UEISim User Manual 2.0

July 2009 Edition

United Electronic Industries
The High-Performance Alternative

Table of contents

United Electronic Industries
The High-Performance Alternative

1. Introduction

UEISim turns a PowerDNx Ethernet data acquisition module into a target on which you can run Simulink models and read/write physical I/Os.

The UEISim host software uses the Simulink add-on “Real-time Workshop” to convert your Simulink model to C code and then cross-compiles it into an executable that runs directly on the UEISim hardware.

You can access all the analog I/Os, digital I/Os, counter timer I/Os offered by PowerDNA from your Simulink model.

You can experiment with control system design, signal processing, data acquisition and similar tasks directly from the Simulink environment using its powerful block library without the need to use any additional tool.

2. Software Installation

The UEISim software runs on a Linux PC or on Windows.

2.1. Pre-requisites

Before installing the UEISim software make sure that the following software is installed on your computer:

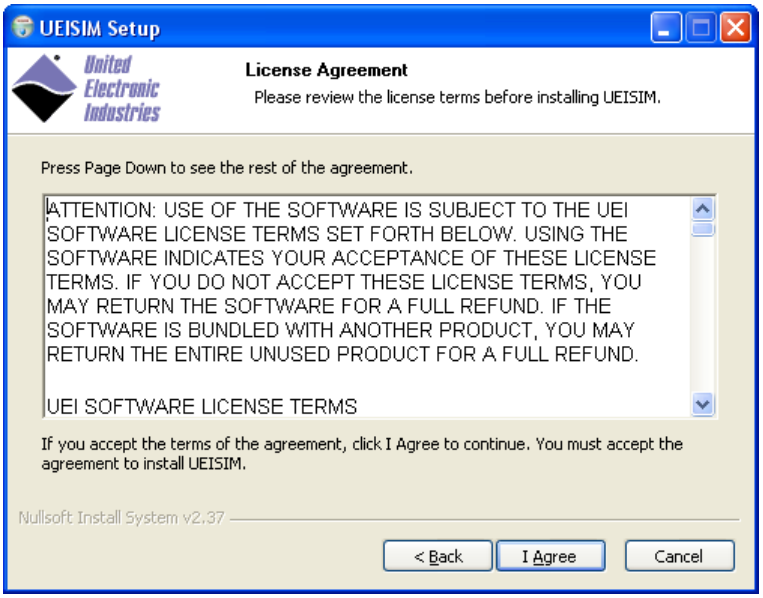
- Matlab R2007b, R2008a, R2008b or R2009a
- Simulink
- Real-time Workshop

2.2. Install UEISim Software for Windows

Insert the UEISIM Software CDROM in your CD drive. If the installer doesn’t start automatically (it depends on whether autorun is enabled or disabled on your PC) run the ueisim_installer.exe program on the CD-ROM.

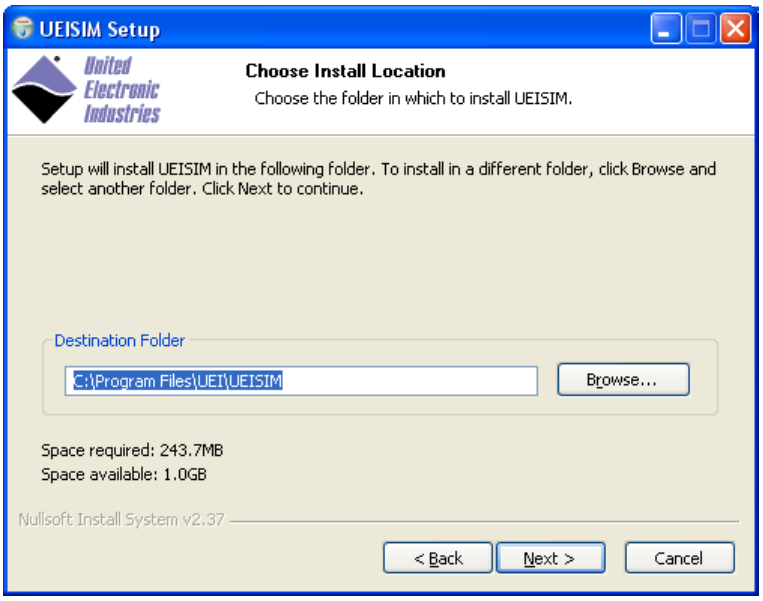
United Electronic Industries
The High-Performance Alternative

Click on Next to move to the next wizard page.

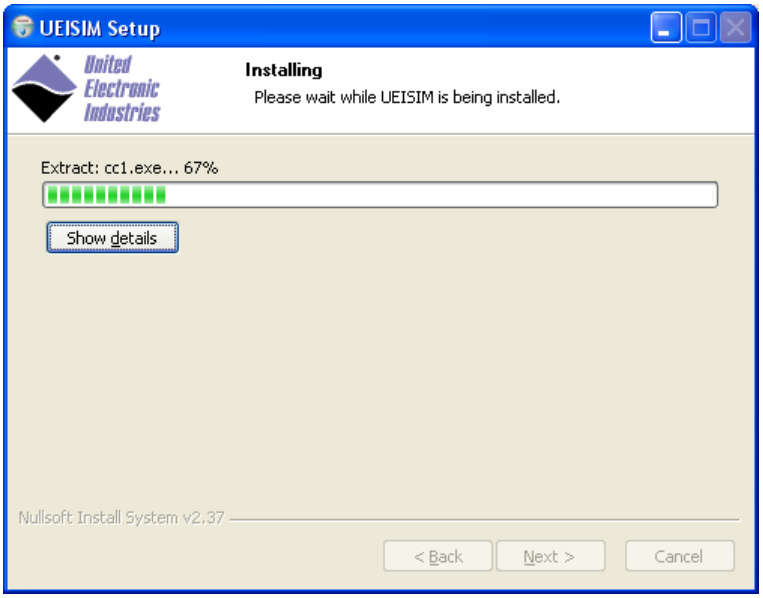


Read the license agreement and click on “I Agree” if you accept the terms of the agreement.

United Electronic Industries
The High-Performance Alternative

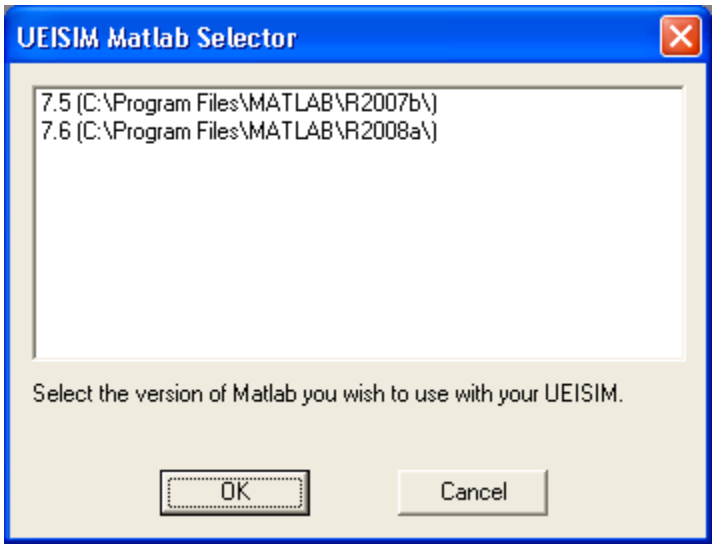


Select the location on your hard drive where you wish to install the software then click “Install”. You need to have at least 250MB of free space.



Once the files are installed, the “UEISIM Matlab Selector” applet will pop-up, letting you select which version of Matlab/Simulink you wish to use with your UEISIM.

United Electronic Industries
The High-Performance Alternative



After the installation is done, you can run that applet again if you want to configure another version of Matlab/Simulink to work with your UEISIM.
You can run the "UEISIM Matlab selector" using the shortcut in the Start/Programs/UEI/UEISIM menu.

Once all the files are installed, click on "Finish" to exit the installer.

United Electronic Industries
The High-Performance Alternative

Important Note: In a few rare occasions, we encountered a problem where the Matlab's ActiveX automation server was not properly registered which prevented our "UEISIM Matlab Selector" applet to work.
When that happens the "UEISIM Matlab Selector" applet will pop-up an error message and you will need to manually configure Matlab's path:
Start Matlab and at the prompt enter the following commands (change the path to the location you selected during the installation):

```
addpath(`c:\program files\uei\ueisim\simulink`)
savepath
```

2.3. Install UEISim Software for Linux

Insert the "UEISim" CDROM in your CD drive. You might need to mount it if your Linux distribution doesn't detect the CDROM automatically.
To mount it, type:

```
mount /dev/cdrom /mnt/cdrom
cd /mnt/cdrom
bash install.sh
```

3. Configuring the UEISim

The IP address must be configured using the serial port.

3.1. Connecting through the serial port

Connect the serial cable to the serial port on the UEISIM cube and the serial port on your PC.

You will need a serial communication program:

- Windows: ucon, MTTTY or HyperTerminal.
- Linux: minicom or cu (part of the uucp package).

The PowerDNA I/O module uses the serial port settings: 57600 bits/s, 8 data bits, 1 stop bit and no parity. Run your serial terminal program and configure the serial communication settings accordingly.

Connect the DC output of the power supply (24VDC) to the "Power In" connector on the PowerDNA cube and connect the AC input on the power supply to an AC power source.

You should see the following message on your screen:

United Electronic Industries
The High-Performance Alternative

```
U-Boot 1.1.4 (Jan 10 2006 - 19:20:03)

CPU:   MPC5200 v1.2 at 396 MHz
       Bus 132 MHz, IPB 66 MHz, PCI 33 MHz

Board: UEI PowerDNA MPC5200 Layer
I2C:   85 kHz, ready
DRAM:  128 MB
Reserving 349k for U-Boot at: 07fa8000
FLASH:  4 MB
In:    serial
Out:   serial
Err:   serial
Net:   FEC ETHERNET

Type "run flash_nfs" to mount root filesystem over NFS

Hit any key to stop autoboot:  5

## Booting image at ffc10000 ...
   Image Name:   Linux-2.6.16.1
   Created:      2006-11-10  16:07:06 UTC
   Image Type:   PowerPC Linux Kernel Image (gzip compressed)
   Data Size:    917636 Bytes = 896.1 kB
   Load Address: 00000000
   Entry Point:  00000000
   Verifying Checksum ... OK
   Uncompressing Kernel Image ... OK
id mach(): done
...
< lots of kernel messages >
...
BusyBox v1.2.2 (2006.11.03-19:16+0000) Built-in shell (ash)
Enter 'help' for a list of built-in commands.

~ #
```

You can now navigate the file system and enter standard Linux commands such as ls, ps, cd…

3.2. Configuring the IP address

Your UEISIM cube is configured at the factory with the IP address 192.168.100.2 to be part of a private network.

