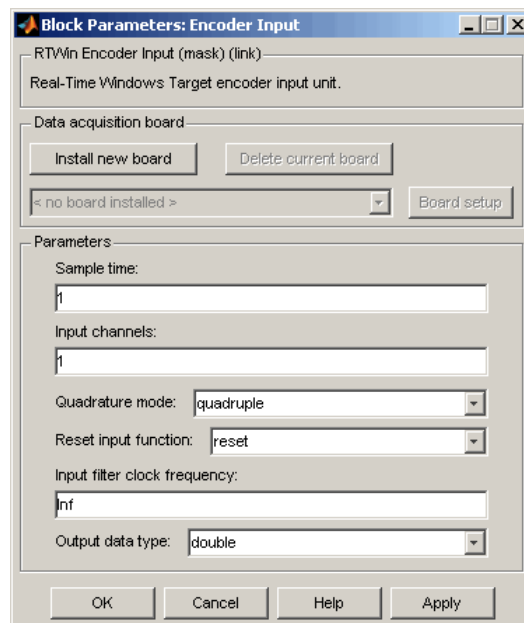
Purpose Select and connect encoder input channels

Library Real-Time Windows Target

Description The Encoder Input block allows you to select and connect specific encoder input channels to your Simulink model. After you have added an Encoder Input block to your model, you can enter the parameters for its I/O driver. The following procedure uses the Humusoft MF604 I/O board as an example.

- 1 Double-click the Encoder Input block.

The Block Parameters: Encoder Input dialog box opens:



Encoder Input

- 2** In the **Sample time** box, enter the same value you entered in the **Fixed step size** box from the Configuration Parameters dialog box, or an integer multiple of that value.
- 3** In the **Input channels** box, enter a channel vector that selects the encoder input channels you are using on this board. The vector can be any valid MATLAB vector form. For example, to select all four encoder input channels on the MF604 board, enter

[1,2,3,4] or [1:4]

- 4** Encoders typically use two sets of stripes, shifted in phase, to optically detect the amplitude and direction of movement. The **Quadrature mode** parameter specifies which encoder stripe edges should be counted.
 - **double** — Counts the rising edges from both stripe sets
 - **single** — Counts the rising edges from one stripe set
 - **quadruple** — Counts rising and falling edges from both stripe sets

Quadruple mode yields four times more pulses per revolution than the single mode. Therefore, quadruple is more precise and is recommended unless other parameters dictate otherwise.

- 5** The encoder interface chip has a reset pin in addition to encoder inputs. This pin is usually connected to the index output of the encoder. However, it can be connected to any signal or not be used at all. The **Reset input function** specifies the function of this pin.
 - **gate** — Enables encoder counting
 - **reset** — Level reset of the encoder count
 - **rising edge index** — Resets the encoder count on the rising edge
 - **falling edge index** — Resets the encoder count on the falling edge

- 6 The encoder interface chip has a built-in lowpass filter that attempts to filter out any high frequencies, which are interpreted as noise. The **Input filter clock frequency** is the cutoff frequency (Hz) of this filter. The cutoff frequency you specify is rounded to the nearest frequency supported by the chip.

If the encoder is moving slowly and high-frequency noise is present, employ the filter to eliminate the noise. This keeps the noise from being counted as encoder pulses. If the encoder is moving quickly, the filter can filter out all of the high-frequency pulses, including those you want to count. In this case, consider leaving the filter disabled by setting the cutoff frequency to Inf.

- 7 Set **Output data type** to specify the type of data that the block will output to the model.
- 8 Click **OK** or **Apply**.

