LabVIEW Wireless Sensor Network Module

2022-09-29

Contents

Wireless Sensor Network Module. 4

- Getting Started. 4
- Related Documentation. 4
- Creating Projects. 5
  - Configuring a Project with Offline Hardware. 5
  - Configuring a Project with Connected Hardware. 6
  - Comparing a Deployed Node to One on the Host. 6
- Creating Targets. 7
- Using LabVIEW WSN Targets on WSN Nodes. 8
  - Limitations in Developing Block Diagrams. 9
  - Performance and Memory Considerations. 18
  - User Memory. 19
  - Using Elemental I/O Nodes. 19
  - Using Elemental I/O Property Nodes. 21
  - Adding and Removing Header Files from an Inline C Node. 41
- Communicating with WSN Nodes. 42
  - Sending and Receiving Messages. 43
- Reading and Writing Data. 43
  - Supported User-Defined I/O Variables. 46
- Accessing User RAM. 47
- Building VIs into Applications. 48
- Deploying Applications to WSN Nodes. 49
- Debugging Applications. 49
  - LabVIEW WSN Module Error Codes. 50
- Wireless Sensor Network Module VIs. 53
  - Elemental I/O Functions. 53
  - WSN Host API VIs. 56
  - WSN VIs. 66
- NI-WSN. 138
  - NI-WSN Related Documentation. 138
  - Searching PDF Versions of WSN Manuals. 139
  - Supported WSN Hardware. 140
  - Configuring Projects. 181

I/O Variables. 183
WSN Gateway Settings. 185
Reading from WSN Channels. 197
Writing to WSN Channels. 198
WSN Error Codes. 199

Wireless Sensor Network Module

June 2013, 372802E-01

Use the Wireless Sensor Network Module to create and download LabVIEW applications to NI WSN nodes over a wireless connection. By programming WSN nodes, you can customize the node's behavior to increase acquisition performance, interface directly with sensors, and extend battery life.

The **LabVIEW Help** uses (WSN) to indicate LabVIEW WSN-specific help topics in the topic title and index.

Getting Started (WSN)

Use the following documents to get started with the Wireless Sensor Network Module:

- Refer to the KnowledgeBase at `ni.com` for a WSN tutorial and the latest information about the Wireless Sensor Network Module.
- Refer to the KnowledgeBase at `ni.com` for more information about the WSN execution model.

Related Documentation (WSN)

The following documents contain information that you might find helpful as you use the Wireless Sensor Network Module.

- **Wireless Sensor Network Module Readme**—Use this file to learn important last-minute information about Wireless Sensor Network Module. Open the readme file by selecting **Start»All Programs»National Instruments»LabVIEW»Readme** and opening `readme_WSN.html`.
- Wireless Sensor Network Module Examples—Use the Wireless Sensor Network Module examples as a starting point for developing WSN VIs and applications. You can modify an example to fit an application, or you can copy

and paste from one or more examples into a VI that you create. Use the NI Example Finder, available by selecting **Help»Find Examples**, to browse or search the example VIs. You also can browse the examples by navigating to the `labview\examples\lvemb\WSN` directory.

- **NI Wireless Sensor Network Devices Getting Started Guide**
- **NI WSN-32xx User Guide and Specifications**
- User guide and specifications for the WSN gateway
- **Measurement & Automation Explorer Help for WSN**, which you can access from the National Instruments Measurement & Automation Explorer (MAX).
- Additional LabVIEW documentation.

Refer to the National Instruments Product Manuals Library for updated documentation resources.

Creating Projects (WSN)

You must create a LabVIEW project before you can build a VI into an application. Projects contain targets, VIs, and build specifications.

Use the Project Explorer window to create an empty project. You can add VIs, .lib files, and other items to the project.

Configuring a Project with Offline Hardware

Complete the following steps to configure the project if you do not have hardware installed.

1. Create a new project or open an existing project.
2. Right-click the project root in the **Project Explorer** window and select **New»Targets and Devices** from the shortcut menu to display the **Add Targets and Devices** dialog box.
3. Click the **New target or device** radio button, select the gateway, and click **OK**. LabVIEW adds a target item to the project.

4. Right-click the gateway in the **Project Explorer** window and select **New»Targets and Devices** from the shortcut menu to display the **Add Targets and Devices** dialog box.
5. Click the **New target or device** radio button, select the node to add, and click **OK**.
6. Assign an ID to the node, and click **OK**.
7. Right-click the node in the **Project Explorer** window and select **Properties** to configure node settings. Click the **Help** button for information about the node settings.

Configuring a Project with Connected Hardware

Complete the following steps to configure the project. The gateway must be powered on, connected to the same subnet as the host computer, and configured in MAX. Refer to the device documentation for installation and configuration information.

1. Create a new project or open an existing project.
2. Right-click the project root in the **Project Explorer** window and select **New»Targets and Devices** from the shortcut menu to display the **Add Targets and Devices** dialog box.
3. Select either **Discover an existing target(s) or device(s)** or **Specify a target or device by IP address**.
4. Select the gateway and click **OK**. It may take several seconds to update the target configuration.
5. Right-click the node in the **Project Explorer** window and select **Properties** to configure node settings. Click the **Help** button for information about the node settings.

Comparing a Deployed Node Configuration to One on the Host

Right-click the gateway in the **Project Explorer** window, and select **Utilities»Compare Project & System**.

Creating Targets (WSN)

When you create a project for an application through the Project Explorer window, you must create a WSN target and add the target to the project. You must create a target for each WSN node on which you plan to run an application. Complete the following steps to add a WSN target to the WSN node.

1. Create a project.
2. In the **Project Explorer** window, right-click the project root and select **New»Targets and Devices** from the shortcut menu to display the Add Targets and Devices dialog box.
3. Select **Existing target or device** to display the available targets and devices.
4. Expand the **WSN Gateway** folder. LabVIEW displays the WSN gateways that you configured in the Measurement & Automation Explorer (MAX).
5. Select a gateway from the list of available gateways.
6. Click the **OK** button. The gateway appears in the **Project Explorer** window.

   Note You must set a time server for the WSN gateway to obtain significant time information in the application. Refer to the **Measurement & Automation Explorer Help for WSN**, available by selecting **Start»All Programs»National Instruments»NI-WSN»Configuring WSN in MAX**, for more information about configuring WSN gateways.

7. Right-click the gateway in the **Project Explorer** window and select **New»Targets and Devices** from the shortcut menu to display the Add Targets and Devices dialog box.
8. Select **Existing target or device** to display the available targets and devices.

9. Expand the **WSN Node** folder. LabVIEW displays the WSN nodes that you configured in MAX.

   Note Refer to the **Measurement & Automation Explorer Help for WSN**, available by selecting **Start»All Programs»National Instruments»NI-WSN»Configuring WSN in MAX**, for more information about configuring WSN nodes.

10. Select a WSN node from the list of available WSN nodes.
11. Click the **OK** button. The WSN node appears under the gateway in the **Project Explorer** window.
12. Right-click the WSN node in the **Project Explorer** window and select **Add LabVIEW WSN Target** from the shortcut menu to add a WSN target to the WSN node.

Using LabVIEW WSN Targets on WSN Nodes (WSN)

When you add a WSN target to a WSN node, LabVIEW creates a top-level VI that you can use to acquire data from and communicate with a WSN node. By default, this VI includes a Case structure with cases for the following states.

- **Start**—Occurs when the application begins to run on the WSN node.
- **Sample**—Occurs when data is read from the WSN node or data is sent to the WSN gateway. This state occurs based on the sampling rate for the WSN node. Use this state to process the data before it is sent to the gateway.
- **Receive**—Occurs when the WSN node receives a user message from the host via the gateway.
- **Network Status Change**—Occurs when the network status of the WSN node changes. Use this state to detect if the WSN node is connected to or disconnected from the WSN gateway and to configure how sampling of data occurs based on the connection state of the WSN node. For example, you can use this state to configure the WSN node to read data less frequently when the WSN node is disconnected from the WSN gateway and to log that data to the

user memory instead. Alternatively, you can configure the WSN node to stop reading data when the WSN node is disconnected from the WSN gateway.

- **DIO Notification**—Occurs when notifications occur on the DIO lines. Use this state to configure the WSN node to respond to external events. For example, you can use this state to sample data from the analog or digital lines or count events.

Note The value of a local variable initialized in one state is not passed to the same local variable in a different state. You can use a global variable instead to pass values between states. Alternatively, you can use an uninitialized shift register or a Feedback Node.

The WSN VI must periodically exit these states to receive messages from the gateway. As a result, if the application delays too long in a state and blocks the state machine, the application might crash. National Instruments recommends that you use the following guidelines when determining how long the application remains in states:

- The VI must exit states in less than heartbeat/2, which is approximately 30 seconds, to maintain a connection to the network. If the WSN node is not connected to the gateway, the VI must exit states in less than heartbeat interval/16, which is approximately 3.8 seconds. Failure to exit states might prevent the WSN node from connecting the gateway.
- Minimize CPU usage by the VI; otherwise, the WSN node might not receive messages.
- If you are using a WSN voltage node, the VI must exit states in less than 9 seconds; otherwise, the MUX cannot maintain high impedance, and analog input accuracy will decrease. Alternatively, you can read the analog input every 9 seconds, which causes the MUX capacitors to recharge.

Note Refer to the KnowledgeBase at `ni.com` for more information about the WSN execution model.

Limitations in Developing Block Diagrams (WSN)

Because of hardware and software differences between devices and PCs, some block diagram features are not supported or are supported differently. If you place an unsupported object on the block diagram, the WSN VI does not run and you receive errors when you try to build the VI into an application. In most cases you see a broken **Run** button.

WSN VIs support the following block diagram objects differently from how VIs running on Windows support these objects:

- Array Functions
- Data Communication functions
- Memory Control functions
- Numeric functions
- String functions
- Structures

WSN VIs also differ in data type support. Some VIs and functions also are unsupported.

Limitations with Array Functions (WSN)

The following functions are unsupported:

- Array to Matrix
- Matrix to Array

Limitations with Data Communication Functions (WSN)

The Wireless Sensor Network Module supports the following Shared Variable node, VI, and functions:

- Local Variable Object Reference
- Write Variable

The Wireless Sensor Network Module only supports the NI-PSP I/O variable type of shared variable. You use I/O variables in WSN VIs to read data from and write data to the host.

Limitations with Memory Control Functions (WSN)

The LabVIEW Wireless Sensor Network Module supports the following Memory Control functions:

- Data Value Reference Read / Write Element
- New Data Value Reference
- Delete Data Value Reference

The Wireless Sensor Network Module supports data value references with the following limitations:

- You cannot create a data value reference to a LabVIEW class object.
- You cannot use a data value reference with static memory models.
- You cannot use the Type Cast or Unflatten From String function to obtain a valid data value reference. You must use the New Data Value Reference function instead.
- To avoid deadlock, do not place an In Place Element structure with a data value reference inside another In Place Element structure with the same data value reference.
- When you use a Data Value Reference Read / Write Element border node, the border node does not return all error codes and messages.

Limitations with Numeric Functions (WSN)

The following VIs and functions are unsupported:

- Color to RGB
- RGB to Color
- Flatten To String
- Scaling VIs

The Round To Nearest and Round Toward –Infinity functions return the same output on timestamps because when a timestamp is rounded to the nearest integer, the timestamp value always is rounded down to the next lowest integer.

You cannot use the Type Cast or Unflatten From String function to obtain a valid data value reference. You must use the New Data Value Reference function instead.

Data Type Support (WSN)

The following data types are supported:

- Arrays
- Booleans
- Clusters

Note While clusters are supported, National Instruments recommends avoiding clusters, including error clusters, in WSN applications because clusters increase the size of the application.

- Double-precision, floating-point numerics (represented as single-precision, 32-bit, floating point numerics)

Note While double-precision, floating-point numerics are supported, National Instruments recommends avoiding double-precision, floating-point numerics in WSN applications because this data type slows the performance of the application.

- Enumerated types
- Fixed-point numerics
- Strings
- 8-bit, 16-bit, and 32-bit signed integers
- Timestamps

Note WSN nodes return timestamps in terms of time elapsed since 12:00 a.m., Thursday, January 1, 1970, Universal Time [01-01-1970 00:00:00]. WSN gateways return timestamps in terms of time elapsed since 12:00 a.m., Friday, January 1, 1904, Universal Time [01-01-1904 00:00:00].