You can change the IP address for the current session using the command:

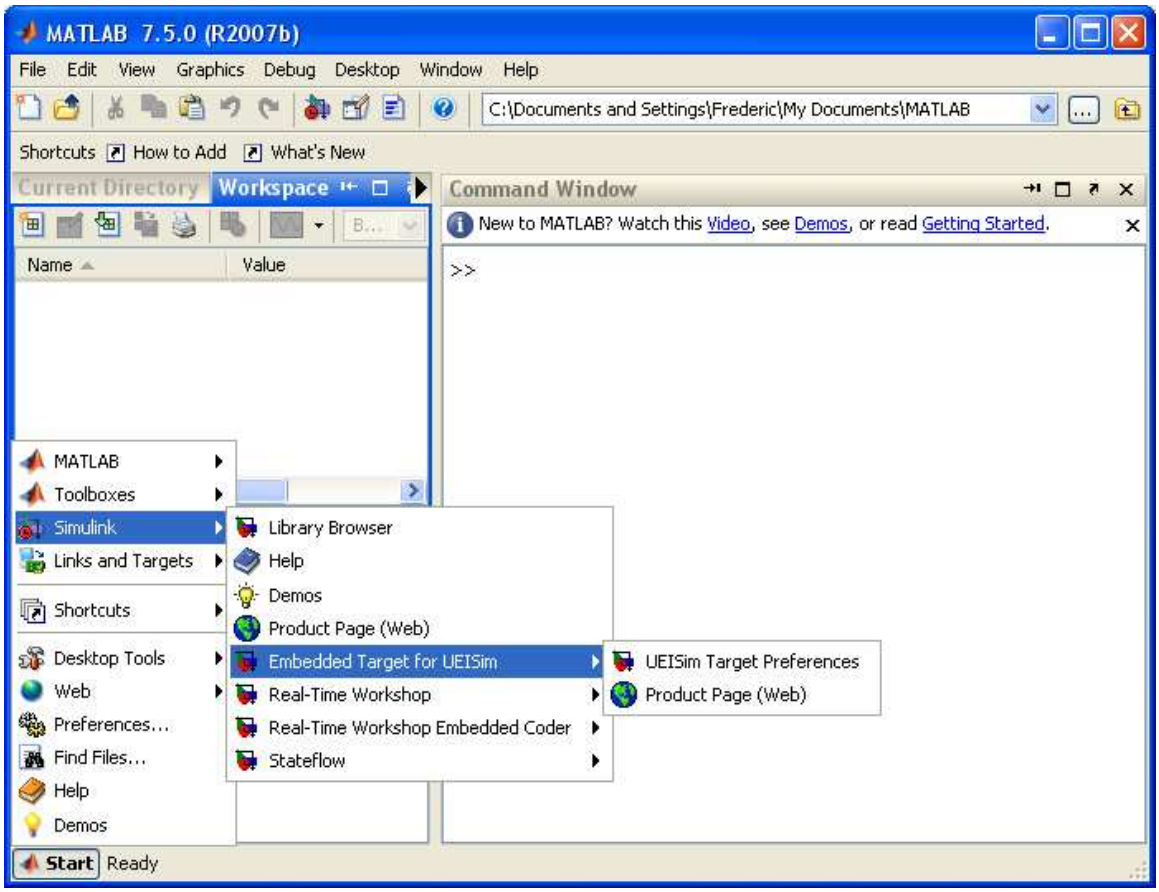
```
setip <new IP address>
```

United Electronic Industries
The High-Performance Alternative

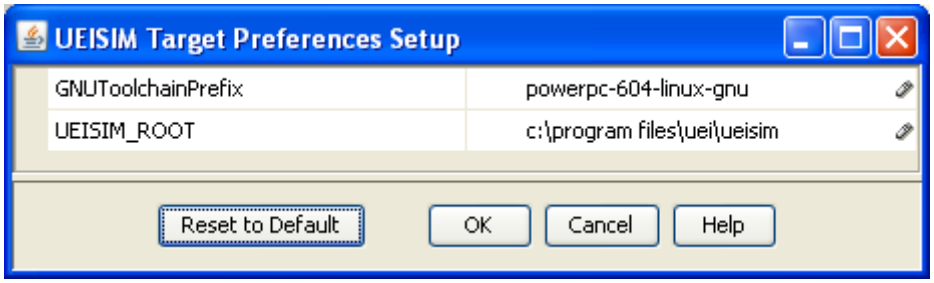
4. Using UEISim add-on from MATLAB/Simulink

4.1. Configuration

Start MATLAB then click on the Start button at the bottom left corner of MATLAB's window.



Select Simulink/Embedded Target for UEISim/UEISim Target Preferences



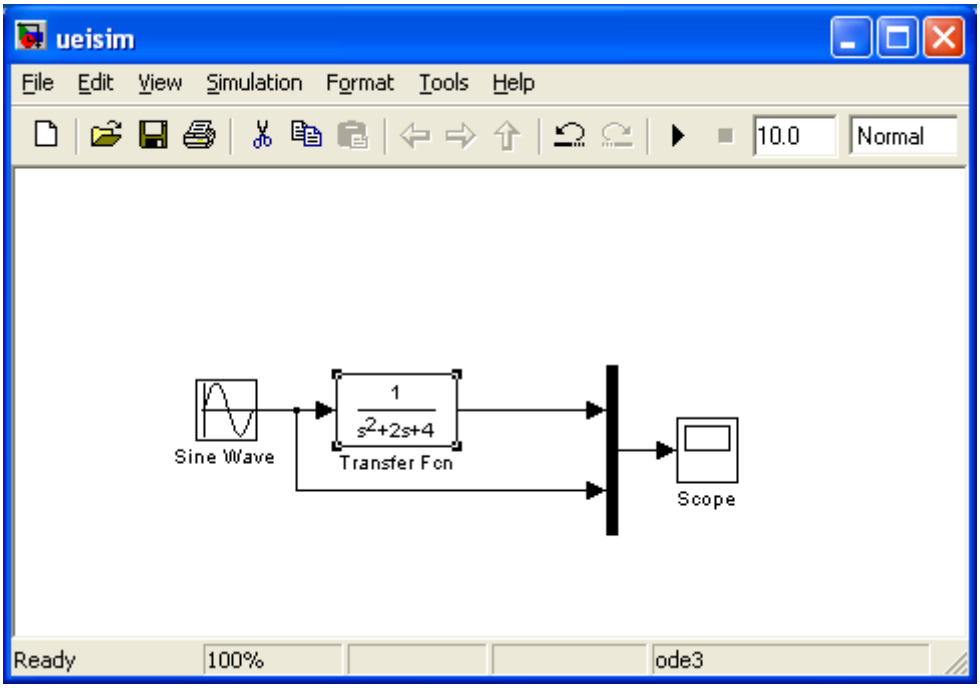
United Electronic Industries
The High-Performance Alternative

GNUToolchainPrefix: specifies the name of the cross-compiling tools used to build a model to a binary that can run on the UEISIM. The default value is correct, don't change it unless told by UEI technical support.

UEISIM_ROOT: The location of the folder where you installed the UEISIM software. Make sure it matches the folder you specifed while running the UEISM Software installer.

4.2. Convert your model

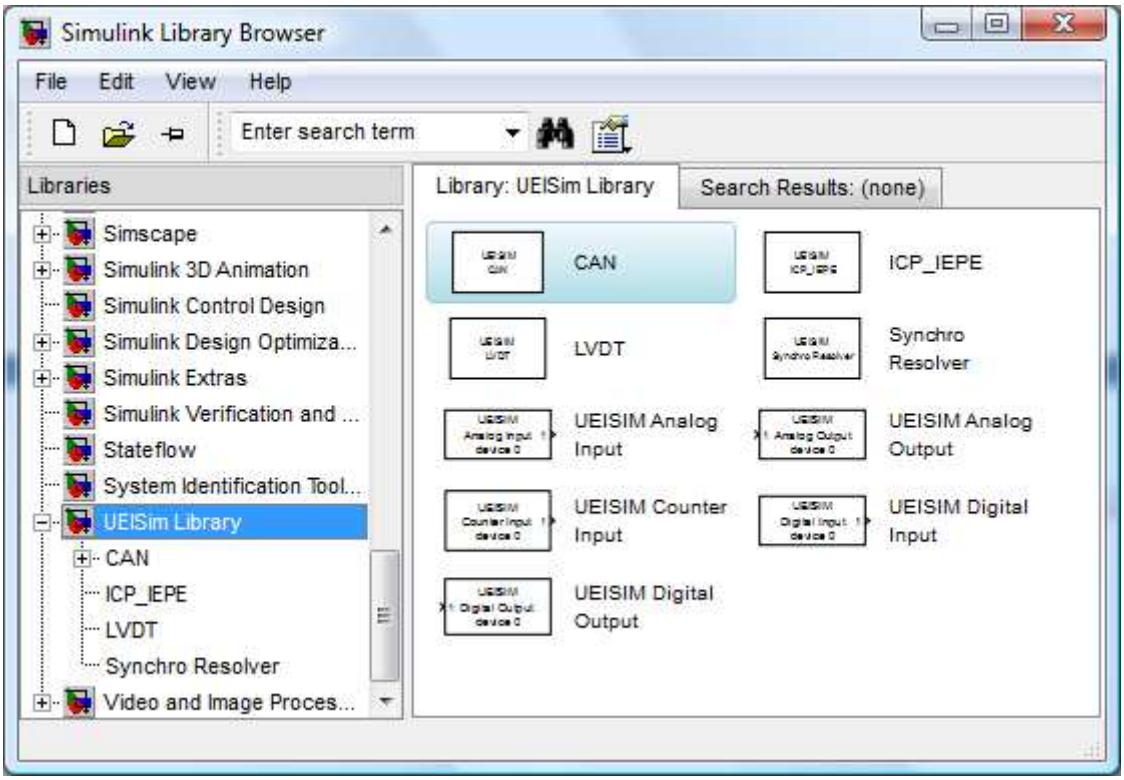
Let's start with an existing model that process some input signal and view the output on a scope.



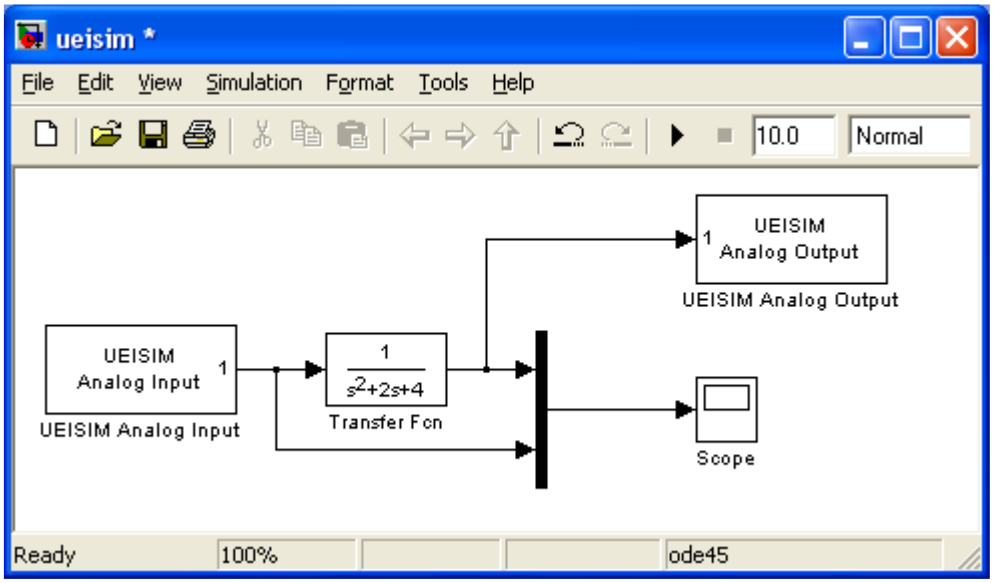
In order to test our model with a real signal, let's use the UEISim analog input and output blocks.

The UEISim I/O blocks are located in the Simulink library:

United Electronic Industries
The High-Performance Alternative



Replace the input sine wave block with an Analog Input block and add an Analog Output block to generate the result as well as display it on the scope.



United Electronic Industries
The High-Performance Alternative

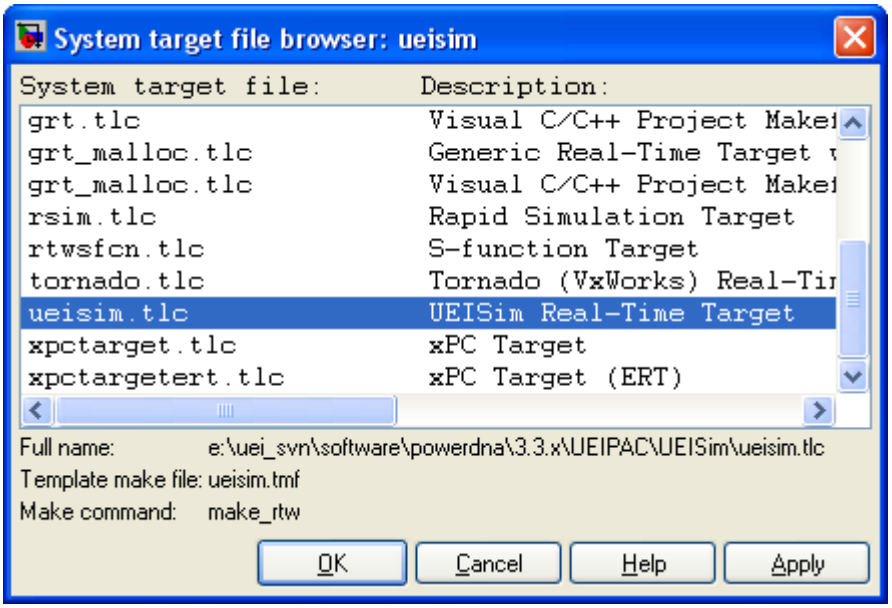
Double-click on the Analog Input and Output blocks to configure the parameters (see chapter 5 for details on the parameters for each of the UEISIM block).

4.3. Create an executable from the model

Select the menu option “Simulation/Configuration Parameters…”

Click on the “Solver” option on the left pane and make sure the solver type is set to “Fixed-step”.