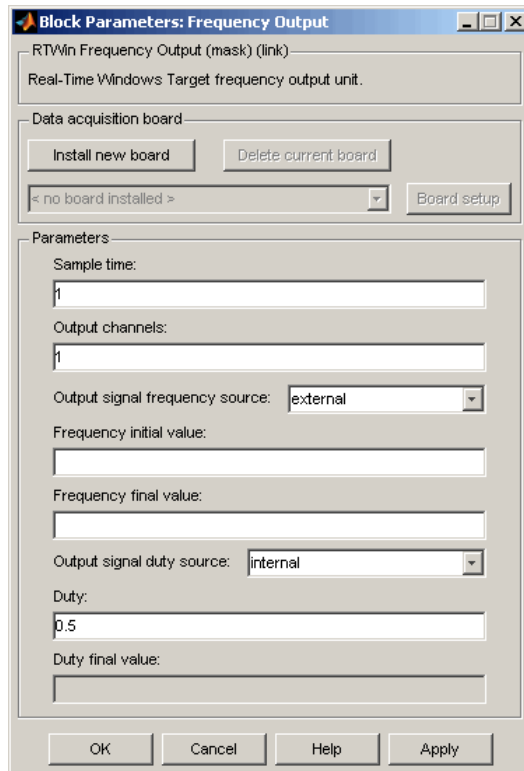
Frequency Output

Purpose Generate and output a pulse-width-modulated square wave

Library Real-Time Windows Target

Description The Frequency Output block generates a pulse-width-modulated square wave that alternates between low (0) and high (1) with a specified frequency and duty cycle. The frequency is specified in Hertz. The duty cycle is specified as a decimal fraction between 0 and 1 inclusive. This fraction specifies the amount of time that the output signal value is high (1). For example, a duty cycle of .7 specifies that the output is high 70% of the time, and low 30% of the time.

After you have added a Frequency Output block to your model, double-click the Frequency Output block to open the Block Parameters: Frequency Output dialog box. The default appearance of the dialog box is:



Specify parameter values as needed, then click **OK** or **Apply**. If you specify a non-default value for **Output signal frequency source** or **Output signal duty source**, the appearance of the dialog box changes as shown below. The parameters are:

Sample time

Enter the same value you entered in the **Fixed step size** box from the Configuration Parameters dialog box, or an integer multiple of that value.

Output channels

A vector that selects the output channels you are using on this board. The vector can be any valid MATLAB vector form.

Frequency Output

Output signal frequency source

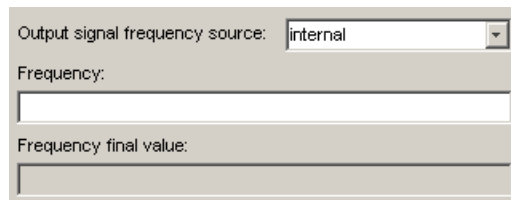
The source that specifies the frequency to output. The frequency is specified in Hertz. The **Output signal frequency source** can be either of the following:

external (Default)

The frequency is specified by an input signal. An input port appears on the block to accept the signal. A **Frequency initial value** and a **Frequency final value** may be needed.

internal

The frequency is specified by a tunable parameter named **Frequency**. If you specify **internal**, **Frequency** replaces **Frequency initial value**, and **Frequency final value** is disabled:



The image shows a configuration dialog box for the 'Output signal frequency source'. At the top, there is a label 'Output signal frequency source:' followed by a dropdown menu currently set to 'internal'. Below this, there are three input fields: 'Frequency:' with an empty text box, 'Frequency final value:' with an empty text box, and a third empty text box at the bottom which is disabled (indicated by a grey background).

Frequency

This tunable parameter appears when **Output signal frequency source** is **internal**. Specify the desired frequency in Hertz.

Frequency initial value

This parameter appears when **Output signal frequency source** is **external**. Optionally specify a frequency in Hertz. The specified frequency takes effect when you connect to the target, and persists until simulation starts, at which time the value of the frequency signal takes effect. You can use this parameter to specify initial conditions and give them time to stabilize. If no value appears, connecting to the target has no effect on the frequency.

Frequency final value

This parameter appears when **Output signal frequency source** is external. Optionally specify a frequency in Hertz. The specified frequency takes effect when simulation completes and persists indefinitely. Disconnecting from the target does not affect the frequency. You can use this parameter to put a connected device into a safe or neutral state after simulation. If no value appears, the frequency in effect at the end of simulation persists afterwards.

Output signal duty source

The source that specifies the duty cycle. The specification is a decimal fraction between 0 and 1 inclusive that determines the amount of time that the output signal value is high (1). For example, a duty cycle of .7 specifies that the output is high 70% of the time, and low 30% of the time. Two duty cycle values have special significance:

0

The output signal is continuously low (0) regardless of any frequency specification currently in effect.

1

The output signal is continuously high (1) regardless of any frequency specification currently in effect.

The **Output signal duty source** can be either of the following:

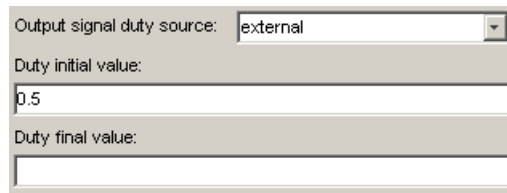
internal (Default)

The duty cycle is specified by a tunable parameter named **Duty**.

external

The duty cycle is specified by an input signal. An input port appears on the block to accept the signal. If you specify external, **Duty initial value** replaces **Duty**, and **Duty final value** is enabled:

Frequency Output



Output signal duty source: external

Duty initial value:
0.5

Duty final value:

Duty

This tunable parameter appears when **Output signal duty source** is **internal**. Specify the desired duty cycle as a decimal fraction between 0 and 1. Specifying 0 or 1 overrides the currently specified frequency and enforces a continuously low or high output signal for the duration of the specification.