- 8-bit, 16-bit, and 32-bit unsigned integers

The following data types are unsupported:

- Complex single-precision, floating-point numerics (CSG)
- Complex double-precision, floating-point numerics (CDB)
- Complex extended-precision, floating-point numerics (CXT)

- Digital
- Dynamic
- Extended-precision floating-point numerics (EXT)
- Paths
- References
- 64-bit signed and unsigned integers
- Variants
- Waveforms

Limitations with Fixed-Point Support (WSN)

The fixed-point data type has limited support. The Wireless Sensor Network Module accepts a maximum word length of 32 bits for fixed-point data.

Note Overflow mode is supported, but overflow status is not supported.

Comparison Functions

The following Comparison functions support the fixed-point data type:

- Equal?
- Equal To 0?
- Greater Or Equal?
- Greater Or Equal To 0?
- Greater?
- Greater Than 0?
- Less Or Equal?
- Less Or Equal To 0?
- Less?
- Less Than 0?
- Not Equal?
- Not Equal To 0?

Conversion Functions

The following Conversion functions support the fixed-point data type:

- Boolean Array To Number
- Number To Boolean Array
- To Byte Integer
- To Double Precision Float
- To Extended Precision Float
- To Fixed-Point
- To Long Integer
- To Quad Integer
- To Single Precision Float
- To Unsigned Byte Integer
- To Unsigned Long Integer
- To Unsigned Quad Integer
- To Unsigned Word Integer
- To Word Integer

Data Manipulation Functions

The following Data Manipulation functions support the fixed-point data type:

- Logical Shift
- Rotate Left With Carry
- Rotate Right With Carry

Fixed-Point Functions

The following Fixed-Point functions are supported:

- Fixed-Point to Integer Cast
- Integer to Fixed-Point Cast

Numeric Functions

The following Numeric functions support the fixed-point data type:

- Absolute Value
- Add
- Decrement
- Increment
- Multiply
- Negate
- Round To Nearest
- Round Toward +Infinity
- Round Toward –Infinity
- Scale By Power Of 2 Function
- Sign
- Subtract
- Square

String/Number Conversion Functions

The following String/Number Conversion functions support the fixed-point data type:

- Decimal String To Number
- Fract/Exp String To Number
- Hexadecimal String To Number
- Number To Decimal String
- Number To Engineering String
- Number To Exponential String
- Number To Fractional String
- Number To Hexadecimal String
- Number To Octal String

- Octal String To Number

Structures

The Inline C Node supports the fixed-point data type.

Limitations with String Functions (WSN)

The following functions are unsupported:

- Array of Strings to Path
- Path to Array of Strings
- Path to String
- String to Path
- Array To Spreadsheet String
- Spreadsheet String To Array
- Match Pattern
- Match Regular Expression

The Format Into String and Scan From String functions do not support the `%x` or `%z` format specifier. The Scan From String function does not support wildcard matches. The Format Into String and Scan From String functions do not support timestamps or clusters.

The Scan String For Tokens function does not support caching of delimiter or operator data.

The Search and Replace String function does not support regular expression mode. Right-clicking the function and selecting **Regular Expression** has no effect on the function. The following Search and Replace String inputs are unsupported because they apply to regular expression mode:

- multiline?
- ignore case?

Limitations with Structures (WSN)

The following structures are unsupported:

- Event structure
- All shared variables except NI-PSP I/O variables
- Timed Structures and VIs

I/O Variables

The Wireless Sensor Network Module only supports the NI-PSP I/O variable type of shared variable. You use I/O variables in WSN VIs to read data from and write data to the host.

Formula Nodes

Formula Nodes have the following restrictions:

- You must use strict C code or LabVIEW cannot build the VI into an application. LabVIEW does not check for strict C code at edit time.
- Clusters are unsupported in the Formula Node.
- Only 1D and 2D arrays are supported in the Formula Node.
- You cannot define variables inside of a Formula Node. For example, `int x ;` and `double y;` result in an error when you build the VI into an application. The only way to create variables in a Formula Node is to create the variables as inputs and outputs.
- The `**` power function is unsupported. To perform a power operation in a Formula Node, you must use the `pow()` function. For example, replace `Y=X**4;` with `Y=pow(X,4);`.
- You cannot declare data types for the inputs or outputs. If you declare outputs inside of a Formula Node, the Formula Node causes syntax errors in the generated code. Create an input with the same name and type as the output to avoid declaring the output inside of a Formula Node.
- All Formula Node outputs are floating-point values.

Conditional Disable Structure

Use the Conditional Disable structure to execute a subdiagram based on the target operating system.

Unsupported VIs and Functions (WSN)

The following VIs and functions are unsupported:

- Application Control VIs and functions
- Class VIs and functions
- Connectivity VIs and functions
- Dialog & User Interface VIs and functions
- Express VIs and functions
- File I/O VIs and functions
- Graphics & Sound VIs
- Mathematics VIs and functions except for the Exponential functions, Hyperbolic functions, Numeric functions, and Trigonometric functions.
- Report Generation VIs
- Synchronization VIs and functions
- Variant functions
- Waveform VIs and functions

Performance and Memory Considerations (WSN)

You can improve the performance of WSN applications in the following ways:

- Avoid unnecessary memory allocation and deallocation by not using heap-allocated data types, such as arrays, clusters and strings. While clusters are supported, National Instruments recommends avoiding clusters, including error clusters, in WSN applications because clusters increase the size of the application.
- Avoid using large constants, such as arrays, inside loops. Avoid building arrays inside loops.

- Avoid unnecessary string manipulation.
- Use shift registers instead of loop tunnels.
- Avoid floating-point calculations.
- Use global variables instead of local variables.
- Use the Conversion VIs and functions instead of coercion dots.
- Use the Inline C Node when your application includes a numeric or array algorithm that can be coded more optimally in C.

User Memory (WSN)

You can access a portion of the flash memory on NI WSN nodes, referred to as **user memory**. Use user memory to store configuration data or other essential information that must persist across power cycles and firmware updates on the WSN node. For example, you can store the sample interval value in user memory, and access this value in the Start case when the WSN node initializes. You also can use user memory to store sensor conversion constants. Send user messages from the host to update configuration data.

Use the User Memory VIs to access data in the user memory sector. National Instruments recommends avoiding data logging to user memory because flash memory has a limited number of read/write cycles (100,000 erase cycles per user memory sector). While you can keep track of the number of erase cycles per user memory sector, the flash memory on a WSN node cannot accommodate heavy read/write traffic. If you continuously log data to user memory, you will likely exceed the specified life of the flash memory before the normal operational life of the WSN node.

Using Elemental I/O Nodes (WSN)

Use the Elemental I/O items in Elemental I/O Nodes on the block diagram. You can drag an Elemental I/O item from the **Project Explorer** window to the block diagram to create a new Elemental I/O Node that contains an Elemental I/O item.

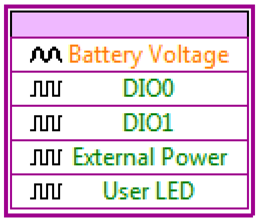
You also can complete the following steps to add an Elemental I/O Node to the block diagram and configure terminals for specific Elemental I/O items.

1. Create a new VI or open an existing VI under an WSN target that contains Elemental I/O items. By default, WSN targets contain Elemental I/O items for the WSN node.
2. Place an Elemental I/O Node on the block diagram.
3. Click the element section of the Elemental I/O Node to add a new Elemental I/O item or select an Elemental I/O item you previously added to the project. You can select any Elemental I/O item you want to use, regardless of the type of I/O resource. The new Elemental I/O item appears in the Elemental I/O Node.

Expanding an Elemental I/O Node

You can expand the Elemental I/O Node to add additional Elemental I/O items.

The following graphic shows an Elemental I/O Node containing multiple Elemental I/O items.



You can expand an Elemental I/O Node by clicking the upper or lower edge of the node with the Positioning tool and dragging the edge up or down. LabVIEW automatically fills in each additional terminal with Elemental I/O items in the order they appear in the **Project Explorer** window.

You can change the order in which the Elemental I/O items appear in the **Project Explorer** window prior to expanding the node. In the **Project Explorer** window, select the Elemental I/O item under the WSN target and drag the Elemental I/O item to the new position in the project tree. If you change the order of items in the **Project Explorer** window after you expand the node on the block diagram, the node does not update until you collapse the node and expand the node again.

You also can expand the Elemental I/O Node by right-clicking an Elemental I/O item in the Elemental I/O Node and selecting **Add Element** from the shortcut menu. A new unconfigured **I/O Name** terminal appears in the Elemental I/O Node. Right-click the unconfigured **I/O Name** terminal and select **Select Elemental I/O** from the shortcut menu.

Note LabVIEW does not necessarily process items in the Elemental I/O Node in the order that they appear. Add individual Elemental I/O Nodes to the block diagram for each Elemental I/O item to control the order of processing and data flow.

Using Elemental I/O Property Nodes (WSN)

You can use Elemental I/O Property Nodes to retrieve and set properties programmatically on Elemental I/O items in a LabVIEW project. The properties you can select are specific to the WSN target and the I/O resource that the Elemental I/O item is associated with in the Project Explorer window.

Complete the following steps to create and configure Elemental I/O Property Nodes.

1. Create a new VI or open an existing VI that is under an WSN target that also contains Elemental I/O items.
2. Place an Elemental I/O Property Node on the block diagram.
3. Right-click the Elemental I/O Property Node and select **Select Item** from the shortcut menu. The **Select Item** submenu displays the Elemental I/O items present in the **Project Explorer** window under the WSN target. Select the Elemental I/O item you want to use.
4. Right-click the **Property** terminal in the Elemental I/O Property Node and select **Select Property** from the shortcut menu to select a property you want to assign to the Elemental I/O item. LabVIEW displays the properties available for the WSN target in the **Select Property** submenu. LabVIEW displays **No Properties Available** in the shortcut menu if the WSN target does not support properties for the Elemental I/O item you select.

   Tip You also can click the Elemental I/O Property Node and use the shortcut menu to select properties available for the WSN target.

You can add additional **Property** terminals in the Elemental I/O Property Node by right-clicking the node and selecting **Add Element** from the shortcut menu. A new **Property** terminal appears in the Elemental I/O Property Node.

Tip You also can expand the Elemental I/O Property Node by clicking on the upper or lower edge of the node with the Positioning tool and dragging the edge up or down.

Properties for NI WSN Nodes

If you are using a LabVIEW WSN target, WSN nodes include the following properties that you can use with the Elemental I/O Property Node:

NI WSN Analog Properties:

- NI WSN-3212 Analog Input Properties
- NI WSN-3202 Analog Input Properties
- NI WSN-3214 Analog Input Properties
- NI WSN-3226 Analog Input Properties

NI WSN Digital I/O Properties:

- NI WSN-3212 Digital I/O Properties
- NI WSN-3202 Digital I/O Properties
- NI WSN-3214 Digital I/O Properties
- NI WSN-3226 Digital I/O Properties
- NI WSN-3230/3231 Digital I/O Properties

NI WSN Nodes Analog Properties

If you are using an NI LabVIEW WSN target, NI WSN nodes include the following analog properties that you can use with the Elemental I/O Property Node:

- NI WSN-3212 Analog Input Properties
- NI WSN-3202 Analog Input Properties
- NI WSN-3214 Analog Input Properties
- NI WSN-3226 Analog Input Properties

Analog Input Properties for the NI WSN-3202 Voltage Node

The following properties are available for Elemental I/O Property Node when using an NI WSN-3202 voltage node with LabVIEW WSN. These properties define the behavior of AI0–AI3 Elemental I/O items.

Property	Description
Range	Returns or sets the gain for the corresponding a nalog input. The default value is **Host Driven**. This property can contain the following values: ▪ ±0.5 V ▪ ±2 V ▪ ±5 V ▪ ±10 V ▪ Host Driven
Sensor Power	Returns or sets the sensor excitation for the corr esponding analog input. The default value is **Ho st Driven**. This property can contain the following values: ▪ 0 ms before sampling ▪ 25 ms before sampling ▪ 100 ms before sampling ▪ 250 ms before sampling ▪ Always On ▪ Host Driven

Analog Input Properties for NI WSN-3212 Thermocouple Node

The following properties are available for Elemental I/O Property Nodes when using an NI WSN-3212 thermocouple node with LabVIEW WSN. These properties define the behavior of Elemental I/O items TC0–TC3.