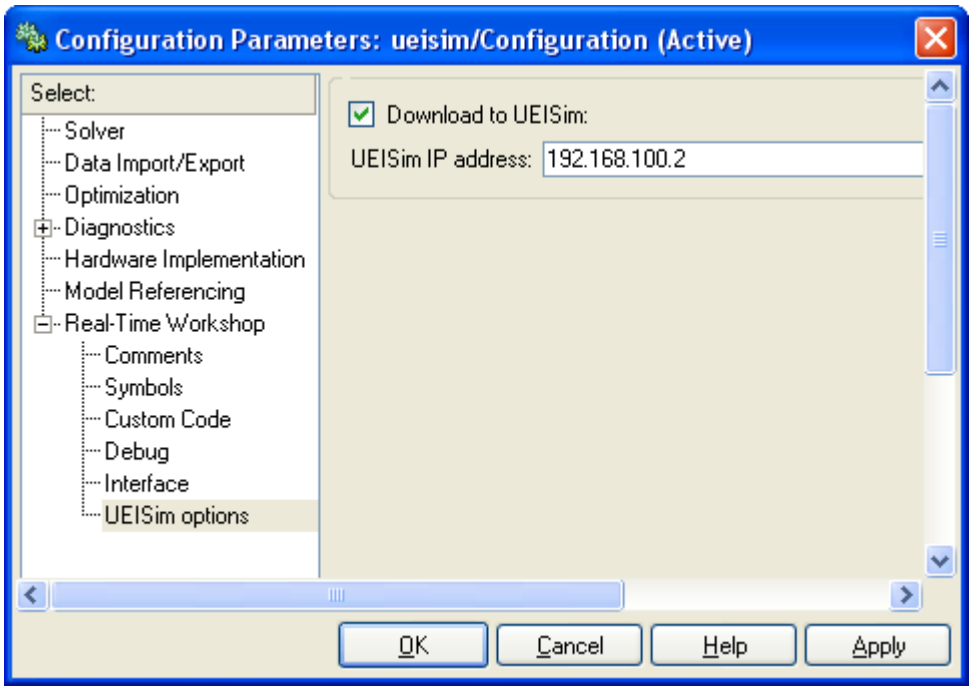
Click on the “Real-Time Workshop” option then on “Browse…” to change the system target file.



Select the UEISim target and click OK.

Click on “UEISim options”

United Electronic Industries
The High-Performance Alternative



- **Download to UEISim**: Check this option to automatically download the simulation executable to the UEISim.
- **UEISim IP address**: Enter the IP address of the UEISim.

Click on “Real-Time Workshop” again and then on “Build”. This will start the code generation and build process.

You should see an output similar to the following in MATLAB’s command window:

```
### Generating code into build directory: C:\test\ueisim_ueipac_rtw
### Invoking Target Language Compiler on ueisim.rtw
      tlc
      -r
      C:\test\ueisim.rtw
      e:\uei_svn\software\powerdna\3.3.x\UEIPAC\Simulink_rtw\ueisim.tlc
      -OC:\test\ueisim_ueipac_rtw
      -Ie:\uei_svn\software\powerdna\3.3.x\UEIPAC\Simulink_rtw
      -IC:\test\ueisim_ueipac_rtw\tlc
      -IC:\Program Files\MATLAB\R2007b\rtw\c\tlc\mw
      -IC:\Program Files\MATLAB\R2007b\rtw\c\tlc\lib
      -IC:\Program Files\MATLAB\R2007b\rtw\c\tlc\blocks
      -IC:\Program Files\MATLAB\R2007b\rtw\c\tlc\fixpt
      -IC:\Program Files\MATLAB\R2007b\stateflow\c\tlc
```


```
        -aEnforceIntegerDowncast=1
        -aFoldNonRolledExpr=1
        -aInlineInvariantSignals=0
        -aInlineParameters=0
        -aLocalBlockOutputs=1
        -aRollThreshold=5
        -aZeroInternalMemoryAtStartup=1
        -aZeroExternalMemoryAtStartup=1
        -aInitFltsAndDblsToZero=1
        -aGenerateReport=0
        -aGenCodeOnly=0
        -aRTWVerbose=1
        -aIncludeHyperlinkInReport=0
        -aLaunchReport=0
        -aGenerateTraceInfo=0
        -aForceParamTrailComments=0
        -aGenerateComments=1
        -aIgnoreCustomStorageClasses=1
        -aIncHierarchyInIds=0
        -aMaxRTWIdLen=31
        -aShowEliminatedStatements=0
        -aIncDataTypeInIds=0
        -aInsertBlockDesc=0
        -aSimulinkBlockComments=1
        -aInlinedPrmAccess="Literals"
        -aTargetFcnLib="ansi_tfl_table_tmw.mat"
        -aIsPILTarget=0
        -aLogVarNameModifier="rt_"
        -aGenerateFullHeader=1
        -aExtMode=0
        -aExtModeStaticAlloc=0
        -aExtModeTesting=0
        -aExtModeStaticAllocSize=1000000
        -aExtModeTransport=0
        -aRTWCAPISignals=0
        -aRTWCAPIParams=0
        -aGenerateASAP2=0
        -aDownloadToUEIPAC=1
        -aUEIPACIPAddress="192.168.15.200"
        -aGenerateTraceInfo=0
        -p10000


### Loading TLC function libraries

.....
### Initial pass through model to cache user defined code
.
### Caching model source code
.............................
```


```
### Writing header file ueisim_types.h
.
### Writing header file ueisim.h
### Writing source file ueisim.c
### Writing header file ueisim_private.h
.
### Writing header file rtmodel.h
### Writing source file ueisim_data.c
### Writing header file rt_nonfinite.h
### Writing source file rt_nonfinite.c
.
### TLC code generation complete.

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
### Evaluating PostCodeGenCommand specified in the model
Adding e:\uei_svn\software\powerdna\3326E1~1.X\UEIPAC\SIMULI~1 to
source and include paths
.
### Processing Template Makefile:
e:\uei_svn\software\powerdna\3.3.x\UEIPAC\Simulink_rtw\ueipac.tmf
### ueisim.mk which is generated from
e:\uei_svn\software\powerdna\3.3.x\UEIPAC\Simulink_rtw\ueipac.tmf is up
to date
### Building ueisim: .\ueisim.bat

<lots of compiler output>

Created executable: ueisim
Downloading ../ueisim to UEIPAC at 192.168.15.200
Downloaded: ueisim
>>
```

The simulation executable is now ready to be executed in the /tmp directory on the UEISim.
Log on the UEISim using the serial port console or Telnet and run the simulation:

```
/tmp # ./ueisim
StepSize: 0.010000 s
Model: 201 Option: 100
Model: 308 Option: 1
Model: 207 Option: 1
Model: 205 Option: 1
Model: 404 Option: 1

** starting the model **
** created ueisim.mat **
```

United Electronic Industries
The High-Performance Alternative

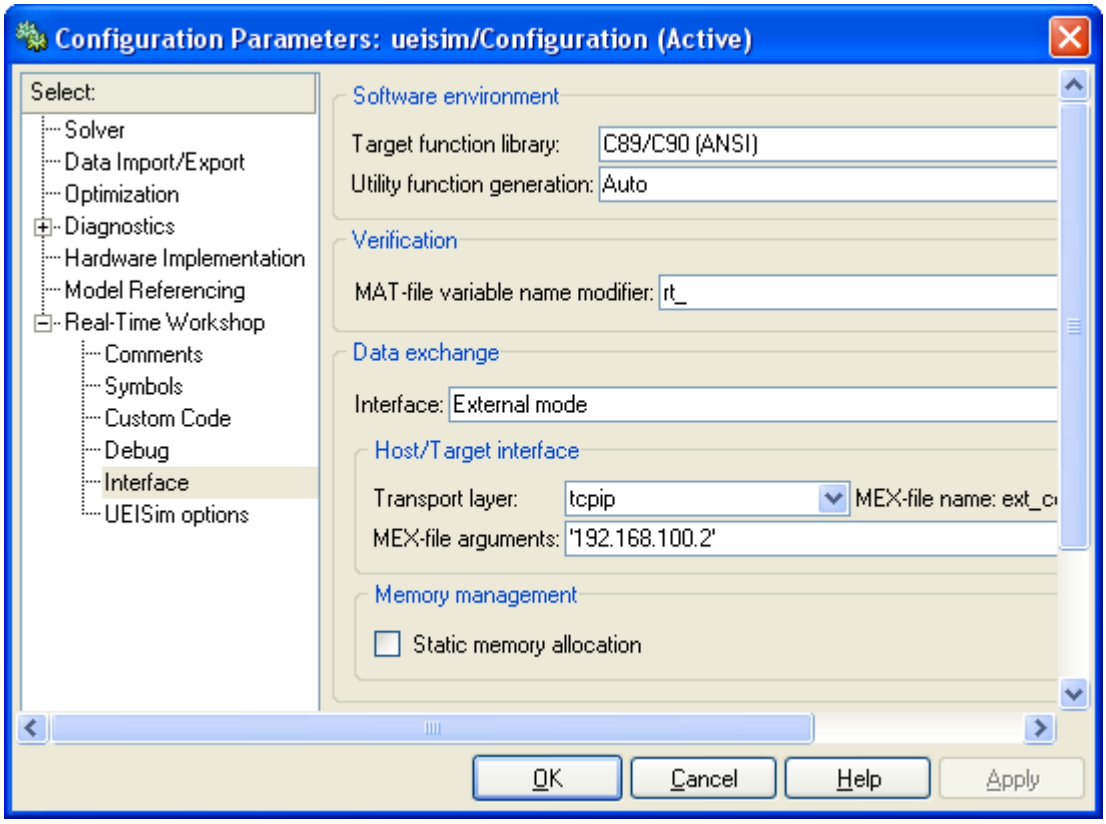
4.4. Connecting to UEISim in external mode

Simulink's external mode allows you to remotely monitor a simulation running on the UEISim.

Select the menu option "Simulation/Configuration Parameters…".

Click on the option "Real-Time Workshop" then on "Interface".

Change the interface to "External mode", set the "Transport layer" to "tcpip" and enter the IP address of the UEISim in the "MEX-file arguments" text field. This is a string argument and the IP address must be typed between quotes.



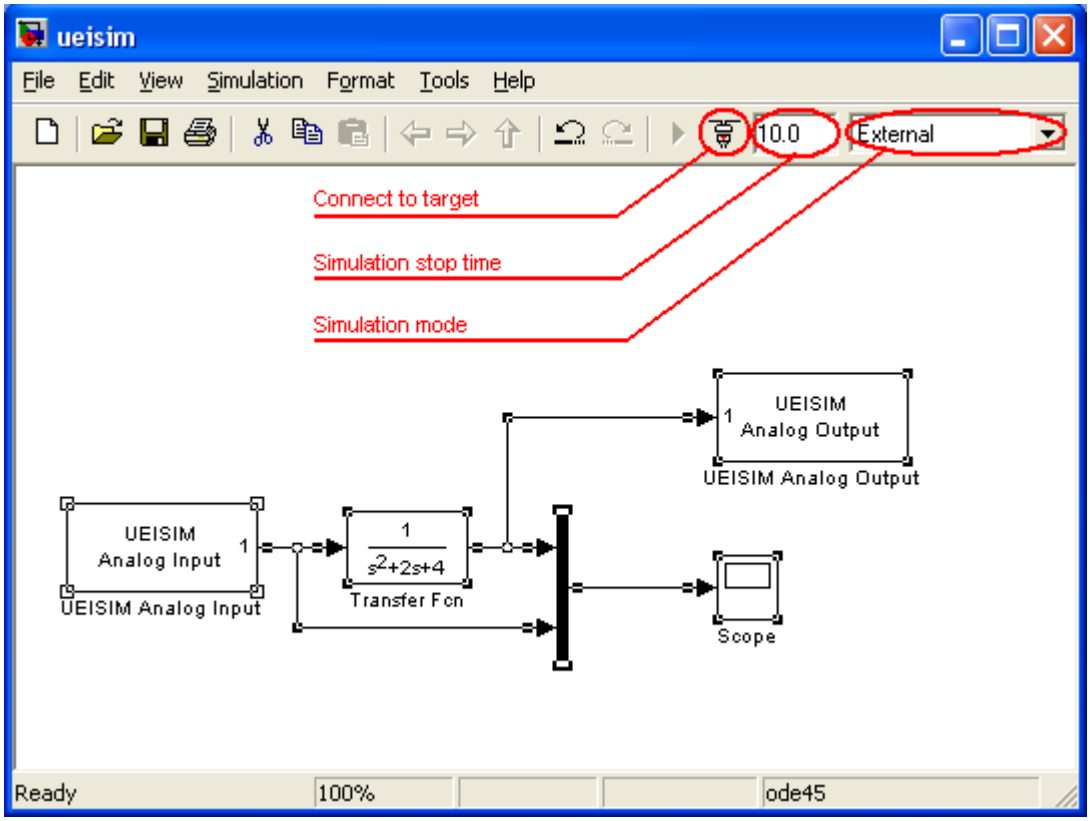
Click on OK and build the model again.