Duty initial value

This parameter appears when **Output signal duty source** is **external**. Optionally specify a duty cycle. The specified duty cycle takes effect when you connect to the target, and persists until simulation starts. If no value appears, connecting to the target has no effect on the duty cycle. You can specify any duty cycle. Often the **Duty initial value** is 0 or 1, enforcing a continuously low or high initial output signal. Specifying 0 or 1 overrides any **Frequency initial value**.

Duty final value

This parameter appears when **Output signal duty source** is **external**. Optionally specify a duty cycle. The specified duty cycle takes effect when simulation completes and persists indefinitely. Disconnecting from the target does not change the duty cycle. If no value appears, the duty cycle in effect at the end of simulation persists indefinitely afterwards. You can specify any duty cycle. Often the **Duty final value** is 0 or 1, enforcing a continuously low or high initial output signal. Specifying 0 or 1 overrides any **Frequency final value**.

Purpose	Connect with input sources that other input blocks cannot
Library	Real-Time Windows Target
Description	<p>The Real-Time Windows Target Other Input block can be used for interfacing input signals that other Real-Time Windows Target input blocks do not accommodate. The Other Input block is rarely used, and for only a few drivers. For details, see the documentation for the driver that you intend to use.</p> <p>If none of the available blocks, including Other Input, provide what you need, you can create your own I/O blocks to work with Real-Time Windows Target applications. See “Custom I/O Driver Blocks” for details.</p>

Other Output

Purpose Connect with output sources that other output blocks cannot

Library Real-Time Windows Target

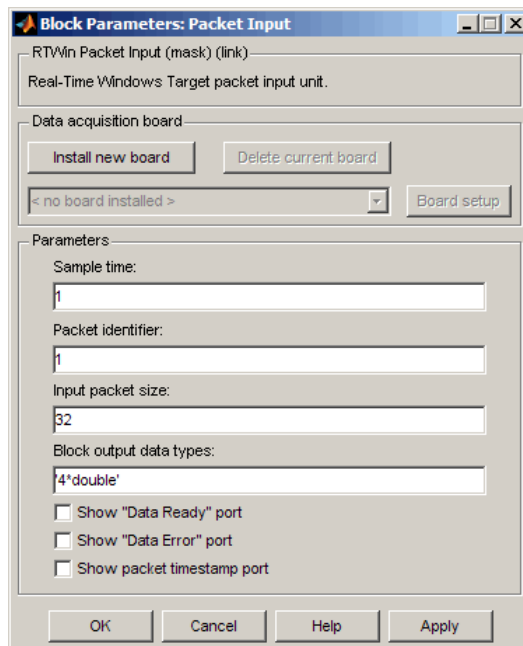
Description The Real-Time Windows Target Other Output block can be used for interfacing input signals that other Real-Time Windows Target output blocks do not accommodate. The Other Output block is rarely used, and for only a few drivers. For details, see the documentation for the driver that you intend to use.

If none of the available blocks, including Other Output, provide what you need, you can create your own I/O blocks to work with Real-Time Windows Target applications. See “Custom I/O Driver Blocks” for details.

Purpose Receive unformatted binary data

Library Real-Time Windows Target

Description The Packet Input block receives unformatted binary data. After you have added a Packet Input block to your model, double-click the Packet Input block to open the Block Parameters: Packet Input dialog box.



The driver must be **Standard Devices > Serial Port**, **Standard Devices > UDP Protocol**, **Standard Devices > File**, or **Vector > CAN Device**. Specify parameter values as needed, then click **OK** or **Apply**. When you install a UDP device, enter port addresses in decimal format in the Standard Devices UDP Protocol dialog box.

The parameters are:

Packet Input

Sample time

Enter the same value you entered in the **Fixed step size** box from the Configuration Parameters dialog box, or an integer multiple of that value.

Packet identifier

Enter the ID of the packet to receive. If your protocol does not have packet IDs, this parameter is disabled.

Input packet size

The number of bytes expected in each input packet. This number must be the same as the number of bytes needed to satisfy the type specifications in **Block output data types**.

Block output data types

A string, or a cell array of strings, that specify how the data in each packet obtained from the device is to be typed and grouped for input to the application. The Packet Input block has an output port corresponding to each string in **Block output data types**. Changing the number of strings automatically changes the number of output ports.