Property	Description
CJC Source	Returns or sets CJC source. The default value is **Host Driven**. This property can contain the following values: ▪ Internal CJC ▪ 0 C ▪ 25 C ▪ Host Driven
Range	Returns or sets the thermocouple range. The def ault value is **Host Driven**. This property can contain the following values: ▪ -273.15 to 1820 Celsius ▪ 0 to 2093 Kelvin ▪ -459.67 to 3308 Fahrenheit ▪ -0.073 to 0.073 Volts ▪ Host Driven
Thermocouple Type	Returns or sets the thermocouple type. The defa ult value is **Host Driven**. This property can contain the following values: ▪ J ▪ K ▪ T ▪ E ▪ R ▪ S

	▪ N ▪ B ▪ Host Driven

Analog Input Properties for NI WSN-3214 Strain Node

The following properties are available for Elemental I/O Property Nodes when using an NI WSN-3214 strain node with LabVIEW WSN. These properties define the behavior of Elemental I/O items AI0–AI3.

Property	Description
Channel Configuration	Returns or sets the Channel Bridge Configuratio n. The default is value **Host Driven**. This property can contain the following values: ▪ Host Driven ▪ Full Bridge Strain Type I ▪ Full Bridge Strain Type II ▪ Full Bridge Strain Type III ▪ Half Bridge Strain Type I ▪ Half Bridge Strain Type II ▪ Quarter Bridge Strain Type I ▪ Full Bridge Ratiometric ▪ Half Bridge Ratiometric ▪ Quarter Bridge Ratiometric ▪ Channel Disabled For detailed descriptions of the bridge configura tions, see NI WSN-3214 Waveform Analog Input Properties and Strain Gage Bridge Configurations.
Gage Factor	Returns or sets the Gage Factor. The Gage factor specifies the sensitivity of the strain gages and r

	elates the change in electrical resistance to the change in strain. Each gage in the bridge must have the same gage factor. Refer to the sensor documentation to determine this value. The default value is **2**. This property can contain the following values: ▪ 0 < Gage Factor < 800
Offset Null	Returns of sets the Offset Null. The offset null value is added to the calibrated ratiometric reading. This value is applied in all bridge configurations and can be set manually or obtained by the Offset Nulling VI. The default value is **0 mV/V**. This property can contain the following values: ▪ -25 to +25 mV/V
Poisson Ratio	Returns or sets the Poisson Ratio. Poisson ratio is the ratio of lateral strain to axial strain in the material you are measuring. The default value is **0**. This property can contain the following values: ▪ -1 to +1
Quarter Bridge Gage Resistance	Returns or sets the Quarter Bridge Gate Resistance. Quarter bridge gage resistance is the resistance in ohms of the gage in an unstrained position. The default value is **350**. This property can contain the following values: ▪ 350 Ohms ▪ 1000 Ohms
Shunt Calibration (1/Actual Gain)	Returns or sets the Shunt Calibration Gain. The shunt calibration value is a gain applied to the post-adc calibrated ratiometric value. This value will always be applied regardless of bridge configuration, as it can be calculated and/or measured through external means. This value is applied i

	n all configurations. This value can be set manu ally or obtained by the Shunt Calibration VI. **Note** The hardware can perform a shunt calibration in Quarter Bridge mode only. S hunt calibration must be done before an of fset null calibration. The default value is **1**. This property can contain the following values: ▪ 0 – 2

Analog Input Properties for the NI WSN-3226 Voltage Ohm Node

The following properties are available for Elemental I/O Property Node when using an NI WSN-3226 voltage/RTD node with LabVIEW WSN. These properties define the behavior of AI0–AI3 Elemental I/O items. You use the Elemental I/O Property Node to set the properties that are configured on a per-channel basis. For analog input properties that apply to all channels in a node, refer to the Config Node VI topic.

Property	Description
Measurement Type	Returns or sets the Measurement Type. The defa ult value is **Host Driven**. This property can contain the following values: ▪ **Voltage**—Measures voltage in V. ▪ **Resistance**—Measures resistance in O hms. ▪ **RTD**—Measures RTD temperature in de grees C, F, or K as specified by RTD Tempe rature Scale. ▪ **Host Driven**—For this attribute, use th e setting from the host computer.
RTD Coefficient	Returns or sets the RTD Coefficient (per degrees Celsius). The default value is **Host Driven**.

	This property can contain the following values: ▪ **3750**—Measures the temperature of an RTD with a=.003750. ▪ **3851**—Measures the temperature of an RTD with a=.003851. ▪ **3911**—Measures the temperature of an RTD with a=.003911. ▪ **3916**—Measures the temperature of an RTD with a=.003916. ▪ **3920**—Measures the temperature of an RTD with a=.003920. ▪ **3928**—Measures the temperature of an RTD with a=.003928. ▪ **Host Driven**—For this attribute, use th e setting from the host computer.
RTD Temperature Scale	Returns or sets the RTD Temperature Scale. The default value is **Host Driven**. This property can contain the following values: ▪ **Deg C**—Measures the temperature in d egree Celsius. ▪ **Deg F**—Measures the temperature in d egrees Fahrenheit. ▪ **Deg K**—Measures the temperature in d egrees Kelvin. ▪ **Host Driven**—For this attribute, use th e setting from the host computer.

NI WSN Nodes Digital I/O Properties

If you are using an NI LabVIEW WSN target, NI WSN nodes include the following properties that you can use with the Elemental I/O Property Node:

- NI WSN-3212 Digital I/O Properties

- NI WSN-3202 Digital I/O Properties
- NI WSN-3214 Digital I/O Properties
- NI WSN-3226 Digital I/O Properties
- NI WSN-3230/3231 Digital I/O Properties

Digital I/O Properties for the NI WSN-3202 Voltage Node

The following Digital I/O properties are available for Elemental I/O Property Nodes when using an NI WSN-3202 node with LabVIEW WSN. These properties define the behavior of DIO0–DIO3 Elemental Property I/O items.

Property	Description
DIO Drive Mode	Returns or sets the drive mode for the correspo nding DIO lines. The default value is **Host Drive n**. This property can contain the following value s: ▪ **Tristate**—Does not drive the DIO line re gardless of the DIO output value. ▪ **Drive Low Only**—For DIO output value 1, does not drive the DIO line. For DIO out put value 0, drive the DIO line low. ▪ **Drive High Only**—For DIO output valu e 1, drives the DIO line high. For DIO outp ut value 0, does not drive the DIO line. ▪ **Drive High and Low**—For DIO output value 1, drives the DIO line high. For DIO o utput value 0, drives the DIO line low. ▪ **Host Driven**—For this attribute, use th e setting from the host computer.
DIO Notification	Returns whether or not the corresponding DIO li ne has triggered a DIO notification.
Generate Notifications	Enables notification when events occur on the c orresponding DIO lines. The default value is **Dis**

	abled. This property can contain the following values: ▪ Disabled ▪ On Falling Edge ▪ On Rising Edge

Digital I/O Properties for the NI WSN-3212 Thermocouple Node

The following Digital I/O properties are available for Elemental I/O Property Nodes when using an NI WSN-3212 node with LabVIEW WSN. These properties define the behavior of DI0–DIO3 Elemental Property I/O items.

Property	Description
DIO Drive Mode	Returns or sets the drive mode for the correspo nding DIO lines. The default value is **Host Drive n**. This property can contain the following value s: ▪ **Tristate**—Does not drive the DIO line re gardless of the DIO output value. ▪ **Drive Low Only**—For DIO output value 1, does not drive the DIO line. For DIO out put value 0, drive the DIO line low. ▪ **Drive High Only**—For DIO output valu e 1, drives the DIO line high. For DIO outp ut value 0, does not drive the DIO line. ▪ **Drive High and Low**—For DIO output value 1, drives the DIO line high. For DIO o utput value 0, drives the DIO line low. ▪ **Host Driven**—For this attribute, use th e setting from the host computer.
DIO Notification	Returns whether or not the corresponding DIO li ne has triggered a DIO notification.

Generate Notifications	Enables notification when events occur on the c orresponding DIO lines. The default value is **Dis abled**. This property can contain the following values: ▪ Disabled ▪ On Falling Edge ▪ On Rising Edge

Digital I/O Properties for the NI WSN-3214 Strain Node

The following Digital I/O properties are available for Elemental I/O Property Nodes when using an NI WSN-3214 node with LabVIEW WSN. These properties define the behavior of DI0–DIO1 Elemental Property I/O items.

Property	Description
DIO Mode	Returns or sets the mode for the corresponding DIO lines. The default value is **Host Driven**. Thi s property can contain the following values: ▪ **DI - 24V Sinking**—Use this mode to co nnect to industrial 24V sourcing output de vices. The input has a low impedance to g round and input thresholds compatible wi th 24 V signaling. ▪ **DI - 24V Sinking with Power Manag ement**—This mode is similar to DI-24V Si nking, but the low impedance path to gro und is removed when the inputs are not b eing actively read. This may reduce power consumption of the output device. It shou ld only be used with output devices that h ave valid output states within 100 µs of ha ving a low impedance path presented to t he output.

- **DI - TTL Logic**—High impedance input s with TTL compatible thresholds; suitabl e for most 3 V or 5 V logic signals.
- **DI - Contact Closure**—The input has T TL compatible thresholds and provides a pull-up resistor to 3 V when the input is ac tively being read. This is suitable for conn ections to contact switches wired betwee n the input and ground. If a contact to gro und remains closed in this setting while DI O Notifications are not set to Disabled, the current into the contact switch increases power consumption and can reduce batte ry life.
- **DO - Drive Low (Sinking)**—For DIO o utput value 1, this mode does not drive th e DIO line. For DIO output value 0, this mo de drives the DIO line low. This setting do es not require a supply voltage on DIO_P WR.
- **DO - Drive High (Sourcing)**—For DIO output value 1, this mode drives the DIO li ne high to the DIO_PWR voltage. For DIO o utput value 0, this mode does not drive th e DIO line. This setting requires a supply v oltage connected to the DIO_PWR pin.
- **DO - 3V TTL Logic (Open-Collector with Pull-Up)**—For DIO output value 1, t his mode pulls-up the DIO line to 3 V throu gh a pull-up resistor. For DIO output value 0, this mode drives the DIO line low. Use t his mode to connect to 3 V and 5 V logic in puts with TTL compatible input threshold s. This setting does not require a supply v oltage on DIO_PWR.
- **DO - Drive High and Low (Sourcing and Sinking)**—For DIO output value 1, th is mode drives the DIO line high to the DIO _PWR voltage. For DIO output value 0, this mode drives the DIO line low. This setting

	requires a supply voltage connected to the DIO_PWR pin in order to drive high. If no supply is connected to DIO_PWR, this mode will operate as if it were set to the DO-Drive Low (Sinking) mode. ▪ **Host Driven**—For this attribute, use the setting from the host computer.
DIO Notification	Returns whether or not the corresponding DIO line has triggered a DIO notification.
Generate Notifications	Enables notification when events occur on the corresponding DIO lines. The default value is **Disabled**. This property can contain the following values: ▪ Disabled ▪ On Falling Edge ▪ On Rising Edge When this value is set to **Disabled**, the following power saving conditions occur whenever the inputs are not actively being read: ▪ In DI - Contact Closure mode the pull-up resistor is disabled, saving power whenever a contact to ground is closed. ▪ In DI - 24V Sinking with Power Management mode, the low impedance path to ground is removed, reducing the current draw from a connected output device. When this value is set to **On Falling Edge** or **On Rising Edge**, the inputs are continuously being actively read and the power saving conditions do not apply.

Digital I/O Properties for the NI WSN-3226 Voltage Ohm Node

The following Digital I/O properties are available for Elemental I/O Property Nodes when using an NI WSN-3226 node with LabVIEW WSN. These properties define the behavior of DI0–DIO1 Elemental Property I/O items.