Logon the UEISim and start the simulation with the command line option '-w'.

```
/tmp # ./ueisim -w
```

United Electronic Industries
The High-Performance Alternative

This option tells the model to wait for commands received over the network before starting execution.



Set the Simulation stop-time to “inf” if you wish to run the simulation continuously.

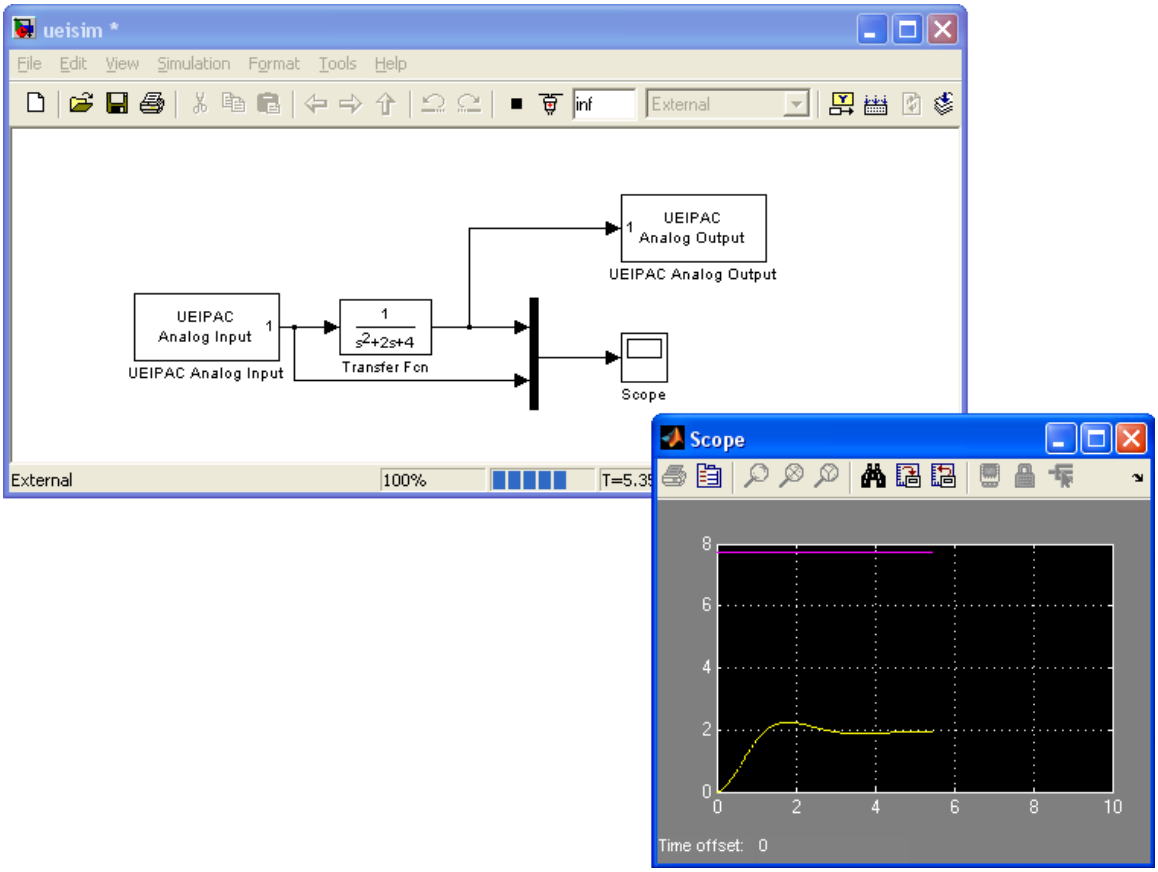
In your model window, change the “simulation mode” from “normal” to “external” using the toolbar combo-box.

Click on the “Connect to target” button.

After a few seconds, you will be notified that the connection is established when the “Start real-time code” button becomes enabled and the word “External” appears in the status bar.

Click on the “Start real-time code” button to start the simulation.
Double-click on the scope to view the acquired signal as well as the result of the transfer function.

United Electronic Industries
The High-Performance Alternative

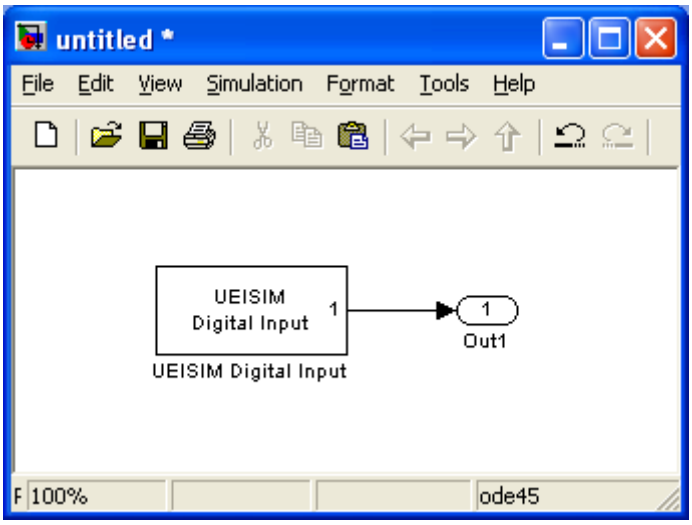


4.5. Logging Data to file

A Matlab MAT data file is automatically created when the model is executed on the UEISIM. By default it only contains one column of data representing the time of each step.

Use the “Out” block to add a column of data to the MAT file. The example below acquires digital inputs and writes them to the MAT file:

United Electronic Industries
The High-Performance Alternative



To look at the content of the MAT file, download the file from the UEISIM (using FTP or SCP) and open it with Matlab.

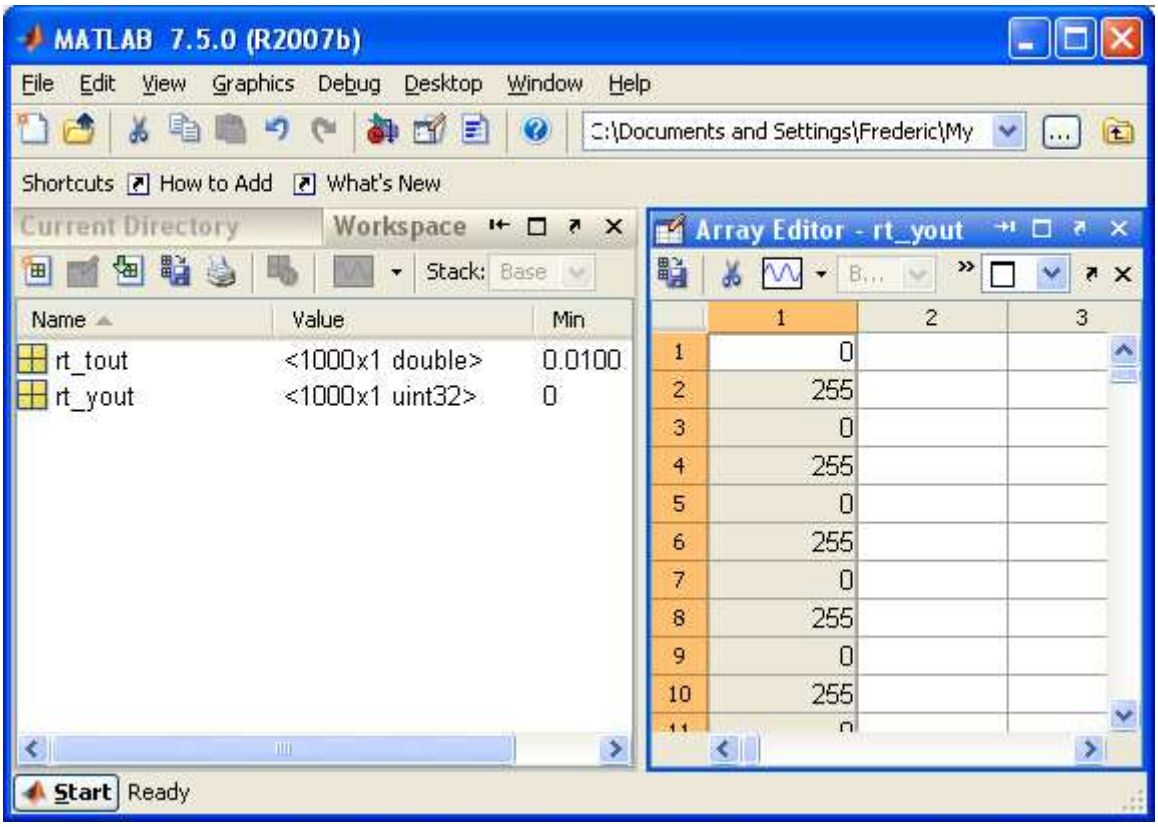
You can download the file directly from Matlab's command line with the following commands:

```
f=ftp('192.168.100.2','root', 'root')
cd(f,'tmp')
binary(f)
mget(f,'untitled.mat')
```

"rt_tout" is the time of each step
"rt_yout" is the data sent to the Out block.

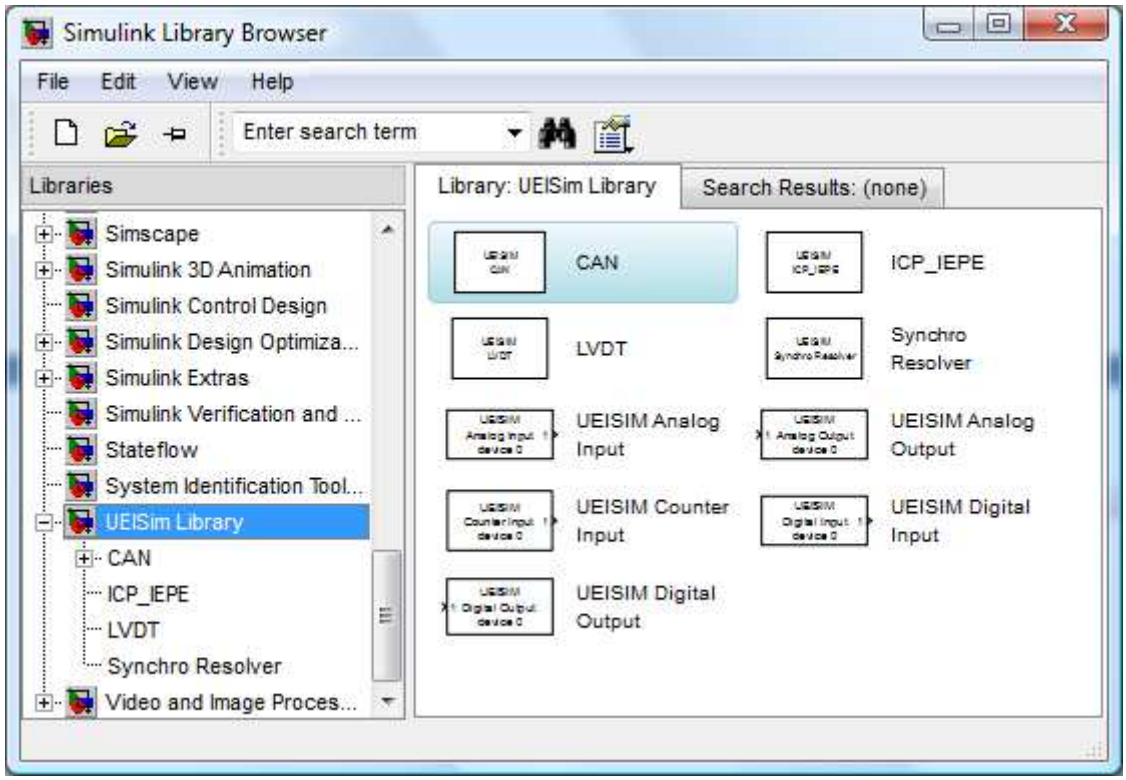
United Electronic Industries

The High-Performance Alternative



United Electronic Industries
The High-Performance Alternative

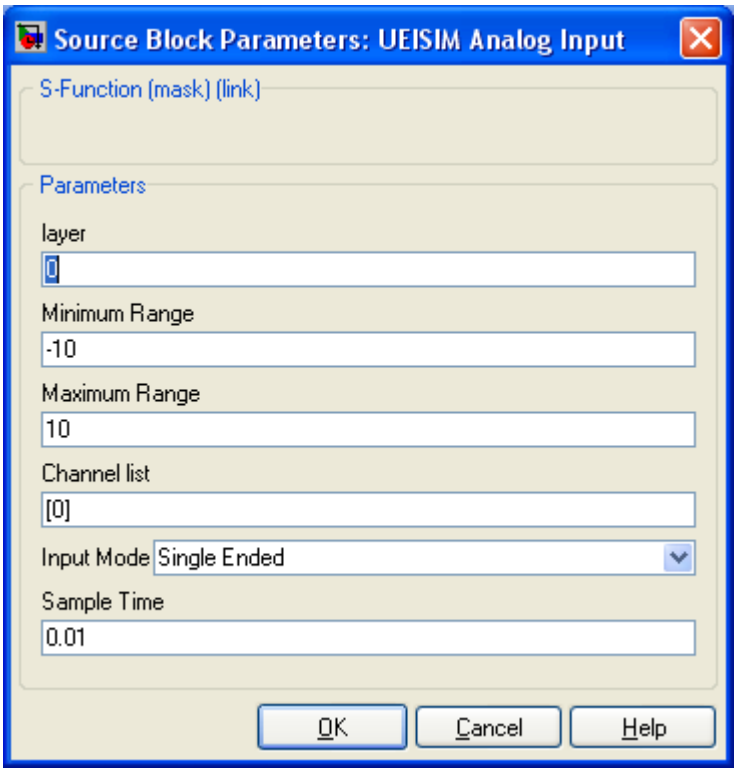
5. UEISIM Blockset



5.1. Analog Input block

The Analog Input block acquires data from the channels specified in the channel list.
Each channel measurement is available as a separate output.
The data type is double.