Each string has the format $[n^*]datatype$. The data described by the string has the type specified by *datatype* and the width specified by *n*; or 1 if *n* is not specified. For example, 'double' means one double value, and '4*int8' means a vector of four int8 values.

By providing an appropriate cell array of strings, you can convert an input packet into any needed types, packaged into vectors in any needed way. For example, specifying {'int16', 'int16', 'double'} creates three ports. The first and second each output an int16 value, and the third outputs a double. Specifying {'2*int16', 'double'} creates two ports: a vector of two int16 values, and a scalar double.

Show “Data Ready” port

If enabled, the block has an output port that signals 1 if the block has new data available, and 0 otherwise.

Show “Data Error” port

If enabled, the block has an output port that signals 1 if a data error has occurred, and 0 otherwise.

Show packet timestamp port

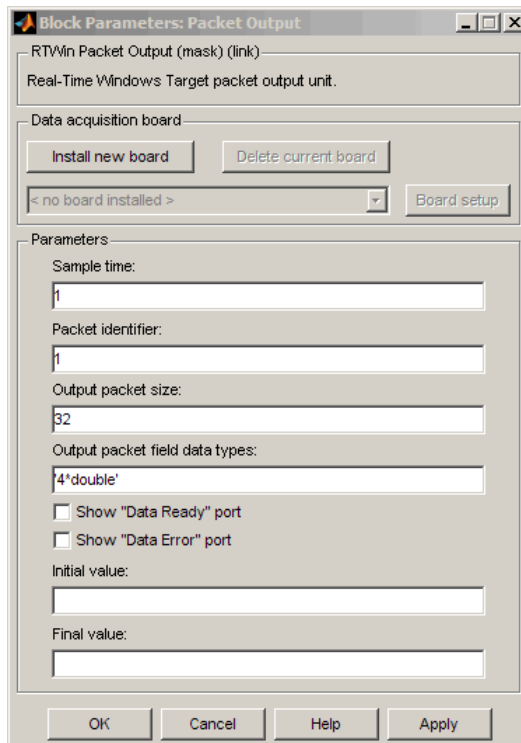
Select this check box to show the timestamp for the CAN message packets. If your protocol does not have packet timestamps, the block ignores this value.

Packet Output

Purpose Transmit unformatted binary data

Library Real-Time Windows Target

Description The Packet Output block sends unformatted binary data. After you have added a Packet Output block to your model, double-click the Digital Output block to open the Block Parameters: Digital Output dialog box.



The driver must be **Standard Devices > Serial Port**, **Standard Devices > UDP Protocol**, **Standard Devices > File**, or **Vector > CAN Device**. Specify parameter values as needed, then click **OK** or **Apply**. When you install a UDP device, enter port addresses in decimal format in the Standard Devices UDP Protocol dialog box.

The parameters are:

Sample time

Enter the same value you entered in the **Fixed step size** box from the Configuration Parameters dialog box, or an integer multiple of that value.

Packet identifier

Enter the ID of the packet to receive. If your protocol does not have packet IDs, this parameter is disabled.

Output packet size

The number of bytes to be transmitted in the output packet. This number must be the same as the number of bytes needed to satisfy the type specifications in **Output packet field data types**.

Output packet field data types

A string, or a cell array of strings, that specify how data provided by the application will be formatted into a packet for output to the device. The Packet Output block has an input port corresponding to each string in **Output packet field data types**. Changing the number of strings automatically changes the number of ports.

Each string has the format $[n*]datatype$. The data described by the string has the type specified by *datatype* and the width specified by *n*; or 1 if *n* is not specified. For example, 'double' means one double value, and '4*int8' means a vector of four int8 values.

The signal input to each port of the Packet Output block can be a scalar or vector of any Simulink data type. The string for each port specifies the type to be used when its signal is output to the device. If the format string for a port matches the type of the signal input to that port, the signal value appears verbatim in the output packet.

You can also perform type conversion on output. For example, if an input signal is a four-element int16 vector, but the

Packet Output

corresponding string is '4*int8', each of the four integers is converted to an int8 before being written to the packet. The resulting data occupies four bytes in the output packet.

Show “Data Ready” port

If enabled, the block has an output port that signals 1 if the block is ready to accept new data, and 0 otherwise.

Show “Data Error” port

If enabled, the block has an output port that signals 1 if a data error has occurred, and 0 otherwise.

Initial value

If specified, a vector that has the same number of elements as the sum of the widths of the input signals across all ports. The specified data is sent when simulation begins, before any other data that is output during simulation.