Property	Description
DIO Mode	Returns or sets the mode for the corresponding DIO lines. The default value is **Host Driven**. Thi s property can contain the following values: ▪ **DI - 24V Sinking**—Use this mode to co nnect to industrial 24V sourcing output de vices. The input has a low impedance to g round and input thresholds compatible wi th 24 V signaling. ▪ **DI - 24V Sinking with Power Manag ement**—This mode is similar to DI-24V Si nking, but the low impedance path to gro und is removed when the inputs are not b eing actively read. This may reduce power consumption of the output device. It shou ld only be used with output devices that h ave valid output states within 100 µs of ha ving a low impedance path presented to t he output. ▪ **DI - TTL Logic**—High impedance input s with TTL compatible thresholds; suitabl e for most 3 V or 5 V logic signals. ▪ **DI - Contact Closure**—The input has T TL compatible thresholds and provides a pull-up resistor to 3 V when the input is ac tively being read. This is suitable for conn ections to contact switches wired betwee n the input and ground. If a contact to gro und remains closed in this setting while DI O Notifications are not set to Disabled, the current into the contact switch increases

	power consumption and can reduce battery life. ▪ **DO - Drive Low (Sinking)**—For DIO output value 1, this mode does not drive the DIO line. For DIO output value 0, this mode drives the DIO line low. This setting does not require a supply voltage on DIO_PWR. ▪ **DO - Drive High (Sourcing)**—For DIO output value 1, this mode drives the DIO line high to the DIO_PWR voltage. For DIO output value 0, this mode does not drive the DIO line. This setting requires a supply voltage connected to the DIO_PWR pin. ▪ **DO - 3V TTL Logic (Open-Collector with Pull-Up)**—For DIO output value 1, this mode pulls-up the DIO line to 3 V through a pull-up resistor. For DIO output value 0, this mode drives the DIO line low. Use this mode to connect to 3 V and 5 V logic inputs with TTL compatible input thresholds. This setting does not require a supply voltage on DIO_PWR. ▪ **DO - Drive High and Low (Sourcing and Sinking)**—For DIO output value 1, this mode drives the DIO line high to the DIO_PWR voltage. For DIO output value 0, this mode drives the DIO line low. This setting requires a supply voltage connected to the DIO_PWR pin in order to drive high. If no supply is connected to DIO_PWR, this mode will operate as if it were set to the DO-Drive Low (Sinking) mode. ▪ **Host Driven**—For this attribute, use the setting from the host computer.
DIO Notification	Returns whether or not the corresponding DIO line has triggered a DIO notification.
Generate Notifications	Enables notification when events occur on the corresponding DIO lines. The default value is **Dis**

	abled. This property can contain the following values: ▪ Disabled ▪ On Falling Edge ▪ On Rising Edge When this value is set to **Disabled**, the followin g power saving conditions occur whenever the i nputs are not actively being read: ▪ In DI - Contact Closure mode the pull-u p resistor is disabled, saving power whene ver a contact to ground is closed. ▪ In DI - 24V Sinking with Power Manage ment mode, the low impedance path to gr ound is removed, reducing the current dra w from a connected output device. When this value is set to **On Falling Edge** or **O n Rising Edge**, the inputs are continuously bei ng actively read and the power saving condition s do not apply.

Digital I/O Properties for the NI WSN-3230/3231 Serial Node

The following Digital I/O properties are available for Elemental I/O Property Nodes when using an NI WSN-3230 and 3231 node with LabVIEW WSN. These properties define the behavior of DI0–DIO1 Elemental Property I/O items.

Property	Description
DIO Mode	Returns or sets the mode for the corresponding DIO lines. The default value is **Host Driven**. Thi s property can contain the following values: ▪ **DI - 24V Sinking**—Use this mode to co nnect to industrial 24V sourcing output de vices. The input has a low impedance to g

round and input thresholds compatible with 24 V signaling.

- **DI - 24V Sinking with Power Management**—This mode is similar to DI-24V Sinking, but the low impedance path to ground is removed when the inputs are not being actively read. This may reduce power consumption of the output device. It should only be used with output devices that have valid output states within 100 µs of having a low impedance path presented to the output.
- **DI - TTL Logic**—High impedance inputs with TTL compatible thresholds; suitable for most 3 V or 5 V logic signals.
- **DI - Contact Closure**—The input has TTL compatible thresholds and provides a pull-up resistor to 3 V when the input is actively being read. This is suitable for connections to contact switches wired between the input and ground. If a contact to ground remains closed in this setting while DIO Notifications are not set to Disabled, the current into the contact switch increases power consumption and can reduce battery life.
- **DO - Drive Low (Sinking)**—For DIO output value 1, this mode does not drive the DIO line. For DIO output value 0, this mode drives the DIO line low. This setting does not require a supply voltage on DIO_PWR.
- **DO - Drive High (Sourcing)**—For DIO output value 1, this mode drives the DIO line high to the DIO_PWR voltage. For DIO output value 0, this mode does not drive the DIO line. This setting requires a supply voltage connected to the DIO_PWR pin.
- **DO - 3V TTL Logic (Open-Collector with Pull-Up)**—For DIO output value 1, t

	his mode pulls-up the DIO line to 3 V throu gh a pull-up resistor. For DIO output value 0, this mode drives the DIO line low. Use t his mode to connect to 3 V and 5 V logic in puts with TTL compatible input threshold s. This setting does not require a supply v oltage on DIO_PWR. ▪ **DO - Drive High and Low (Sourcing and Sinking)**—For DIO output value 1, th is mode drives the DIO line high to the DIO _PWR voltage. For DIO output value 0, this mode drives the DIO line low. This setting requires a supply voltage connected to th e DIO_PWR pin in order to drive high. If no supply is connected to DIO_PWR, this mo de will operate as if it were set to the DO- Drive Low (Sinking) mode. ▪ **Host Driven**—For this attribute, use th e setting from the host computer.
DIO Notification	Returns whether or not the corresponding DIO li ne has triggered a DIO notification.
Generate Notifications	Enables notification when events occur on the c orresponding DIO lines. The default value is **Dis abled**. This property can contain the following values: ▪ Disabled ▪ On Falling Edge ▪ On Rising Edge When this value is set to **Disabled**, the followin g power saving conditions occur whenever the i nputs are not actively being read: ▪ In DI - Contact Closure mode the pull-u p resistor is disabled, saving power whene ver a contact to ground is closed. ▪ In DI - 24V Sinking with Power Manage ment mode, the low impedance path to gr

	ound is removed, reducing the current draw from a connected output device. When this value is set to **On Falling Edge** or **On Rising Edge**, the inputs are continuously being actively read and the power saving conditions do not apply.

Digital I/O Properties for the NI WSN-3230/3231 Serial Node

The following Digital I/O properties are available for Elemental I/O Property Nodes when using an NI WSN-3230 and 3231 node with LabVIEW WSN. These properties define the behavior of DI0–DIO1 Elemental Property I/O items.

Property	Description
DIO Mode	Returns or sets the mode for the corresponding DIO lines. The default value is **Host Driven**. This property can contain the following values: ▪ **DI - 24V Sinking**—Use this mode to connect to industrial 24V sourcing output devices. The input has a low impedance to ground and input thresholds compatible with 24 V signaling. ▪ **DI - 24V Sinking with Power Management**—This mode is similar to DI-24V Sinking, but the low impedance path to ground is removed when the inputs are not being actively read. This may reduce power consumption of the output device. It should only be used with output devices that have valid output states within 100 µs of having a low impedance path presented to the output. ▪ **DI - TTL Logic**—High impedance inputs with TTL compatible thresholds; suitable for most 3 V or 5 V logic signals.

- **DI - Contact Closure**—The input has TTL compatible thresholds and provides a pull-up resistor to 3 V when the input is actively being read. This is suitable for connections to contact switches wired between the input and ground. If a contact to ground remains closed in this setting while DIO Notifications are not set to Disabled, the current into the contact switch increases power consumption and can reduce battery life.
- **DO - Drive Low (Sinking)**—For DIO output value 1, this mode does not drive the DIO line. For DIO output value 0, this mode drives the DIO line low. This setting does not require a supply voltage on DIO_PWR.
- **DO - Drive High (Sourcing)**—For DIO output value 1, this mode drives the DIO line high to the DIO_PWR voltage. For DIO output value 0, this mode does not drive the DIO line. This setting requires a supply voltage connected to the DIO_PWR pin.
- **DO - 3V TTL Logic (Open-Collector with Pull-Up)**—For DIO output value 1, this mode pulls-up the DIO line to 3 V through a pull-up resistor. For DIO output value 0, this mode drives the DIO line low. Use this mode to connect to 3 V and 5 V logic inputs with TTL compatible input thresholds. This setting does not require a supply voltage on DIO_PWR.
- **DO - Drive High and Low (Sourcing and Sinking)**—For DIO output value 1, this mode drives the DIO line high to the DIO_PWR voltage. For DIO output value 0, this mode drives the DIO line low. This setting requires a supply voltage connected to the DIO_PWR pin in order to drive high. If no supply is connected to DIO_PWR, this mo

	de will operate as if it were set to the DO-Drive Low (Sinking) mode. ▪ **Host Driven**—For this attribute, use the setting from the host computer.
DIO Notification	Returns whether or not the corresponding DIO line has triggered a DIO notification.
Generate Notifications	Enables notification when events occur on the corresponding DIO lines. The default value is **Disabled**. This property can contain the following values: ▪ Disabled ▪ On Falling Edge ▪ On Rising Edge When this value is set to **Disabled**, the following power saving conditions occur whenever the inputs are not actively being read: ▪ In DI - Contact Closure mode the pull-up resistor is disabled, saving power whenever a contact to ground is closed. ▪ In DI - 24V Sinking with Power Management mode, the low impedance path to ground is removed, reducing the current draw from a connected output device. When this value is set to **On Falling Edge** or **On Rising Edge**, the inputs are continuously being actively read and the power saving conditions do not apply.

Adding and Removing Header Files from an Inline C Node (WSN)

You can add or remove header files when you generate the C code in an Inline C Node.

Adding Header Files

Complete the following steps to include header files when you generate the C code in an Inline C Node.

1. Place an Inline C Node on the block diagram.
2. Right-click the Inline C Node and select **Add or Remove Header Files** from the shortcut menu to open the Add/Remove Header Files dialog box.
3. Click the **Add file** button.
4. Navigate to and select the header file(s) you want to include. Click the **Open** button.
5. (Optional) Place a checkmark in the **Show full paths** checkbox if you want to show the file path to the header file(s) you selected.
6. Click the **OK** button.

Removing Header Files

Complete the following steps to remove any header files you might have added when you generate the C code in an Inline C Node.

1. Place an Inline C Node on the block diagram.
2. Right-click and select **Add or Remove Header Files** from the shortcut menu to open the Add/Remove Header Files dialog box.
3. Select the header file(s) you want to remove.
4. Click the **Remove file** button.
5. Click the **OK** button.

Communicating with WSN Nodes (WSN)

You program, monitor, and control a wireless sensor network through a VI that runs on the host computer. The host VI uses the WSN Host API VIs to send messages to and receive messages from the WSN gateway and WSN nodes in the network, get information about the WSN nodes in the network, and deploy firmware to WSN nodes.

Add a VI under **My Computer** in the Project Explorer window to create a host VI.

You can use the WSN Host API VIs on any RT target, such as the NI 9792 WSN Real-Time Gateway. You must use the NI Measurement & Automation Explorer (MAX) to install the Host API VIs on the RT target before you can run them on the target.

Sending and Receiving Messages (WSN)

The host VI uses the WSN Host API VIs to send messages to and receive messages from the WSN gateway and WSN nodes in the network. You can use debug and user messages with LabVIEW WSN targets.

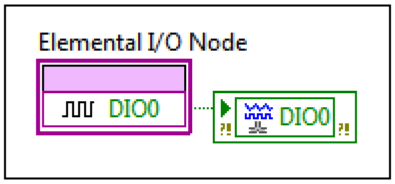
Debug messages contain run-time error information and are sent by the WSN node to the WSN gateway. You cannot send debug messages from the host.

User messages are reserved for the user. The host VI can send user messages to or receive user messages from the WSN node. To receive user messages from a VI running on a WSN RT gateway, use `localhost` or `127.0.0.1` as the IP address of the WSN RT gateway.