United Electronic Industries
The High-Performance Alternative

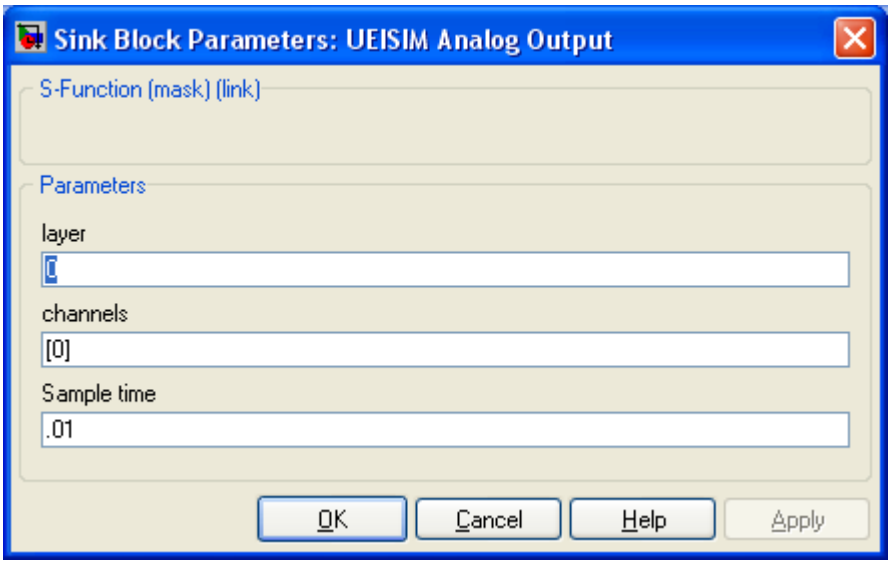


- **layer**: The Id of the analog input layer associated with this block. (layer Ids start at 0 with the top layer)
- **Minimum Range**: The minimum voltage expected at the input
- **Maximum Range**: The maximum voltage expected at the input
- **Channel list**: Array of channels to acquire from
- **Input Mode**: Single Ended or Differential
- **Sample Time**: The rate at which the block executes during simulation (it also sets the hardware ADC clock).

5.2. Analog Output

The Analog Output block updates the voltage generated by the channels specified in the channel list. Each channel update is specified as a separate input.
The data type is double.

United Electronic Industries
The High-Performance Alternative

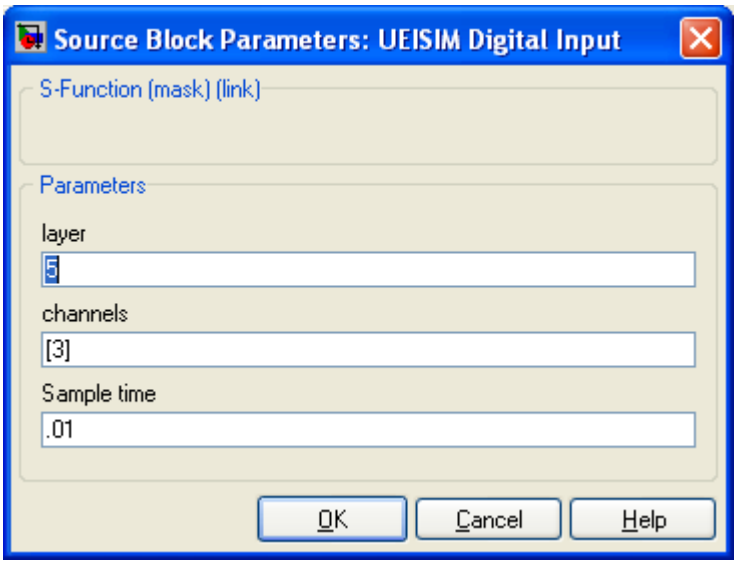


- **layer**: The Id of the analog output layer associated with this block (layer Ids start at 0 with the top layer)
- **Channels**: Array of channels to generate to
- **Sample Time**: The rate at which the block executes during simulation (it also sets the hardware DAC clock).

5.3. Digital Input

The Digital Input block acquires the digital state of the channels specified in the channel list. Each channel is available as a separate output.
A channel is a group of input lines. The number of input lines contained in each channel depends on the hardware (for example the DIO-405 groups its input lines in one port of twelve lines).
The data type is uint32. Each bit of the value read from a given channel corresponds to the state of one input line.

United Electronic Industries
The High-Performance Alternative

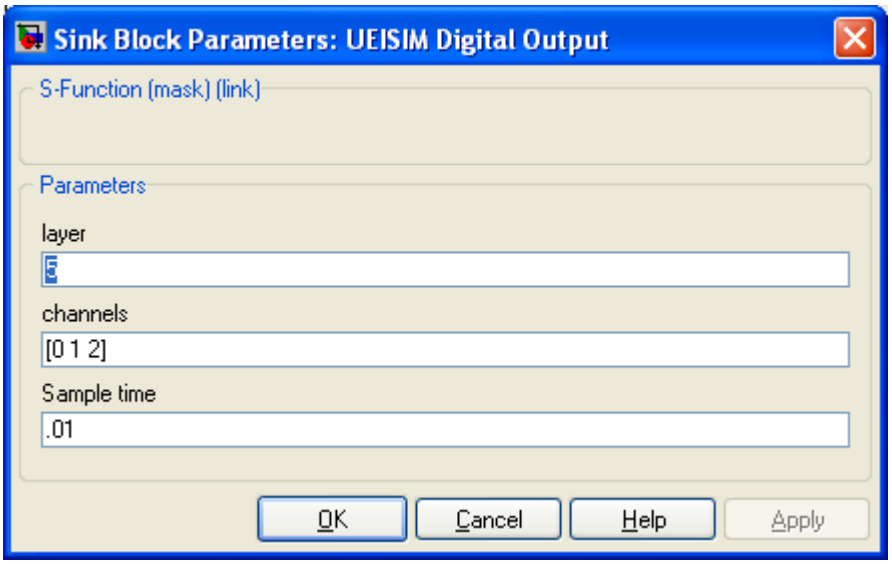


- **layer**: The Id of the digital input layer associated with this block (layer Ids start at 0 with the top layer)
- **Channels**: Array of ports to read from. Input lines are organized into ports (read the manual of your digital layer to find out how many lines there are in each port).
- **Sample Time**: The rate at which the block executes during simulation (it also sets the hardware clock).

5.4. Digital Output

The Digital Output block updates the digital state of the channels specified in the channel list. Each channel is available as a separate input.
A channel is a group of output lines. The number of output lines contained in each channel depends on the hardware (for example the DIO-405 groups its output lines in one port of twelve lines).
The data type is uint32. Each bit of the value written to a given channel corresponds to the state of one output line.

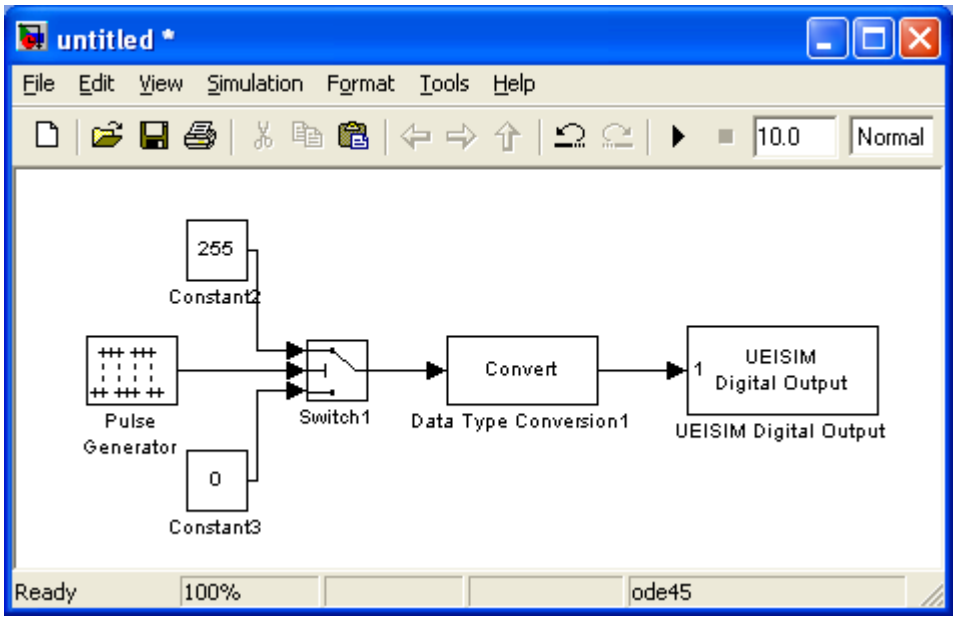
United Electronic Industries
The High-Performance Alternative



- **layer**: The Id of the digital output layer associated with this block (layer Ids start at 0 with the top layer)
- **Channels**: Array of ports to write to. Input lines are organized into ports (read the manual of your digital layer to find out how many lines there are in each port).
- **Sample Time**: The rate at which the block executes during simulation (it also sets the hardware clock).

The type of the signals connected to the DI block must be “uint32”. You can use Simulink’s “Data Type Conversion block” to convert your signal as shown in the example below:

United Electronic Industries
The High-Performance Alternative



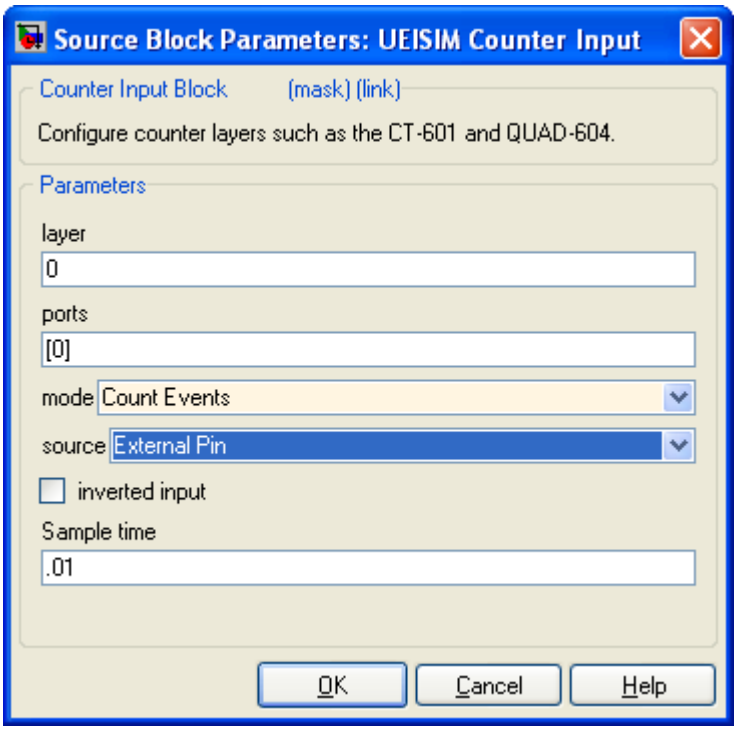
5.5. Counter Input

The Counter Input block acquires the current count of the counters specified in the channel list. Each counter is available as a separate output.
The data type is uint32.