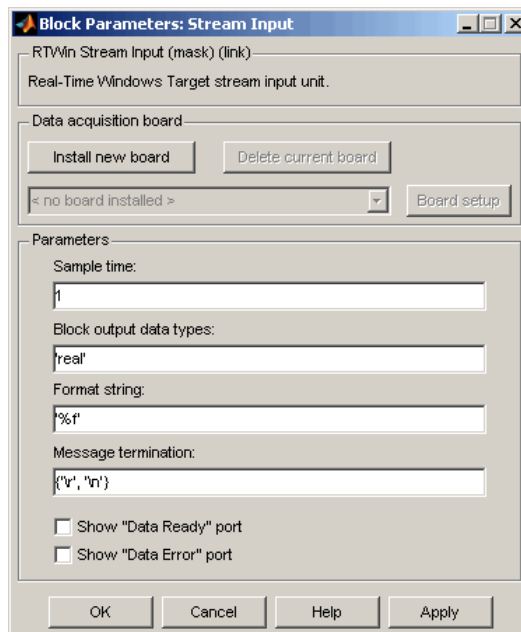
Final value

If specified, a vector that has the same number of elements as the sum of the widths of the input signals across all ports. The specified data is sent when simulation ends, after any other data that is output during simulation.

Purpose Receive formatted ASCII data

Library Real-Time Windows Target

Description The Stream Input block receives formatted ASCII data. After you have added a Stream Input block to your model, double-click the Stream Input block to open the Block Parameters: Stream Input dialog box.



The driver must be **Standard Devices > Serial Port**, **Standard Devices > File**, or **Standard Devices > UDP Protocol**. Specify parameter values as needed, then click **OK** or **Apply**. When you install a UDP device, enter port addresses in decimal format in the Standard Devices UDP Protocol dialog box.

The parameters are:

Sample time

Enter the same value you entered in the **Fixed step size** box from the Configuration Parameters dialog box, or an integer multiple of that value.

Block output data types

A string or a cell array of strings. The block has as many output ports as the number of strings. Each string specifies a data type by name, and optionally a number of elements. For example, 'double' means one double value, and '4*int8' means four int8 values.

The values made available on output ports are grouped and typed according to the **Block output data types** strings. For example, if **Block output data types** is {'2*int8', '3*double'}, the block outputs an int8 vector of width 2 on the first output port, and a double vector of width 3 on the second output port.

Format string

A specification in the same format used by C library I/O routines like scanf. The format string describes the data to be received. The number of elements in the string must equal the number of data items specified in **Block output data types**.

For example, if **Block output data types** is {'2*int8', '3*double'}, and **Format string** is '%d %d %f %f %f', the block reads an ASCII representation of two integers and three doubles. The block makes the resulting values available to the application in an int8 vector of width 2 on the first output port, and a double vector of width 3 on the second output port.

If the data type specified for a value in **Block output data types** differs from the type of the corresponding element in **Format string**, type conversion occurs automatically. The block reads data as specified by **Format string**, converts the data to match the **Block output data types**, and provides the data to the application on the block output ports.

Message termination

A string, cell array of strings, or a number. If the value is a string, receiving this sequence of characters terminates data input. If the value is a cell array, any of the strings in the cell array terminates data input. If the value is a number, data input terminates after reading the specified number of characters.

Show “Data Ready” port

If enabled, the block has an output port that signals 1 if the block has new data available, and 0 otherwise.

Show “Data Error” port

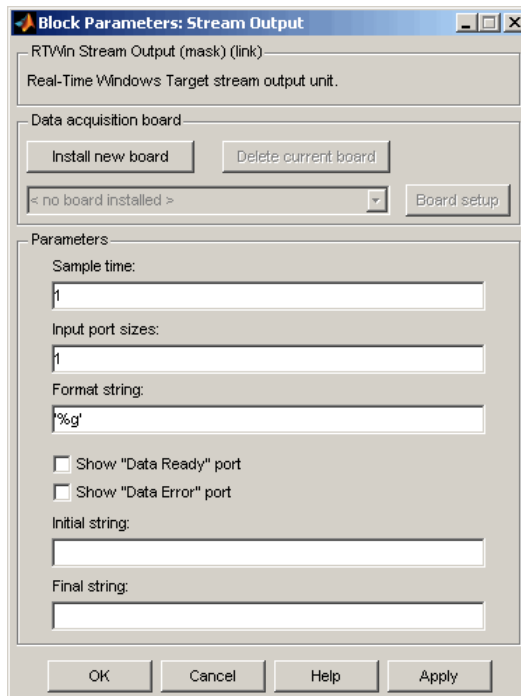
If enabled, the block has an output port that signals 1 if a data error has occurred, and 0 otherwise.