You can send an unlimited number of user and debug messages in a state. Both user and debug messages are sent immediately.

Reading and Writing Data (WSN)

WSN VIs use Elemental I/O to read from and write to analog and digital channels. You use I/O variables to read data from and write data to the host. To transfer the value of a hardware channel to the host, wire the Elemental I/O item to its corresponding static I/O variable, as shown in the following block diagram.

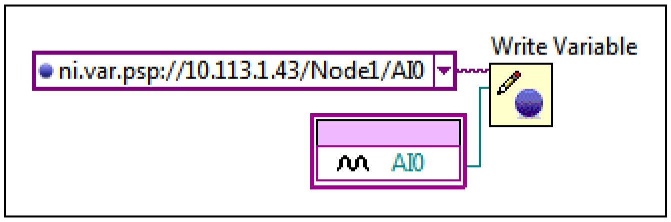


You access data using static I/O variables from the project tree in the Project Explorer window or programmatically using supported Shared Variable functions with the following URL format:

`ni.var.psp://`**gateway**`/`**node**`/`**channel**

- **gateway**—The IP address of the WSN gateway. For NI 9792 WSN Real-Time Gateway targets, you can use `localhost` for local I/O variable access.
- **node**—The WSN node name. The default node name is **Nodex** in the project tree, where **x** is the ID assigned to the WSN node when you added it to the LabVIEW project. Node names become active after you deploy the project to the gateway.
- **channel**—The I/O variable name. Default I/O variable names depend on the WSN node type. I/O variable names become active after you deploy the project to the gateway.

The following block diagram demonstrates how to use the AI0 I/O variable to write data from AI0 on Node1.



Note Use the supported Shared Variable functions if the data type of an I/O variable or user-defined I/O variable is a waveform. For I/O variables with other data types, use the static I/O variables for the project.

Creating User-Defined I/O Variables

You can create user-defined I/O variables to send data from the WSN node to the host. You can create a maximum of eight user-defined I/O variables per WSN node.

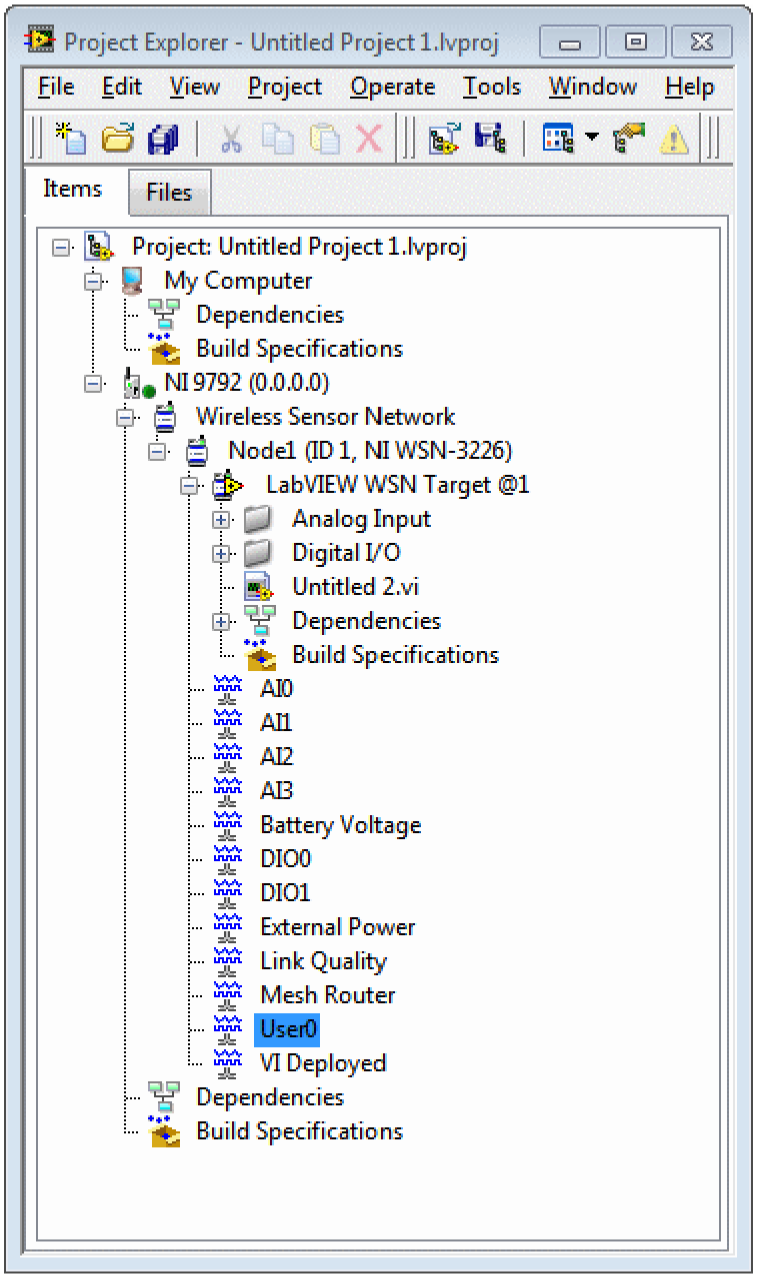
Complete the following steps to create a user-defined I/O variable.

1. Right-click the WSN node in the Project Explorer window and select **New»User-Defined Variable**.
2. Select the data type for the I/O variable. The **Length** text box automatically displays the maximum size for the selected data type.

   Note Some WSN nodes support only certain types of I/O variables.

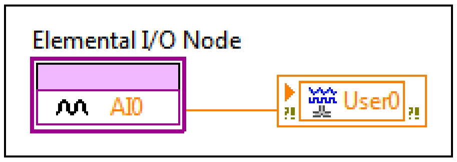
You only can create user-defined I/O variables that send data from the WSN node to the WSN gateway. Sending data from the WSN gateway to the WSN node is unsupported.

The following LabVIEW project includes User0, a user-defined I/O variable.


Project Explorer - Untitled Project 1.lvproj
File Edit View Project Operate Tools Window Help
Items Files

```
Project: Untitled Project 1.lvproj
    My Computer
        Dependencies
        Build Specifications
    NI 9792 (0.0.0.0)
        Wireless Sensor Network
            Node1 (ID 1, NI WSN-3226)
                LabVIEW WSN Target @1
                    Analog Input
                    Digital I/O
                    Untitled 2.vi
                    Dependencies
                    Build Specifications
                AI0
                AI1
                AI2
                AI3
                Battery Voltage
                DIO0
                DIO1
                External Power
                Link Quality
                Mesh Router
                User0
                VI Deployed
        Dependencies
        Build Specifications
```

The following block diagram demonstrates how to write the value of AI0 on Node1 to User0. User0 then sends that data to the WSN gateway.



Buffering Data

When you use I/O variables in a WSN VI, LabVIEW buffers the data in the I/O variable buffer, which is a first-in-first-out (FIFO) buffer. Buffering the data prevents data loss if the network connection fails.

You can use the Memory Configuration page to configure the size of the I/O variable buffer. When you increase the I/O variable buffer size, the amount of total RAM that the I/O variable buffer uses might increase significantly. If the network connection fails, the I/O variable buffer retains any data that LabVIEW wrote to the buffer.

You must enable buffering for each I/O variable to enable I/O variable buffering.

Supported User-Defined I/O Variables (WSN)

You can create user-defined I/O variables to send data from the WSN node to the host. A maximum of eight user-defined I/O variables can be created per node. The following table shows the supported variable types for specific WSN nodes.

	NI WSN-3202	NI WSN-3212	NI WSN-3226	NI WSN-3214	NI WSN-3230	NI WSN-3231
Boolean	✓	✓	✓	✓	✓	✓
Int16	✓	✓	✓	✓	✓	✓
Int32	✓	✓	✓	✓	✓	✓
Int8	✓	✓	✓	✓	✓	✓
Single	✓	✓	✓	✓	✓	✓
String†	✓	✓	✓	✓	✓	✓
UInt16	✓	✓	✓	✓	✓	✓
UInt32	✓	✓	✓	✓	✓	✓
UInt8	✓	✓	✓	✓	✓	✓
Waveform of Int16‡	⃠	⃠	⃠	✓	⃠	⃠

Waveform of Int32[‡]	🛇	🛇	🛇	✓	🛇	🛇
Waveform of Single[‡]	🛇	🛇	🛇	✓	🛇	🛇
Waveform of UInt16[‡]	🛇	🛇	🛇	✓	🛇	🛇
Waveform of UInt32[‡]	🛇	🛇	🛇	✓	🛇	🛇
Waveform of UInt8[‡]	🛇	🛇	🛇	✓	🛇	🛇

[†] Maximum string length is 256 characters.
[‡] Maximum waveform element length is 8192 elements.

Related information

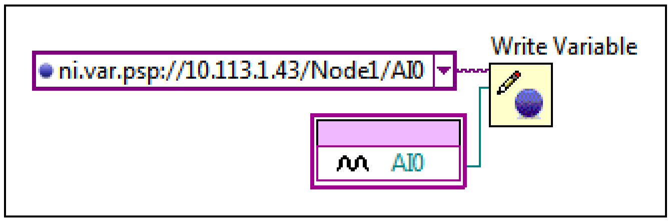
Creating User-Defined I/O Variables in Reading and Writing Data (WSN).

Accessing User RAM (WSN)

Some WSN nodes have user RAM, which is extra memory on the WSN node that you can use to store additional data. Refer to the WSN node documentation to determine if the node has user RAM.

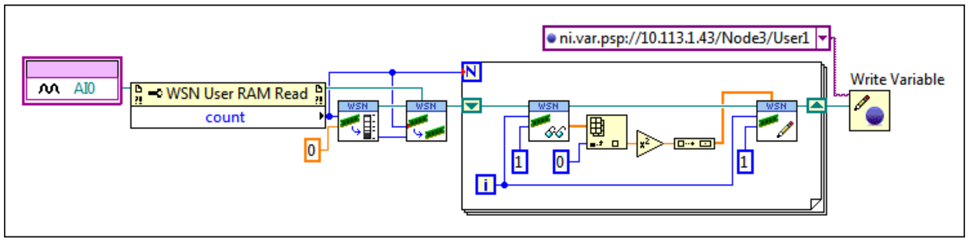
If a WSN node has user RAM, the data acquired from some physical channels is stored in the user RAM. In this case, the Elemental I/O item for the physical channel returns a reference to the data stored in the user RAM. You use the Write Variable function to write data to the I/O variables in the project.

The following block diagram reads data from AI0 on Node1 and uses the Write Variable function to write that data to the AI0 I/O variable.



You use the User RAM VIs to allocate and free space in the user RAM and to read data from and write data to specific blocks of memory in the user RAM.

The following block diagram reads data from AI0 on Node3 and copies the data to an allocated block in the user RAM. The example then writes the square of each element to the user RAM and then writes that data to User1, a user-defined I/O variable.



The user RAM shares RAM with the I/O variable buffer on WSN nodes. Use the Memory Configuration page to configure the amount of space allocated to the I/O variable buffer and the amount of RAM available as user RAM.

Building VIs into Applications (WSN)

After you develop a WSN VI for the WSN target, you can create a build specification and build the VI into an application that you can download and run on the WSN node.

Complete the following steps to create a build specification.

1. Right-click **Build Specifications** under the WSN target in the Project Explorer window and select **New»WSN Application** to open the **Build Specification Properties** dialog box.
2. Select the top-level VI for the application and click the blue arrow button to move the VI to the **Top-level VI** text box. WSN applications can have only one top-level VI.
3. Click the **OK** button.

After you create the build specification you can build the VI into an application.

Right-click a build specification under the WSN target in the Project Explorer window and select **Build** to build the WSN VI into an application. The **Build** option builds only files that have changed since the last time you built the application.

Note You can build WSN VIs on multiple WSN nodes in the same project simultaneously; however, you cannot build multiple VIs on the same WSN node simultaneously.

The WSN System Monitor launches automatically when you build or deploy a WSN application or run a WSN VI. Use the WSN System Monitor to display information about all WSN nodes and WSN gateways in the LabVIEW project, determine the build and deployment progress of a WSN application on the target, and view debug messages sent by WSN nodes to the WSN gateway.