The value read depends on the counter operting mode:

- Count Events: Reads the number of rising edges detected on the counter input since the model started
- Pulse Width: The delay between the last rising and falling edges detected on the counter input. Delay is returned in 66MHz clock ticks; divide the value by 66000000.0 to convert to seconds.
- Period: The delay between the two last rising edges detected on the counter input. Delay is returned in 66MHz clock ticks; divide the value by 66000000.0 to convert to seconds.
- Quadrature: Reads the position measured by a quadrature encoder.

United Electronic Industries
The High-Performance Alternative



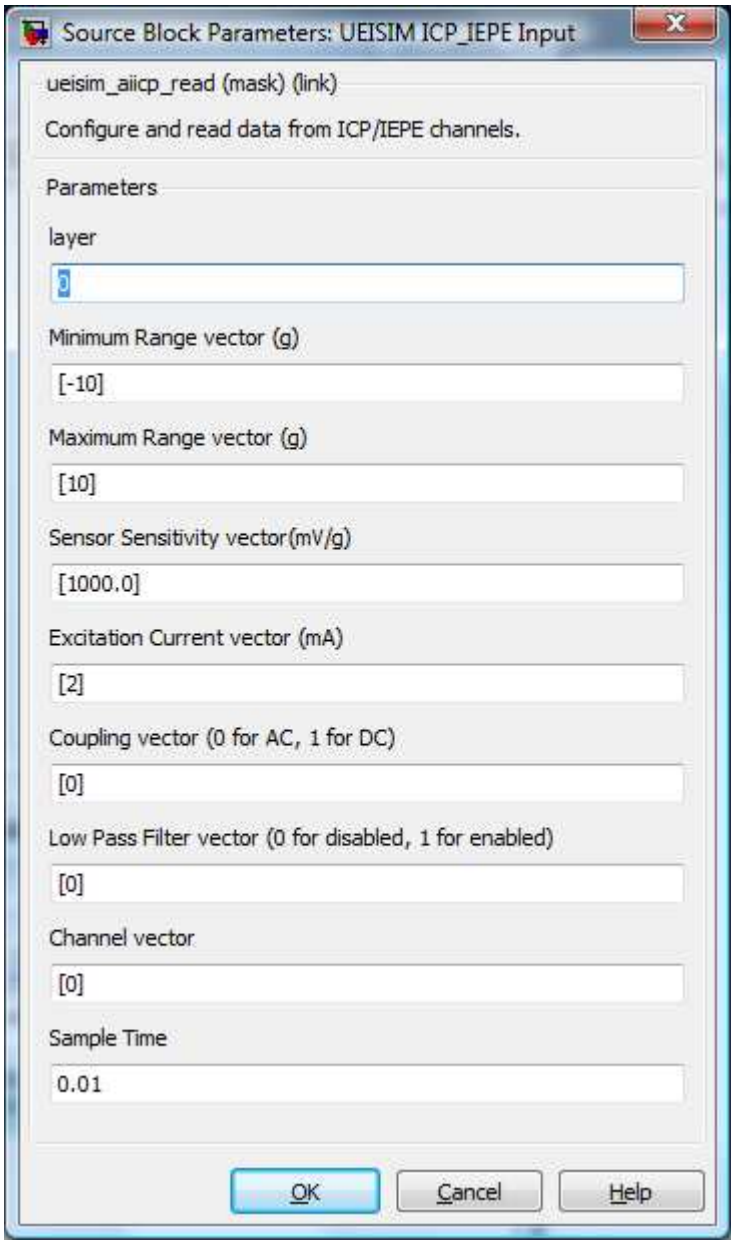
- **layer**: The Id of the digital output layer associated with this block (layer Ids start at 0 with the top layer)
- **ports**: Array of ports to read from.
- **mode**: The operation mode. Possible values are “Count Events”, “Measure Pulse width”, “Measure period” and “Quadrature Encoder”.
- **source**: The source of the input signal. Possible values are “Internal Clock” and “External Pin”.
- **inverted input**: the input signal is inverted when this is checked.
- **Sample Time**: The rate at which the block executes during simulation (it also sets the hardware clock).

5.6. ICP/IEPE sensors

Use the ICP/IEPE block to acquire data from ICP or IEPE sensors. Those sensors are only supported by analog input hardware that can provide excitation current to power the sensors (for example the AI-211).

The data type of the value returned for each configured channel is double.

United Electronic Industries
The High-Performance Alternative



- **layer**: The Id of the analog input layer associated with this block. (layer Ids start at 0 with the top/left layer)
- **Minimum Range vector**: The minimum value expected at the input of each channel
- **Maximum Range vector**: The maximum value expected at the input of each channel

United Electronic Industries
The High-Performance Alternative

- **Sensor Sensitivity vector**: The sensitivity of the sensor(s) connected to each channel
- **Excitation Current vector**: The excitation current used to power sensor(s) connected to each channel
- **Coupling vector**: The coupling (AC or DC) used on each channel
- **Low Pass Filter vector**: Turns on or off the anti-aliasing low pass filter on each channel
- **Channel vector**: Array of channels to acquire from
- **Sample Time**: The rate at which the block executes during simulation (it also sets the hardware ADC clock).

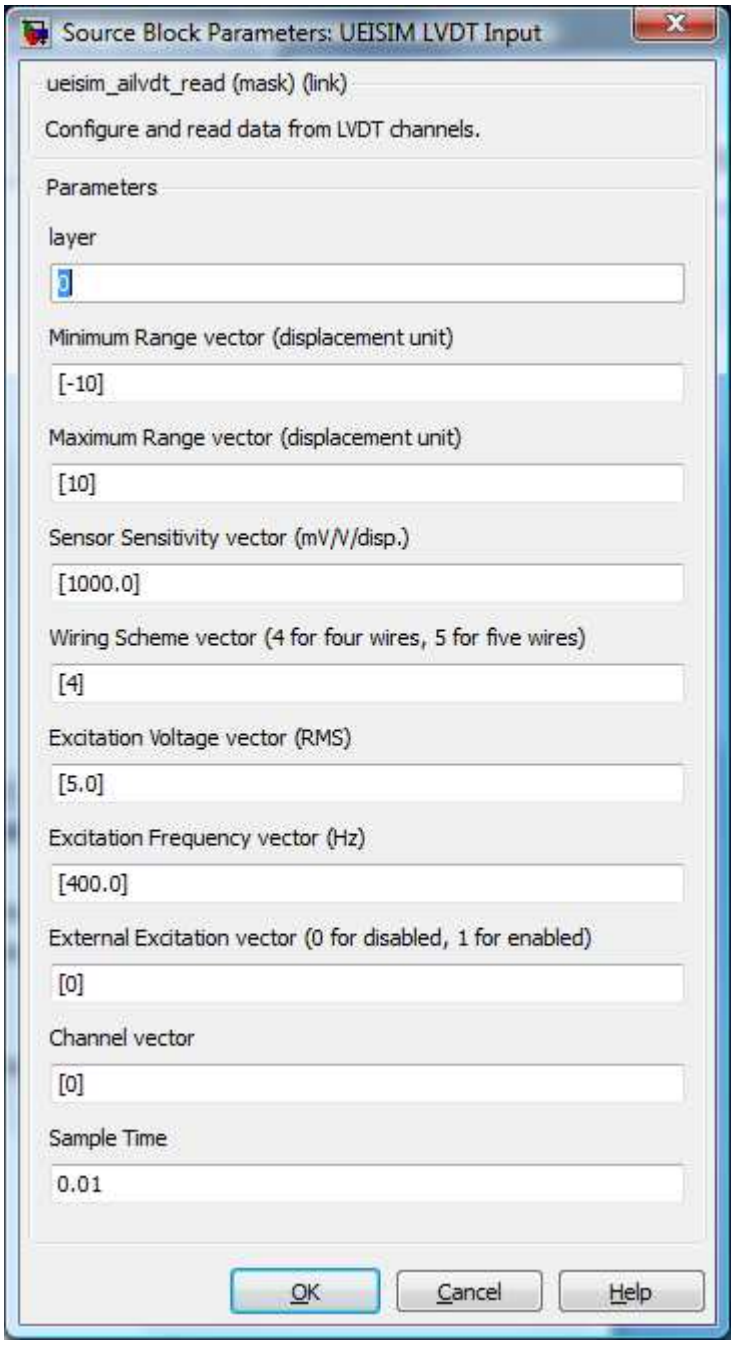
5.7. LVDT

Use the LVDT blocks to acquire data from LVDT sensors and also simulate voltage emitted by real LVDT sensors.
Those sensors are only supported by analog input hardware that can provide excitation current to power the LVDTs (for example the AI-254).

5.7.1. LVDT Input

The data type of the value returned for each configured channel is double

United Electronic Industries
The High-Performance Alternative



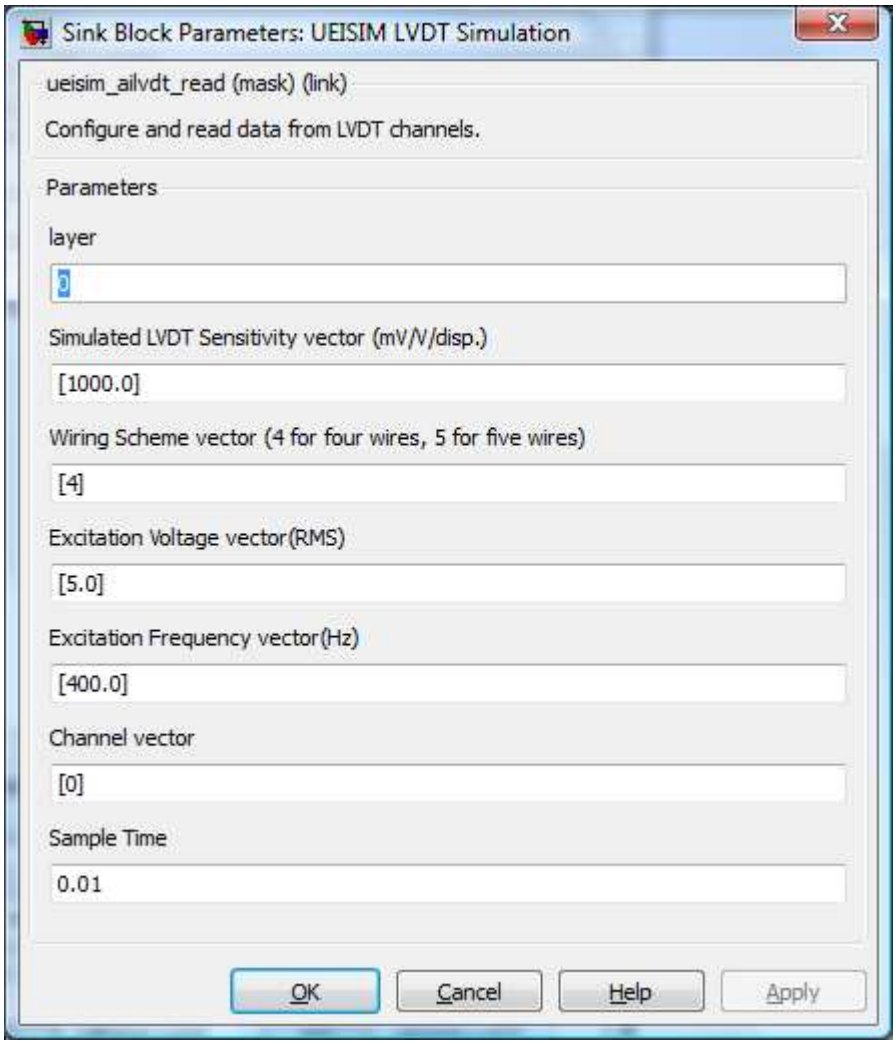
- **layer**: The Id of the analog input layer associated with this block. (layer Ids start at 0 with the top/left layer)

- **Minimum Range vector**: The minimum value expected at the input of each channel
- **Maximum Range vector**: The maximum value expected at the input of each channel
- **Sensor Sensitivity vector**: The sensitivity of the LVDT(s) connected to each channel
- **Wiring Scheme vector**: The wiring scheme (4 or 5 wires) used to connect LVDT(s) to each channel
- **Excitation Voltage vector**: The excitation voltage used to power LVDT(s) connected to each channel
- **Excitation Frequency vector**: The excitation frequency used to power LVDT(s) connected to each channel
- **External Excitation vector**: Specifies whether channel(s) provide excitation to LVDT(s) or whether excitation is supplied externally
- **Channel vector**: Array of channels to acquire from
- **Sample Time**: The rate at which the block executes during simulation (it also sets the hardware ADC clock).