Stream Output

Purpose Transmit formatted ASCII data

Library Real-Time Windows Target

Description The Stream Output block sends formatted ASCII data. After you have added a Stream Output block to your model, double-click the Stream Output block to open the Block Parameters: Stream Output dialog box.



The driver must be **Standard Devices > Serial Port**, **Standard Devices > File**, or **Standard Devices > UDP Protocol**. Specify parameter values as needed, then click **OK** or **Apply**. When you install a UDP device, enter port addresses in decimal format in the Standard Devices UDP Protocol dialog box.

The parameters are:

Sample time

Enter the same value you entered in the **Fixed step size** box from the Configuration Parameters dialog box, or an integer multiple of that value.

Input port sizes

A port width, or vector of port widths. The number of elements determines the number of input ports. Each port has the width specified by the corresponding element. For example, specifying **3, 1, 2** indicates that the block has three input ports. The first is a vector of width 3, the second is a scalar, and the third is a vector of width 2, for a total of six elements. Only the widths need to be specified; any Simulink data type can be input.

Format string

A specification in the same format used by C library I/O routines like `printf`. The format string describes the data to be sent. The format string must have the same number of specifiers as the sum of the elements of **Input port sizes**.

For example, assume that **Input port sizes** is **3, 1, 2**. Then a **Format string** of `'%d %d %d %f %d %d'` would output ASCII representing: three integers from the vector on the first input port; a double from the scalar on the input second port; and two integers from the vector on the third input port.

If the data type specified in the **Format string** for a value differs from the type of the actual value, type conversion occurs automatically, and converted data that conforms to the **Format string** is output as ASCII to the device.

Show “Data Ready” port

If enabled, the block has an output port that signals 1 if the block is ready to accept new data, and 0 otherwise.

Show “Data Error” port

If enabled, the block has an output port that signals 1 if a data error has occurred, and 0 otherwise.

Stream Output

Initial string

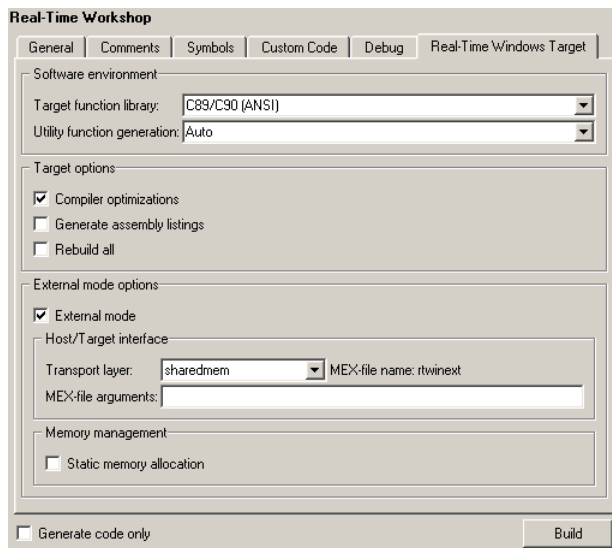
If specified, a string that is sent when simulation begins, before any other data that is output during simulation. The string can contain anything; it could be used to initialize a device. It is sent literally as specified, without translation using the **Format string**.

Final string

If specified, a string that is sent when simulation ends, after any other data that is output during simulation. The string can contain anything; it could be used to shut down a device. It is sent literally as specified, without translation using the **Format string**.

Configuration Parameters

Real-Time Workshop Pane: Real-Time Windows Target



In this section...

“Real-Time Windows Target Tab Overview” on page 4-3

“Target function library” on page 4-4

“Utility function generation” on page 4-6

“Compiler optimizations” on page 4-7

“Generate assembly listings” on page 4-8

“Rebuild all” on page 4-9

“External mode” on page 4-10

“Transport layer” on page 4-12

“MEX-file arguments” on page 4-14

“Static memory allocation” on page 4-16

“Static memory buffer size” on page 4-18

Real-Time Windows Target Tab Overview

Control the code created by Real-Time Workshop® code generation software for a Real-Time Windows Target application.

Configuration

This tab appears only if you specify `rtwin.tlc` as the System target file.

See Also

- Real-Time Windows Target User's Guide
- Real-Time Windows Target Reference Manual
- Real-Time Windows Target Release Notes

Target function library

Specify a floating-point math library extension.