Tip Specify a default WSN target build specification by right-clicking a build specification in the Project Explorer window and selecting **Set as Default**. If you create multiple build specifications for the target, you must specify a default build specification to use the **Run** button with WSN VIs.

Deploying Applications to WSN Nodes (WSN)

You must create a build specification to deploy an application to a WSN target.

Right-click the build specification and select **Deploy Application** to deploy an application. You also can right-click the top-level VI for the application and select **Run**.

Tip You can use the WSN Host API VIs to deploy applications programmatically.

You can deploy WSN VIs on multiple WSN nodes in the same project; however, you cannot deploy multiple VIs on the same WSN node.

The WSN System Monitor launches automatically when you build or deploy a WSN application or run a WSN VI. Use the WSN System Monitor to display information about all WSN nodes and WSN gateways in the LabVIEW project, determine the build and deployment progress of a WSN application on the target, and view debug messages sent by WSN nodes to the WSN gateway.

Debugging Applications (WSN)

LabVIEW does not support debugging applications on the WSN node. To debug a WSN application, send debug messages to the WSN gateway via radio communications. You also can blink the LED on the device to indicate sample state so that you have a visual indication that the application is working.

Debug messages contain diagnostic error information and are sent by the WSN node to the WSN gateway if the WSN node crashes. You cannot send debug messages from the host. When the WSN node sends debug messages, the radio powers on, and the message is sent immediately and synchronously. Error messages that are too large for a single radio packet are truncated and sent via multiple radio messages. Use the Config Node to determine the maximum packet size.

You can view debug messages in the WSN System Monitor. Right-click **LabVIEW WSN Target** in the Project Explorer window and select **WSN System Monitor** to display the WSN System Monitor.

LabVIEW WSN Module Error Codes

The LabVIEW WSN Module can return the following error codes. Refer to the KnowledgeBase for more information about correcting errors in the LabVIEW WSN Module.

Code	Description
−25339	You must enable the serial port receiver power b efore calling this function, or no new data can b e read.
−25338	The device does not support reading or writing t he specified number of bytes from the serial por t at one time.
−25337	The specified number of bytes cannot be read o r written within the specified timeout.
−25336	The specified serial port configuration is unsupp orted on this device.
−25327	Not enough user RAM.

−25326	Invalid user RAM refnum. This error can occur w hen you try to perform an operation on a null re fnum or a previously closed refnum.
−25325	Verification of user memory failed.
−25324	Failed to write to user memory.
−25323	Failed to write to user memory because the loca tion is not empty. To correct this error, you must erase the entire user memory sector before you can write to the same location again.
−25322	User memory out of bounds. The operation spe cified a location outside the user memory secto r.
−25321	Failed to erase the specified user memory secto r.
−25320	Null pointer exception.
−25313	Internal error: Hardware failure.
−25312	The node is not connected.
−25311	Invalid argument.
−25310	User message missed.
−25309	Radio failure: An internal error has occurred. Ple ase contact National Instruments technical sup port at ni.com/support.
−25308	Flash failure: An internal error has occurred. Ple ase contact National Instruments technical sup port at ni.com/support.
−25307	Command queue overflow: An internal error has occurred. Please contact National Instruments t echnical support at ni.com/support.
−25306	Digital output line failure: An internal error has o ccurred. Please contact National Instruments te chnical support at ni.com/support.
−25305	ADC failure: An internal error has occurred. Plea se contact National Instruments technical supp ort at ni.com/support.
−25304	CJC failure: An internal error has occurred. Pleas e contact National Instruments technical suppo rt at ni.com/support.

−25303	SPI communication failure: An internal error has occurred. Please contact National Instruments t echnical support at ni.com/support.
−25302	Serial receiving failure: An internal error has occ urred. Please contact National Instruments tech nical support at ni.com/support.
−25301	Serial transmission failure: An internal error has occurred. Please contact National Instruments t echnical support at ni.com/support.
−25300	An internal error has occurred in the LabVIEW Wi reless Sensor Network Module. Please contact N ational Instruments technical support at ni.com /support.
−25283	The node ID does not exist in the gateway's nod e table.
−25282	Invalid message reference.
−25281	Invalid node reference.
−25280	Invalid gateway reference.
−25254	Connection timed out. Gateway might be in a b ad state.
−25253	Double interface registration: An internal error h as occurred. Please contact National Instrument s technical support at ni.com/support.
−25252	Incompatible or corrupted data was received fro m the gateway.
−25251	Memory error. Unable to deallocate used memo ry.
−25250	Memory allocation failed.
−25230	The request processor failed to deserialize a req uest. This error can occur if protocol versions ar e incompatible or data is corrupt.
−25229	The request processor failed to deserialize a req uest. This error can occur if protocol versions ar e incompatible.
−25228	The request terminated due to an unhandled ex ception.
−25227	Memory allocation failed.

−25226	The client was denied access.
−25225	The client aborted the connection.
−25224	Network communication failed. This error can occur if the gateway IP address is incorrect, the gateway is disconnected, or the gateway is unable to complete the request before timing out.
−25223	Connection timed out.
−25222	Unable to connect to the named service.
−25221	Unable to connect to the named gateway.
−25220	Generic connection error.
21240	The serial port receiver received a break condition on the line.
21241	The specified number of bytes were read from the serial port before the specified termination string was detected.

Wireless Sensor Network Module VIs

June 2013, 372803E-01

Requires: LabVIEW WSN Module. This topic might not match its corresponding palette in LabVIEW depending on your operating system, licensed product(s), and target.

Use the WSN VIs to create applications for LabVIEW WSN targets.

Subpalette	Description
Elemental I/O Functions	Use the Elemental I/O functions to perform Elemental I/O operations on WSN targets.
WSN Host API VIs	Use the WSN Host API VIs to program, monitor, and control a wireless sensor network from a host computer.
WSN VIs	Use the WSN VIs to create applications for LabVIEW WSN targets.

Elemental I/O Functions

Owning Palette: Wireless Sensor Network Module VIs, WSN VIs

Requires: LabVIEW WSN Module. This topic might not match its corresponding palette in LabVIEW depending on your operating system, licensed product(s), and target.

Use the Elemental I/O functions to perform Elemental I/O operations on WSN targets.

Palette Object	Description
Elemental IO Node	Performs specific Elemental I/O operations on WSN targets. You can configure the Elemental I/O Node with one or more Elemental I/O items. Elemental I/O resources can perform both read and write operations.
Elemental IO Property Node	Gets or sets one or more properties for an Elemental I/O item in the project. The I/O resources available and the associated properties vary by target and configuration.

Elemental IO Node

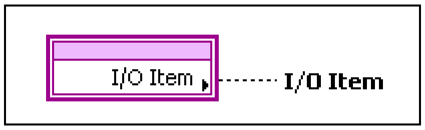
Owning Palette: Elemental I/O Functions

Requires: LabVIEW WSN Module

Performs specific Elemental I/O operations on WSN targets. You can configure the Elemental I/O Node with one or more Elemental I/O items. Elemental I/O resources can perform both read and write operations.

To add Elemental I/O items to an Elemental I/O Node, right-click the **I/O Item** section and select **Add Element** from the shortcut menu. You also can expand or contract the Elemental I/O Node by clicking the upper or lower edge of the node with the Positioning tool and dragging the edge up or down.

Details



I/O Item is the data read from or written to the Elemental I/O item you specify. For Elemental I/O items that support the read operation, this parameter is an indicator. For I/O items that do not support the read operation, this parameter is a control. For Elemental I/O items that support reading and writing, this parameter can be either a control or an indicator.

Elemental IO Node Details

When you right-click the **I/O Item** terminal of the Elemental I/O Node and select **Select Elemental I/O** from the shortcut menu, LabVIEW displays the Elemental I/O items that appear in the Project Explorer window below the same WSN target as the VI you are currently editing. You also can right-click the **I/O Item** terminal and select **Add New Elemental I/O** from the shortcut menu to add new Elemental I/O items under the WSN target.

Digital input and output (DIO) resources can perform both read and write operations. To change the operation of an Elemental I/O item, right-click the element and select **Change to Read** or **Change to Write** from the shortcut menu.

When you configure an Elemental I/O Node with multiple I/O items, the execution timing of the I/O items depends on the specific hardware devices and I/O items you use. Some Elemental I/O operations execute in parallel, while others execute sequentially.

Tip Right-click an element in the Elemental I/O Node and select **Find Elemental I/O in Project** from the shortcut menu to highlight the Elemental I/O item in the Project Explorer window.

Elemental IO Property Node

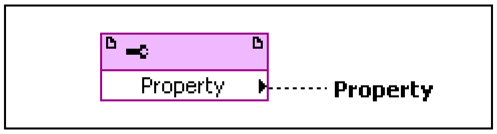
Owning Palette: Elemental I/O Functions

Requires: LabVIEW WSN Module

Gets or sets one or more properties for an Elemental I/O item in the project. The I/O resources available and the associated properties vary by target and configuration.

To select a property, first configure the Elemental I/O Property Node with an Elemental I/O item. Then, click the **Property** section of the Elemental I/O Property Node and select a property from the shortcut menu. To add additional properties to an Elemental I/O Property Node, right-click the **Property** section and select **Add Element** from the shortcut menu. You also can expand or contract the Elemental I/O Property Node by clicking the upper or lower edge of the node with the Positioning tool and dragging the edge up or down.

Details



POLY **Property** is the value of the Elemental I/O property you specify. To specify whether this parameter is a control or an indicator, right-click an element in the section of the Elemental I/O Property Node and select **Change to Read** or **Change to Write** from the shortcut menu.

Elemental IO Property Node Details

Support for use of the Elemental I/O Property Node varies by target, Elemental I/O item, and property. Not all Elemental I/O items have properties you can specify with the Elemental I/O Property Node.

When you expand the Elemental I/O Property Node, LabVIEW displays a **Property** parameter for each element of the Node.

Tip Right-click the Elemental I/O Property Node and select **Find Item in Project** from the shortcut menu to highlight the Elemental I/O item in the Project Explorer window.

WSN Host API VIs

Owning Palette: Wireless Sensor Network Module VIs

Requires: LabVIEW WSN Module. This topic might not match its corresponding palette in LabVIEW depending on your operating system, licensed product(s), and target.

Use the WSN Host API VIs to program, monitor, and control a wireless sensor network from a host computer.

Note You can use the WSN Host API VIs on any RT target, such as the NI 9792 WSN Real-Time Gateway. You must use the NI Measurement & Automation Explorer (MAX) to install the Host API VIs on the RT target before you can run them on the target.

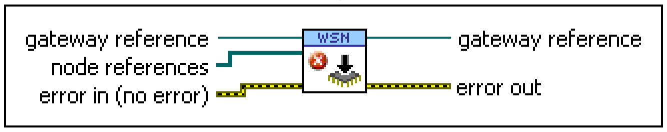
Palette Object	Description
WSN Cancel Firmware Deployment	Cancels the uploading of the firmware to the WSN nodes in the network.
WSN Cancel Message	Cancels sending a message to a WSN node.
WSN Close Gateway	Closes the reference to the WSN gateway.
WSN Close Message	Closes the reference to a message sent to a WSN node.
WSN Close Node	Closes the reference to the specified WSN node.
WSN Deploy Firmware	Downloads the application to the WSN nodes.
WSN Discover All Nodes	Returns information about the WSN network from the WSN gateway.
WSN Gateway Time	Returns the current time from the WSN gateway.
WSN Get New Messages	Returns all new messages received by the WSN gateway from a WSN node since a specified starting time.
WSN Get Node Info	Returns information about the specified WSN node from the gateway.
WSN Message Status	Returns the status of a message sent to a WSN node.
WSN Open Gateway	Creates a reference to the WSN gateway.
WSN Open Node Reference	Creates a reference to the specified WSN node.
WSN Send Message	Sends a user message to a WSN node.

WSN Cancel Firmware Deployment VI

Owning Palette: WSN Host API VIs

Requires: LabVIEW WSN Module

Cancels the uploading of the firmware to the WSN nodes in the network.



gateway reference specifies the reference to the WSN gateway.

node references includes references to all WSN nodes in the network. This VI sequentially downloads the application specified in **firmware image path** to each WSN node.

error in describes error conditions that occur before this node runs. This input provides standard error in functionality.

gateway reference returns the reference to the WSN gateway.