5.7.2. LVDT Simulation

The data type of the value written to each configured channel is double

United Electronic Industries
The High-Performance Alternative



- **layer**: The Id of the analog input layer associated with this block. (layer Ids start at 0 with the top/left layer)
- **Simulated LVDT Sensitivity vector**: The sensitivity of the LVDT(s) simulated by each channel
- **Wiring Scheme vector**: The wiring scheme (4 or 5 wires) used to connect the LVDT(s) simulated by each channel
- **Excitation Voltage vector**: The excitation voltage used to power LVDT(s) simulated by each channel
- **Excitation Frequency vector**: The excitation frequency used to power LVDT(s) simulated by each channel
- **Channel vector**: Array of channels to simulate from

United Electronic Industries
The High-Performance Alternative

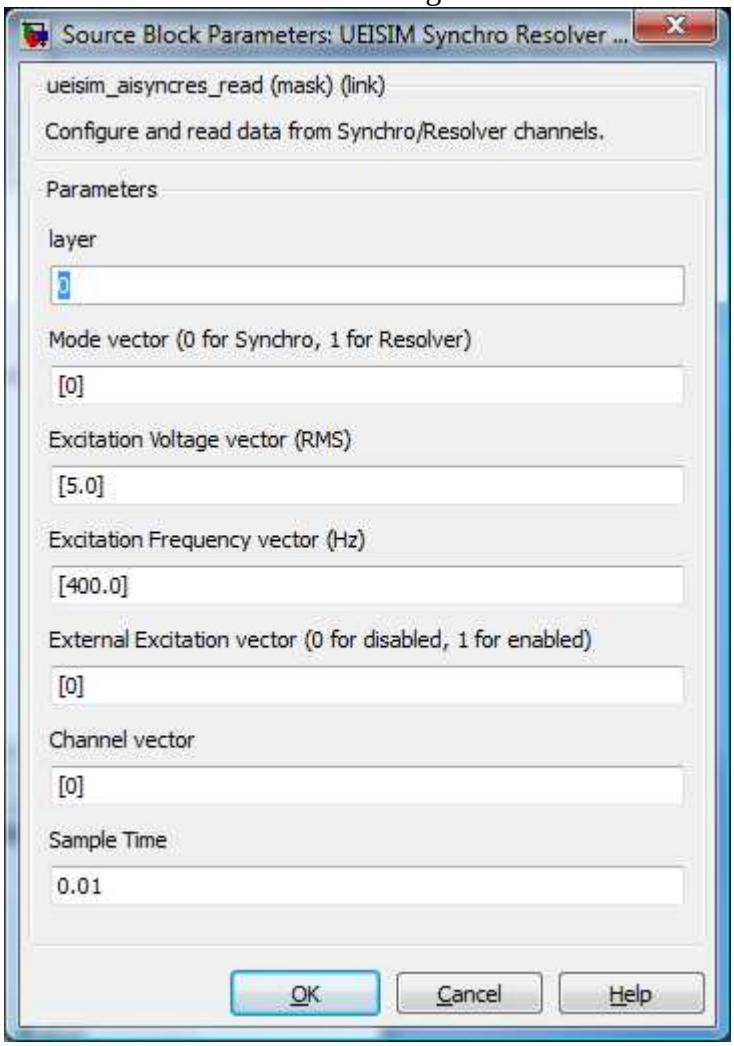
- **Sample Time**: The rate at which the block executes.

5.8. Synchro/Resolver

Use the Synchro/Resolver blocks to acquire data from Synchros or Resolvers and also simulate voltage emitted by real Synchros or Resolvers.
Those sensors are only supported by analog input hardware that can provide excitation current to power the Synchro/Resolvers (for example the AI-255).

5.8.1. Synchro/Resolver Input

The data type of the value returned for each configured channel is double

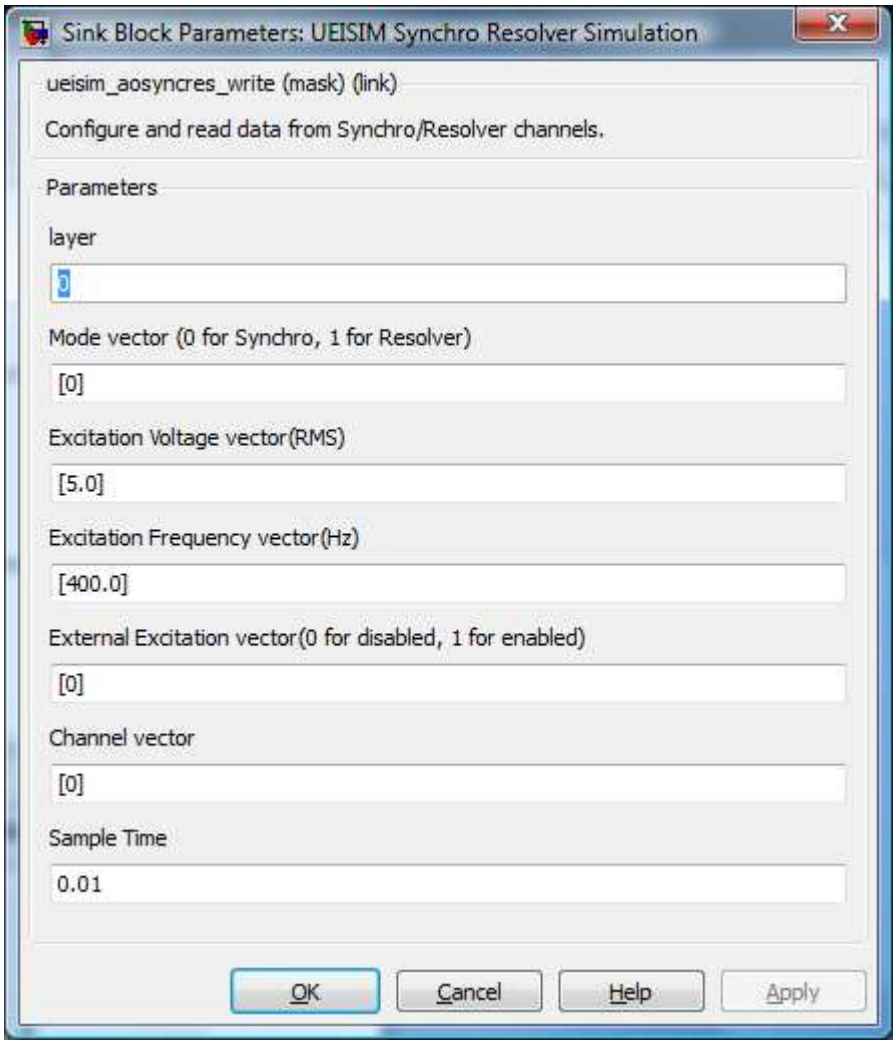


- **layer**: The Id of the analog input layer associated with this block. (layer Ids start at 0 with the top/left layer)
- **Mode vector**: Specifies whether a Synchro or a Resolver is connected to each channel
- **Excitation Voltage vector**: The excitation voltage used to power Synchro/Resolvers(s) connected to each channel
- **Excitation Frequency vector**: The excitation frequency used to power Synchro/Resolver(s) connected to each channel
- **External Excitation vector**: Specifies whether channel(s) provide excitation to Synchro/Resolver(s) or whether excitation is supplied externally
- **Channel vector**: Array of channels to acquire from
- **Sample Time**: The rate at which the block executes during simulation (it also sets the hardware ADC clock).

5.8.2. Synchro/Resolver Output

The data type of the value written to each configured channel is double

United Electronic Industries
The High-Performance Alternative



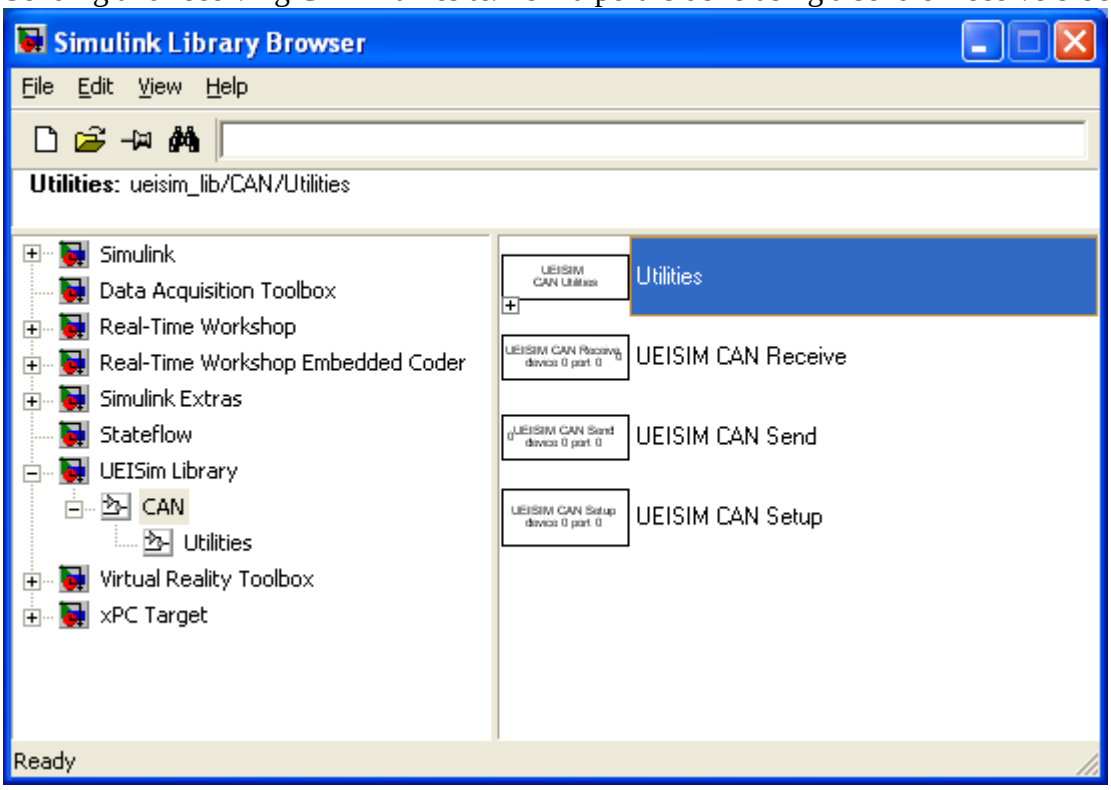
- **layer**: The Id of the analog input layer associated with this block. (layer Ids start at 0 with the top/left layer)
- **Mode vector**: Specifies whether each channel is simulating a Synchro or a Resolver
- **Excitation Voltage vector**: The excitation voltage used to power Synchro/Resolver(s) simulated by each channel
- **Excitation Frequency vector**: The excitation frequency used to power Synchro/Resolver(s) simulated by each channel
- **External Excitation vector**: Specifies whether channel(s) provide excitation to Synchro/Resolver(s) or whether excitation is supplied externally
- **Channel vector**: Array of channels to simulate from

United Electronic Industries
The High-Performance Alternative

- **Sample Time**: The rate at which the block executes

5.9. CAN bus communication

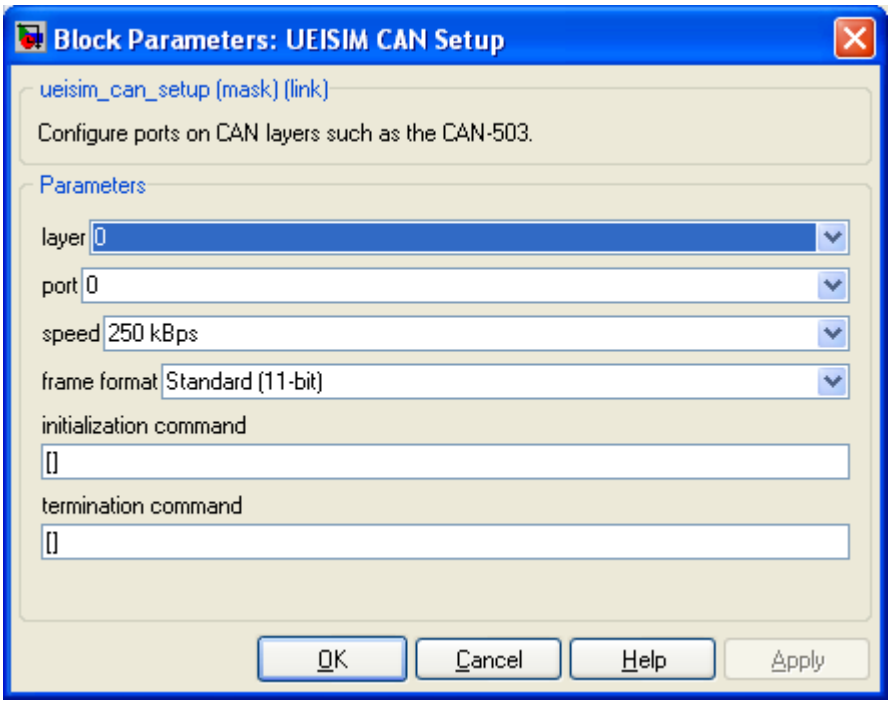
CAN communication blocks give access to the CAN-503 CAN ports. The configuration of each port is done using an independent setup block.
Sending and receiving CAN frames to/from a port is done using a send or receive block.