Settings

Default: C89/C90 (ANSI)

C89/C90 (ANSI)

Generates calls to the ISO/IEC 9899:1990 C standard math library for floating-point functions.

C99 (ISO)

Generates calls to the ISO/IEC 9899:1999 C standard math library.

GNU99 (GNU)

Generates calls to the GNU gcc math library, which provides C99 extensions as defined by compiler option `-std=gnu99`.

C++ (ISO)

Generates calls to the ISO/IEC 14882:2003 C++ standard math library. This setting is visible only if you selected C++ for the **Language** parameter on the Real-Time Workshop pane of the Configuration Parameters dialog box.

Tips

- Before setting this parameter, verify that your compiler supports the library you want to use. If you select a parameter value that your compiler does not support, compiler errors can occur.
- **Restriction** — Stateflow[®] software supports only C89/C90 (ANSI). Selecting a different parameter has no effect on code generated for Stateflow components.

Command-Line Information

Parameter: GenFloatMathFcnCalls

Type: string

Value: 'ANSI_C' | 'C99 (ISO)' | 'GNU99 (GNU)' | 'C++ (ISO)'

Default: 'ANSI_C'

Recommended Settings

Application	Setting
Debugging	No impact
Traceability	No impact
Safety precaution	No impact

Utility function generation

Specify location for generating utility functions

Settings

Default: Auto

Auto

Operates as follows:

- When the model contains Model blocks, place utilities within the `slprj/target/_sharedutils` directory.
- When the model does not contain Model blocks, place utilities in the build directory (generally, in `model.c` or `model.cpp`).

Shared location

Directs code for utilities to be placed within the `slprj` directory in your working directory.

Command-Line Information

Parameter: UtilityFuncGeneration

Type: string

Value: 'Auto' | 'Shared location'

Default: 'Auto'

Recommended Settings

Application	Setting
Debugging	Shared location
Traceability	Shared location
Efficiency	Shared location
Safety precaution	No impact

Compiler optimizations

Instruct the C compiler to generate real-time code fully optimized for speed.

Settings

Default: on



On

Optimizes the code for speed.



Off

Performs no optimizations.

Command-Line Information

Parameter: CCOptimize

Type: Boolean

Value: on | off

Default: on

Recommended Settings

Application	Setting
Debugging	Off
Traceability	Off
Efficiency	On
Safety precaution	On

Generate assembly listings

Instruct the C compiler to generate assembly listings for the generated code.

Settings

Default: off

On
Generates assembly listings.

Off
Suppresses assembly listings.

Command-Line Information

Parameter: CCListing

Type: Boolean

Value: on | off

Default: off

Recommended Settings

Application	Setting
Debugging	No impact
Traceability	No impact
Efficiency	No impact
Safety precaution	No impact

Rebuild all

Force all the object files to always be rebuilt regardless of their time stamps.

Settings

Default: off



On

Rebuilds all object files at every build.



Off

Rebuilds only object files whose time stamps show they are outdated.

Command-Line Information

Parameter: RebuildAll

Type: Boolean

Value: on | off

Default: off

Recommended Settings

Application	Setting
Debugging	On
Traceability	No impact
Efficiency	No run-time impact. (Build may take longer.)
Safety precaution	On

External mode

Enable client/server communication between Simulink software and an application.

Settings

Default: on

- On
Enables external mode.
- Off
Disables external mode.

Dependencies

Selecting **External mode** enables:

- **Transport layer**
- **MEX-file arguments**
- **Static memory allocation**

Command-Line Information

Parameter: ExtMode

Type: string

Value: 'on' | 'off'

Default: 'on'

Recommended Settings

Application	Setting
Debugging	No impact
Traceability	No impact
Efficiency	No impact
Safety precaution	No impact

See Also

External Mode

Transport layer

Specify the transport protocol for external mode communication.

Settings

Default: sharedmem

sharedmem

Uses a shared memory transport mechanism. The MEX-file name is `rtwinext`.

Tips

- The MEX-file name displayed next to **Transport layer** cannot be edited in the Configuration Parameters dialog box.
- For targets provided by The MathWorks, like Real-Time Windows Target, the MEX-file name is specified in:

```
matlabroot/toolbox/simulink/simulink/extmode_transports.m
```

Dependency

This parameter is enabled by checking **External Mode**.

Command-Line Information

Parameter: ExtModeTransport

Type: integer

Value: 0

Default: 0

Recommended Settings

Application	Setting
Debugging	No impact
Traceability	No impact

Application	Setting
Efficiency	No impact
Safety precaution	No impact

See Also

- Creating an External Mode Communication Channel
- Target Interfacing

MEX-file arguments

Specify external mode MEX arguments.