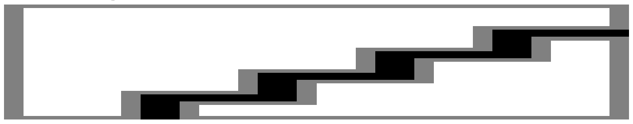
error out contains error information. This output provides standard error out functionality.

WSN Cancel Message VI

Owning Palette: WSN Host API VIs

Requires: LabVIEW WSN Module

Cancels sending a message to a WSN node. Cancelling a message does not close references to the message. If the gateway has already sent the message, the message will not be cancelled.



message reference is the reference to the message.

error in describes error conditions that occur before this node runs. This input provides standard error in functionality.

message status returns the status of the message. The output can return the following values.

Cancelled	Sending the message was cancelled.
Failed	Sending the message failed.
Sent to node	Sending the message completed.
Waiting to send	The gateway is waiting to send the message to the WSN node.

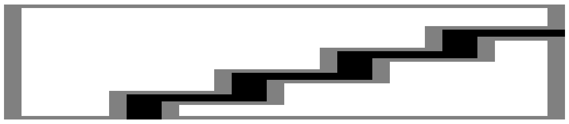
error out contains error information. This output provides standard error out functionality.

WSN Close Gateway VI

Owning Palette: WSN Host API VIs

Requires: LabVIEW WSN Module

Closes the reference to the WSN gateway.



gateway reference specifies the reference to the WSN gateway.

error in describes error conditions that occur before this node runs. This input provides standard error in functionality.

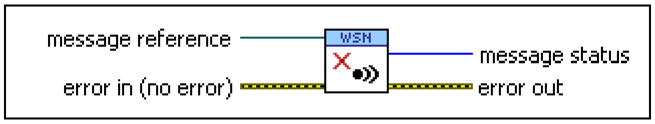
error out contains error information. This output provides standard error out functionality.

WSN Close Message VI

Owning Palette: WSN Host API VIs

Requires: LabVIEW WSN Module

Closes the reference to a message sent to a WSN node.



message reference is the reference to the message.

error in describes error conditions that occur before this node runs. This input provides standard error in functionality.

message status returns the status of the message. The output can return the following values.

Cancelled	Sending the message was cancelled.
Failed	Sending the message failed.
Sent to node	Sending the message completed.
Waiting to send	The gateway is waiting to send the message to the WSN node.

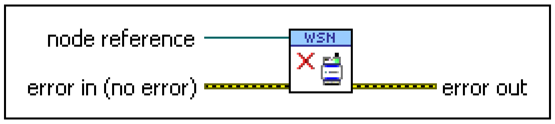
error out contains error information. This output provides standard error out functionality.

WSN Close Node VI

Owning Palette: WSN Host API VIs

Requires: LabVIEW WSN Module

Closes the reference to the specified WSN node.



node reference is the reference to the WSN node.

error in describes error conditions that occur before this node runs. This input provides standard error in functionality.

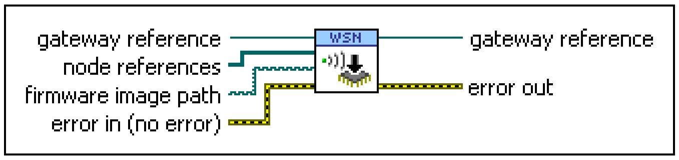
error out contains error information. This output provides standard error out functionality.

WSN Deploy Firmware VI

Owning Palette: WSN Host API VIs

Requires: LabVIEW WSN Module

Downloads the application to the WSN nodes. When you build a WSN application, you create a new firmware image and deploy that firmware image in a `.pkg` file to the WSN node. You can use the National Instruments Measurement & Automation Explorer (MAX) to restore the firmware image.



gateway reference specifies the reference to the WSN gateway.

node references includes references to all WSN nodes in the network. This VI sequentially downloads the application specified in **firmware image path** to each WSN node.

firmware image path is the path to the `.pkg` file that contains the firmware image.

error in describes error conditions that occur before this node runs. This input provides standard error in functionality.

gateway reference returns the reference to the WSN gateway.

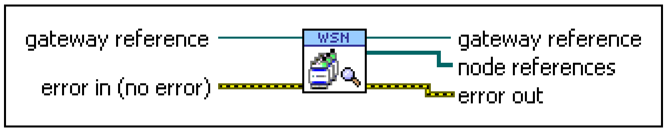
error out contains error information. This output provides standard error out functionality.

WSN Discover All Nodes VI

Owning Palette: WSN Host API VIs

Requires: LabVIEW WSN Module

Returns information about the WSN network from the WSN gateway.



gateway reference specifies the reference to the WSN gateway.

error in describes error conditions that occur before this node runs. This input provides standard error in functionality.

gateway reference returns the reference to the WSN gateway.

node references returns a reference for each WSN node added to the WSN gateway in the National Instruments Measurement & Automation Explorer (MAX).

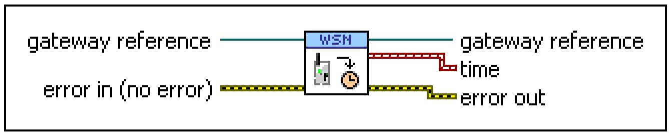
error out contains error information. This output provides standard error out functionality.

WSN Gateway Time VI

Owning Palette: WSN Host API VIs

Requires: LabVIEW WSN Module

Returns the current time from the WSN gateway.



gateway reference specifies the reference to the WSN gateway.

error in describes error conditions that occur before this node runs. This input provides standard error in functionality.

gateway reference returns the reference to the WSN gateway.

time returns the current timestamp from the WSN gateway. WSN gateways return timestamps in terms of time elapsed since 12:00 a.m., Friday, January 1, 1904, Universal Time [01-01-1904 00:00:00].

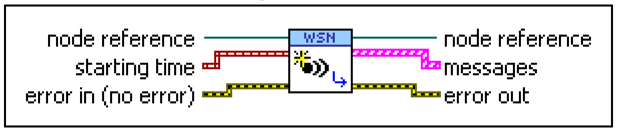
error out contains error information. This output provides standard error out functionality.

WSN Get New Messages VI

Owning Palette: WSN Host API VIs

Requires: LabVIEW WSN Module

Returns all new messages received by the WSN gateway from a WSN node since a specified starting time.



node reference is the reference to the WSN node.

starting time is the time at which to start checking for new messages from the WSN node.

error in describes error conditions that occur before this node runs. This input provides standard error in functionality.

node reference returns the reference to the specified WSN node.

messages returns the message type (user or debug message), timestamp, and message string for each message the gateway receives.

Note WSN nodes return timestamps in terms of time elapsed since 12:00 a.m., Thursday, January 1, 1970, Universal Time [01-01-1970 00:00:00]. WSN gateways return timestamps in terms of time elapsed since 12:00 a.m., Friday, January 1, 1904, Universal Time [01-01-1904 00:00:00].

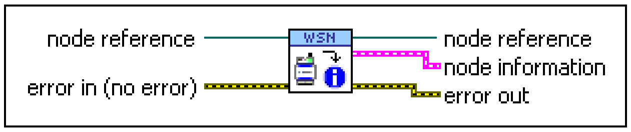
error out contains error information. This output provides standard error out functionality.

WSN Get Node Info VI

Owning Palette: WSN Host API VIs

Requires: LabVIEW WSN Module

Returns information about the specified WSN node from the gateway.



node reference is the reference to the WSN node.

error in describes error conditions that occur before this node runs. This input provides standard error in functionality.

node reference returns the reference to the specified WSN node.

node information returns information about the specified WSN node. This parameter returns the following information about the WSN node:

- Serial number for the WSN node
- Wireless ID for the WSN node
- WSN node type
- Status of the firmware update (not started, in progress, error, or done)
- Percentage of the firmware update that has completed
- Current version of the firmware
- State of the battery (no battery, critical low, low, or OK)
- Time at which the last packet was received from the WSN node
- Network link quality (no signal, poor, fair, good, or excellent)
- Power supply type (wall mounted or battery)
- Network mode (router or end device)

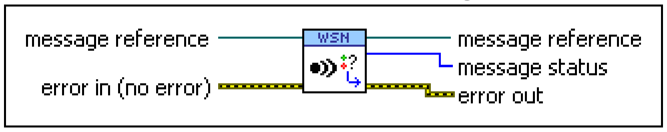
error out contains error information. This output provides standard error out functionality.

WSN Message Status VI

Owning Palette: WSN Host API VIs

Requires: LabVIEW WSN Module

Returns the status of a message sent to a WSN node.



message reference is the reference to the message.

error in describes error conditions that occur before this node runs. This input provides standard error in functionality.

message reference returns a reference to the message.

message status returns the status of the message. The output can return the following values.

Cancelled	Sending the message was cancelled.
Failed	Sending the message failed.
Sent to node	Sending the message completed.
Waiting to send	The gateway is waiting to send the message to the WSN node.

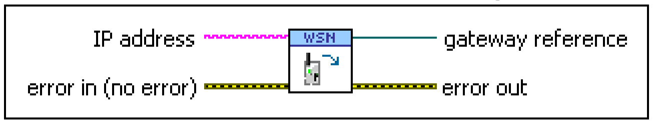
error out contains error information. This output provides standard error out functionality.

WSN Open Gateway VI

Owning Palette: WSN Host API VIs

Requires: LabVIEW WSN Module

Creates a reference to the WSN gateway.



IP address is the IP address or name of the WSN gateway. You can get the IP address and name of the WSN gateway from the National Instruments Measurement & Automation Explorer (MAX). To receive user messages from a VI running on a WSN RT gateway, use `localhost` or `127.0.0.1` as the IP address of the WSN RT gateway.

error in describes error conditions that occur before this node runs. This input provides standard error in functionality.

gateway reference returns the reference to the WSN gateway.

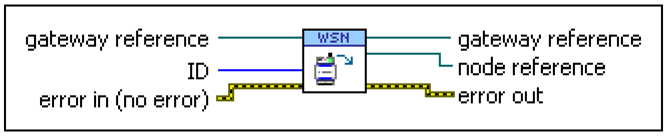
error out contains error information. This output provides standard error out functionality.

WSN Open Node Reference VI

Owning Palette: WSN Host API VIs

Requires: LabVIEW WSN Module

Creates a reference to the specified WSN node.



gateway reference specifies the reference to the WSN gateway.

ID specifies the unique identifier that specifies the WSN node for which you want to create a reference. You can get the unique identifier for the WSN node from the National Instruments Measurement & Automation Explorer (MAX).

error in describes error conditions that occur before this node runs. This input provides standard error in functionality.

gateway reference returns the reference to the WSN gateway.

node reference returns the reference to the specified WSN node.

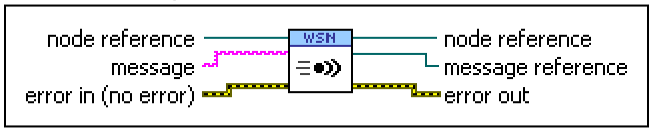
error out contains error information. This output provides standard error out functionality.

WSN Send Message VI

Owning Palette: WSN Host API VIs

Requires: LabVIEW WSN Module

Sends a user message to a WSN node. This VI can run concurrently while other VIs are running.



node reference is the reference to the WSN node.

message is the user message to send to the specified WSN node.

error in describes error conditions that occur before this node runs. This input provides standard error in functionality.

node reference returns the reference to the specified WSN node.

message reference returns a reference to the message.

error out contains error information. This output provides standard error out functionality.

WSN VIs

Owning Palette: Wireless Sensor Network Module VIs

Requires: LabVIEW WSN Module. This topic might not match its corresponding palette in LabVIEW depending on your operating system, licensed product(s), and target.

Use the WSN VIs to create applications for LabVIEW WSN targets.

Subpalette	Description
Elemental I/O Functions	Use the Elemental I/O functions to perform Ele mental I/O operations on WSN targets.
User Memory VIs	Use the User Memory VIs to access data in the u ser memory sector.
User RAM VIs	Use the User RAM VIs to access data in the user RAM on WSN nodes.