5.9.1. CAN Setup block

Configure communication settings on a given CAN port.
There must be one setup block for each port used in the model.

United Electronic Industries
The High-Performance Alternative



- **layer**: The Id of the CAN layer associated with this block (layer Ids start at 0 with the top layer)
- **port**: The Id of the port to configure (port Ids start at 0)
- **speed**: The speed in bits/s used on the CAN bus connected to this port
- **frame format**: The type of frame sent or received (Standard or Extended)
- **initialization command**: A sequence of frames to send to the CAN bus right before the model start.
- **Termination command**: A sequence of frames to send to the CAN bus right before the model terminates.

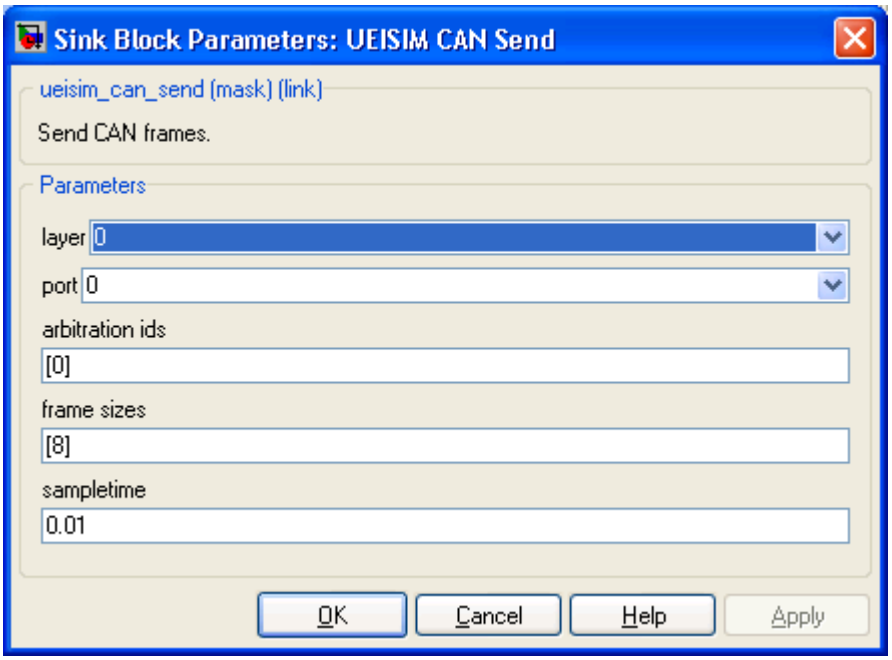
The initialization and termination sequences use the following format [id1 len1 dataMSB1 dataLSB1 id2 len2 dataMSB2 dataLSB2 …]. For example to send a CAN frame with ID 0x12 and 5 bytes of data (0x01 0z02 0x03 0x04 0x05) use the following:

```
[ hex2dec('12') 5 hex2dec('05') hex2dec('04030201')]
```

5.9.2. CAN Send block

Send a group of CAN frames to one CAN port. You can create multiple instance of this block to send multiple groups of frames at different rate.

United Electronic Industries
The High-Performance Alternative



- **layer**: The Id of the CAN layer associated with this block (layer Ids start at 0 with the top layer)
- **port**: The Id of the port to send to (port Ids start at 0)
- **arbitration ids**: A list of arbitration IDs to send
- **frame sizes**: The size of the data payload for each frame
- **sample time**: The rate at which the block executes during simulation

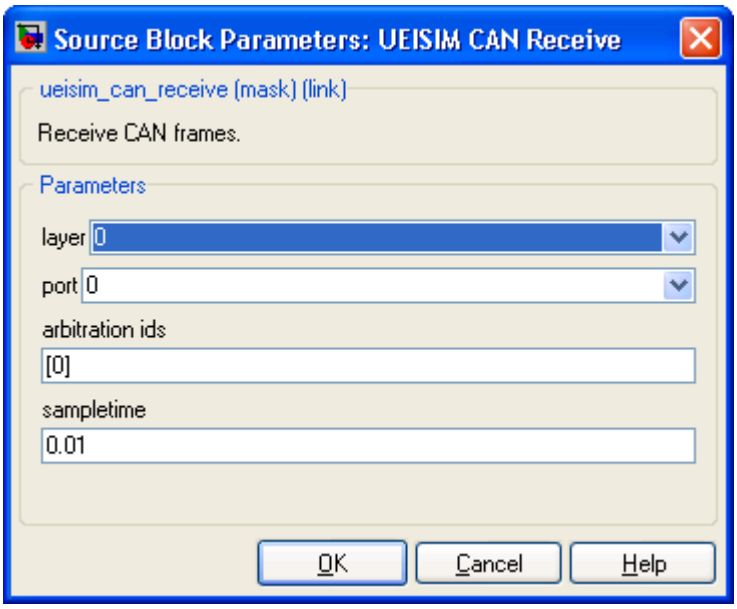
The block displays an input port for connecting the value of the data payload for each frame.
The data payload is specified using the double data type, which is big enough to carry the 64 bits required for a full payload (8 bytes maximum).

Refer to section about packing/unpacking data into payload below.

5.9.3. CAN Receive block

Receive a group of CAN frames from one CAN port. You can create multiple instance of this block to receive multiple group of frames at different rate.

United Electronic Industries
The High-Performance Alternative



- **layer**: The Id of the CAN layer associated with this block (layer Ids start at 0 with the top layer)
- **port**: The Id of the port to receive from (port Ids start at 0)
- **arbitration ids**: A list of arbitration IDs to receive
- **sample time**: The rate at which the block executes during simulation

The block outputs the value of the data payload of each frame.
The data payload is specified using the double data type which is big enough to carry the 64 bits required for a full payload (8 bytes maximum).

Refer to section about packing/unpacking data into payload below.

5.9.4. Utility blocks

Utility blocks are used to pack and unpack data stored in the payload of CAN frames that are sent or received. You can specify the data types and position of multiple signals within a single CAN frame.

Each signal is specified using four parameters:

- data type: the type of the signal, possible values are boolean, int8, uint8, int16, uint16, int32, uint32, single or double.
- endianness: the endianness of the signal, possible values are intel or motorola.
- start bit: the position of the first bit of the signal in the 8 bytes data payload of the CAN frame.

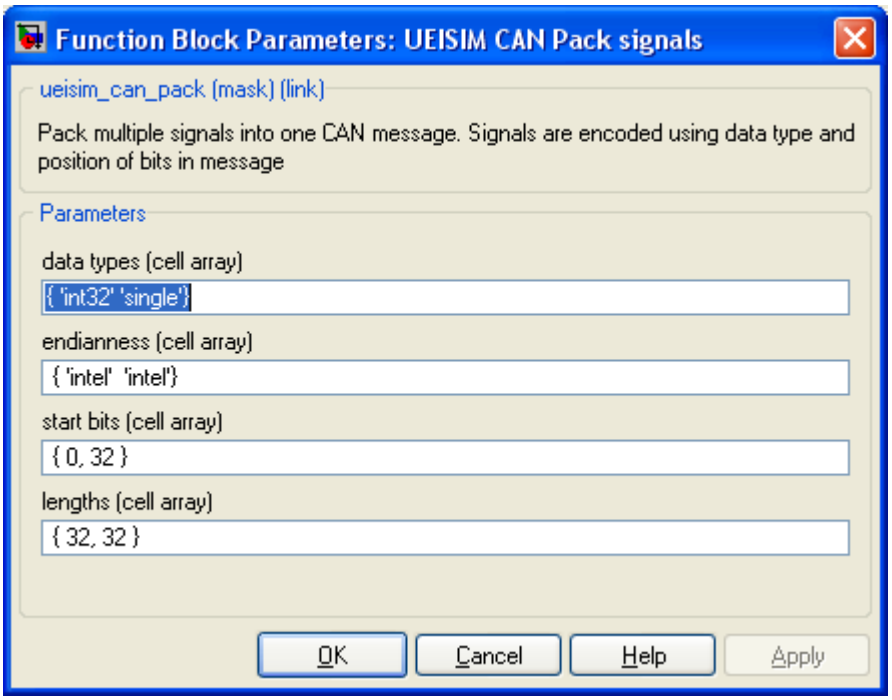
United Electronic Industries
The High-Performance Alternative

- bit length: the number of bits used to represent the signal in the 8 bytes data payload.

For example you could specify that a CAN frame contains a 16-bit integer starting at bit 0, another 16-bit integer coded in big endian format starting at bit 16 and a single precision floating-point starting at bit 32.

5.9.4.1. CAN pack block

Pack multiple signals into one CAN message. Signals are encoded using data type and position of bits in message.

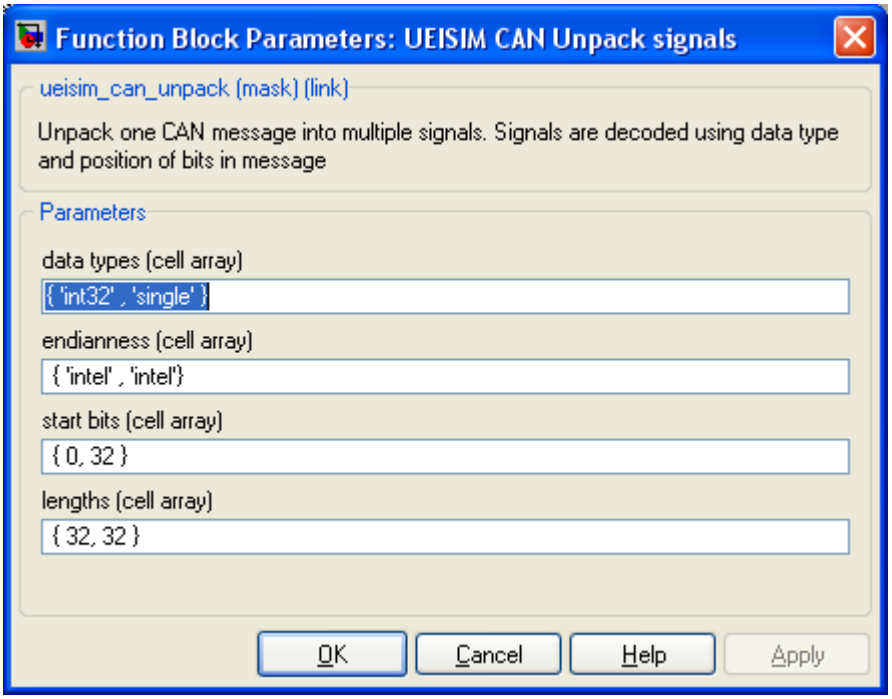


- **Data types**: A cell array containing the data types of the signals to pack in the message
- **Endianness**: A cell array containing the endianness of the signals to pack
- **Start bits**: A cell array containing the index of the first bit of the signals to pack
- **Bit length**: A cell array containing the number of bits of the signals to pack

The block displays an input port for each signal and outputs one double value containing the packed signals. The output value is ready to be connected to the CAN Send block.

5.9.4.2. CAN unpack block

Unpack one CAN message into multiple signals. Signals are decoded using data type and position of bits in message



- **Data types**: A cell array containing the data types of the signals to unpack from the message
- **Endianness**: A cell array containing the endianness of the signals to unpack
- **Start bits**: A cell array containing the index of the first bit of the signals to unpack
- **Bit length**: A cell array containing the number of bits of the signals to unpack

The block displays one input port to connect a double value coming from the CAN Receive block. It also displays an output port for each signal to unpack from the CAN message.

5.9.5. CAN examples

The following example configures two ports on the same CAN-503, send frames with Ids 102 and 258 out of port 0 and receives frames with Ids 102 and 258 from port 1.

United Electronic Industries
The High-Performance Alternative

If port 0 and port1 are connected to the same CAN bus, you will receive what you send.