Settings

Default: ''

For a shared memory transport, `rtwinext` allows two optional arguments. The arguments are positional, so the first argument must appear if the second appears.

- An argument that is currently unused. Specify '' for this argument if necessary.
- An argument that specifies verbosity. Specify 0 for Nonverbose, or 1 for Verbose.

The default is Nonverbose mode (0). To specify Verbose mode, set **MEX-file arguments** to:

```
'' 1
```

where '' is the empty string and 1 specifies Verbose mode.

Dependency

This parameter is enabled by checking **External Mode**.

Command-Line Information

Parameter: ExtModeMexArgs

Type: string followed by integer

Value: '' | '' 0 | '' 1

Default: ''

Recommended Settings

Application	Setting
Debugging	No impact
Traceability	No impact
Efficiency	No impact
Safety precaution	No impact

See Also

- Target Interfacing
- Client/Server Implementations

Static memory allocation

Control the memory buffer for external mode communication.

Settings

Default: off



On

Enables the **Static memory buffer size** parameter to allocate dynamic memory.



Off

Uses a static memory buffer for external mode instead of allocating dynamic memory (calls to malloc).

Tip

To determine how much memory you need to allocate, enable verbose mode on the target to display the amount of memory it tries to allocate and is available.

Dependencies

- This parameter is enabled by checking **External Mode**.
- This parameter enables **Static memory buffer size**.

Command-Line Information

Parameter: ExtModeStaticAlloc

Type: string

Value: 'on' | 'off'

Default: 'off'

Recommended Settings

Application	Setting
Debugging	No impact
Traceability	No impact

Application	Setting
Efficiency	No impact
Safety precaution	No impact

See Also

External Mode Interface Options

Static memory buffer size

Specify the memory buffer size for external mode communication.

Settings

Default: 1000000

Enter the number of bytes to preallocate for external mode communications buffers in the target.

Tips

- If you enter too small a value for your application, external mode issues an out-of-memory error.
- To determine how much memory you need to allocate, enable verbose mode on the target to display the amount of memory it tries to allocate and is available.

Dependency

This parameter is enabled by **Static memory allocation**.

Command-Line Information

Parameter: ExtModeStaticAllocSize

Type: integer

Value: Any valid size

Default: 1000000

Recommended Settings

Application	Setting
Debugging	No impact
Traceability	No impact
Efficiency	No impact
Safety precaution	No impact

See Also

External Mode Interface Options

A

- Analog Input block
 - configuring 3-2
- Analog Output block
 - configuring 3-4

C

- configuration parameters
 - pane
 - Compiler optimizations 4-7
 - External mode 4-10
 - Generate assembly listings 4-8
 - Rebuild all 4-9
 - Real-Time Windows Target tab 4-3
 - real-time workshop (interface)
 - target floating-point math environment 4-4
 - utility function 4-6
 - Real-Time Workshop (interface)
 - mex-file arguments 4-14
 - static memory allocation 4-16
 - static memory buffer size 4-18
 - transport layer 4-12
- configuration set
 - specifying default 1-3
- configuring
 - Analog Input block 3-2
 - Analog Output block 3-4
 - Counter Input block 3-7
 - Digital Input block 3-12
 - Digital Output block 3-14
 - Encoder Input block 3-17
 - Frequency Output block 3-20
 - Packet Input block 3-27
 - Packet Output block 3-30
 - Stream Input block 3-33
 - Stream Output block 3-36
- Counter Input block
 - configuring 3-7

D

- Digital Input block
 - configuring 3-12
- Digital Output block
 - configuring 3-14

E

- Encoder Input block
 - configuring 3-17

F

- Frequency Output block
 - configuring 3-20

I

- I/O blocks
 - Analog Input block 3-2
 - Analog Output block 3-4
 - Counter Input block 3-7
 - Digital Input block 3-12
 - Digital Output block 3-14
 - Encoder Input block 3-17
 - Frequency Output block 3-20
 - Packet Input block 3-27
 - Packet Output block 3-30
 - Stream Input block 3-33
 - Stream Output block 3-36

P

- Packet Input block
 - configuring 3-27
- Packet Output block
 - configuring 3-30

R

- Real-Time Windows Target kernel

- installing 1-5
- removing 1-5
- rtwho function 1-2
- rtwinconfigset function 1-3
- rtwintgt function 1-5

S

- Stream Input block
 - configuring 3-33
- Stream Output block
 - configuring 3-36