Additional Subpalettes	Description
Configuration & Communication VIs	Use the Configuration & Communication VIs to c onfigure and communicate with LabVIEW WSN n odes.

Serial VIs	Use the Serial VIs to configure and communicat e with the NI WSN-9230/9231 serial nodes.
User Calibration VIs	Use the User Calibration VIs to calibrate WSN str ain nodes.

Elemental I/O Functions

Owning Palette: Wireless Sensor Network Module VIs, WSN VIs

Requires: LabVIEW WSN Module. This topic might not match its corresponding palette in LabVIEW depending on your operating system, licensed product(s), and target.

Use the Elemental I/O functions to perform Elemental I/O operations on WSN targets.

Palette Object	Description
Elemental IO Node	Performs specific Elemental I/O operations on WSN targets. You can configure the Elemental I/ O Node with one or more Elemental I/O items. E lemental I/O resources can perform both read a nd write operations.
Elemental IO Property Node	Gets or sets one or more properties for an Eleme ntal I/O item in the project. The I/O resources av ailable and the associated properties vary by tar get and configuration.

Elemental IO Node

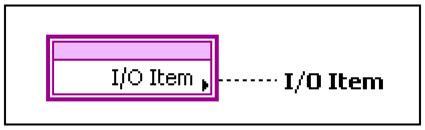
Owning Palette: Elemental I/O Functions

Requires: LabVIEW WSN Module

Performs specific Elemental I/O operations on WSN targets. You can configure the Elemental I/O Node with one or more Elemental I/O items. Elemental I/O resources can perform both read and write operations.

To add Elemental I/O items to an Elemental I/O Node, right-click the **I/O Item** section and select **Add Element** from the shortcut menu. You also can expand or contract the Elemental I/O Node by clicking the upper or lower edge of the node with the Positioning tool and dragging the edge up or down.

Details



I/O Item is the data read from or written to the Elemental I/O item you specify. For Elemental I/O items that support the read operation, this parameter is an indicator. For I/O items that do not support the read operation, this parameter is a control. For Elemental I/O items that support reading and writing, this parameter can be either a control or an indicator.

Elemental IO Node Details

When you right-click the **I/O Item** terminal of the Elemental I/O Node and select **Select Elemental I/O** from the shortcut menu, LabVIEW displays the Elemental I/O items that appear in the Project Explorer window below the same WSN target as the VI you are currently editing. You also can right-click the **I/O Item** terminal and select **Add New Elemental I/O** from the shortcut menu to add new Elemental I/O items under the WSN target.

Digital input and output (DIO) resources can perform both read and write operations. To change the operation of an Elemental I/O item, right-click the element and select **Change to Read** or **Change to Write** from the shortcut menu.

When you configure an Elemental I/O Node with multiple I/O items, the execution timing of the I/O items depends on the specific hardware devices and I/O items you use. Some Elemental I/O operations execute in parallel, while others execute sequentially.

Tip Right-click an element in the Elemental I/O Node and select **Find Elemental I/O in Project** from the shortcut menu to highlight the Elemental I/O item in the Project Explorer window.

Elemental IO Property Node

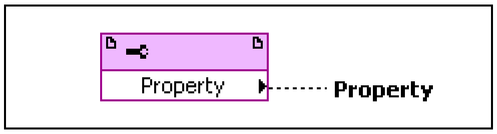
Owning Palette: Elemental I/O Functions

Requires: LabVIEW WSN Module

Gets or sets one or more properties for an Elemental I/O item in the project. The I/O resources available and the associated properties vary by target and configuration.

To select a property, first configure the Elemental I/O Property Node with an Elemental I/O item. Then, click the **Property** section of the Elemental I/O Property Node and select a property from the shortcut menu. To add additional properties to an Elemental I/O Property Node, right-click the **Property** section and select **Add Element** from the shortcut menu. You also can expand or contract the Elemental I/O Property Node by clicking the upper or lower edge of the node with the Positioning tool and dragging the edge up or down.

Details



POLY **Property** is the value of the Elemental I/O property you specify. To specify whether this parameter is a control or an indicator, right-click an element in the section of the Elemental I/O Property Node and select **Change to Read** or **Change to Write** from the shortcut menu.

Elemental IO Property Node Details

Support for use of the Elemental I/O Property Node varies by target, Elemental I/O item, and property. Not all Elemental I/O items have properties you can specify with the Elemental I/O Property Node.

When you expand the Elemental I/O Property Node, LabVIEW displays a **Property** parameter for each element of the Node.

Tip Right-click the Elemental I/O Property Node and select **Find Item in Project** from the shortcut menu to highlight the Elemental I/O item in the Project Explorer window.

User Memory VIs

Owning Palette: WSN VIs

Requires: LabVIEW WSN Module. This topic might not match its corresponding palette in LabVIEW depending on your operating system, licensed product(s), and target.

Use the User Memory VIs to access data in the user memory sector.

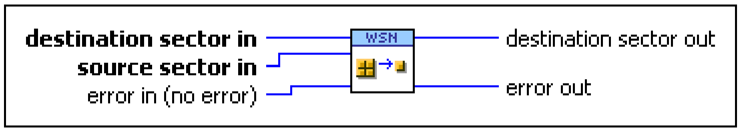
Palette Object	Description
Copy Sector	Copies all data from the source sector to the des tination sector in user memory.
Erase Sector	Erases the specified user memory sector. This VI erases one 4 KB user memory sector.
Get Number of Erase Cycles Remaining	Returns the number of erase operations remaini ng for a user memory sector.
Is Sector Empty	Checks if a user memory sector is empty. This VI returns TRUE if **sector in** is empty.
Read from Sector	Reads data from a user memory sector starting at the specified byte offset. The offset is advanc ed by the length of the data.
Write to Sector	Writes data into a user memory sector starting a t the specified byte offset. The offset is advance d by the length of the data. After writing the dat a, the data is read to verify that the write succee ded. An error code is returned if the write or veri fy fails.

Copy Sector VI

Owning Palette: User Memory VIs

Requires: LabVIEW WSN Module

Copies all data from the source sector to the destination sector in user memory. The destination sector must be empty before the VI can copy data to it. If the destination sector is not empty, the VI erases the destination sector before copying the data. The VI erases one 4 KB sector at a time.



destination sector in specifies the user memory sector to copy **source sector in** to.

source sector in specifies the user memory sector to copy.

error in (no error) describes error conditions that occur before this VI runs. The default is no error. If an error occurred before this VI runs, the VI passes the **error in** value to **error out**. This VI runs normally only if no error occurred before this VI runs. If an error occurs while this VI runs, it runs normally and sets its own error status in **error out**. Use **error in** and **error out** to check errors and to specify execution order by wiring **error out** from one node to **error in** of the next node.

destination sector out returns the address of the user memory sector to which the data was copied.

error out contains an error code. If **error in** indicates that an error occurred before this VI ran, **error out** contains the same value. Otherwise it contains the error code that this VI produces. The value will be 0 if no error occurred, and non-zero if an error occurred.

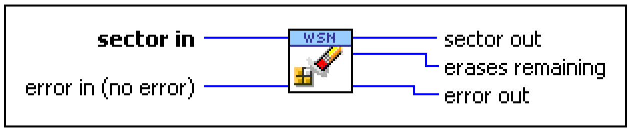
This VI can return the LabVIEW WSN Module error codes.

Erase Sector VI

Owning Palette: User Memory VIs

Requires: LabVIEW WSN Module

Erases the specified user memory sector. This VI erases one 4 KB user memory sector. You must erase user memory sectors that have been previously written to before writing data to the same location again. The erase is successful only if the sector is not empty. **error out** returns an error code if erasing the user memory sector fails.



sector in specifies the user memory sector.

error in (no error) describes error conditions that occur before this VI runs. The default is no error. If an error occurred before this VI runs, the VI passes the **error in** value to **error out**. This VI runs normally only if no error occurred before this VI runs. If an error occurs while this VI runs, it runs normally and sets its own error status in **error out**. Use **error in** and **error out** to check errors and to specify execution order by wiring **error out** from one node to **error in** of the next node.

sector out returns the address of the specified user memory sector.

erases remaining returns the remaining number of erase operations for the user memory sector. If **erases remaining** returns a negative value, the maximum number of erase operations has been exceeded, and that user memory sector might not function as expected.

error out contains an error code. If **error in** indicates that an error occurred before this VI ran, **error out** contains the same value. Otherwise it contains the error code that this VI produces. The value will be 0 if no error occurred, and non-zero if an error occurred.

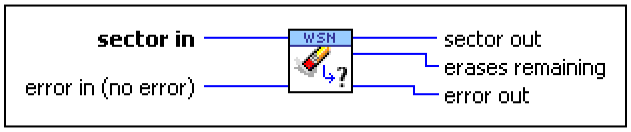
This VI can return the LabVIEW WSN Module error codes.

Get Number of Erase Cycles Remaining VI

Owning Palette: User Memory VIs

Requires: LabVIEW WSN Module

Returns the number of erase operations remaining for a user memory sector.



sector in specifies the user memory sector.

error in (no error) describes error conditions that occur before this VI runs. The default is no error. If an error occurred before this VI runs, the VI passes the **error in** value to **error out**. This VI runs normally only if no error occurred before this VI runs. If an error occurs while this VI runs, it runs normally and sets its own error status in **error out**. Use **error in** and **error out** to check errors and to specify execution order by wiring **error out** from one node to **error in** of the next node.

sector out returns the address of the specified user memory sector.

erases remaining returns the remaining number of erase operations for the user memory sector. If **erases remaining** returns a negative value, the maximum number of erase operations has been exceeded, and that user memory sector might not function as expected.

error out contains an error code. If **error in** indicates that an error occurred before this VI ran, **error out** contains the same value. Otherwise it contains the error code that this VI produces. The value will be 0 if no error occurred, and non-zero if an error occurred.

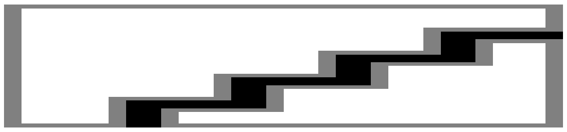
This VI can return the LabVIEW WSN Module error codes.

Is Sector Empty VI

Owning Palette: User Memory VIs

Requires: LabVIEW WSN Module

Checks if a user memory sector is empty. This VI returns TRUE if **sector in** is empty.



sector in specifies the user memory sector.

error in (no error) describes error conditions that occur before this VI runs. The default is no error. If an error occurred before this VI runs, the VI passes the **error in** value to **error out**. This VI runs normally only if no error occurred before this VI runs. If an error occurs while this VI runs, it runs normally and sets its own error status in **error out**. Use **error in** and **error out** to check errors and to specify execution order by wiring **error out** from one node to **error in** of the next node.

sector out returns the address of the specified user memory sector.

empty? is TRUE if the user memory sector is empty.

error out contains an error code. If **error in** indicates that an error occurred before this VI ran, **error out** contains the same value. Otherwise it contains the error code that this VI produces. The value will be 0 if no error occurred, and non-zero if an error occurred.

This VI can return the LabVIEW WSN Module error codes.

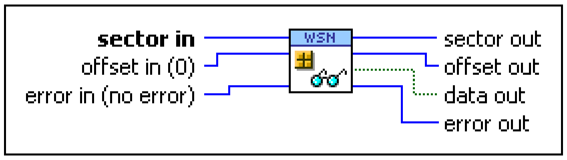
Read from Sector VI

Owning Palette: User Memory VIs

Requires: LabVIEW WSN Module

Reads data from a user memory sector starting at the specified byte offset. The offset is advanced by the length of the data. You must manually select the polymorphic instance you want to use.

Read From Sector Boolean



U8 **sector in** specifies the user memory sector.

U16 **offset in (0)** specifies the byte offset to start reading data from in the user memory sector.

I16 **error in (no error)** describes error conditions that occur before this VI runs. The default is no error. If an error occurred before this VI runs, the VI passes the **error in** value to **error out**. This VI runs normally only if no error occurred before this VI runs. If an error occurs while this VI runs, it runs normally and sets its own error status in **error out**. Use **error in** and **error out** to check errors and to specify execution order by wiring **error out** from one node to **error in** of the next node.

U8 **sector out** returns the address of the specified user memory sector.

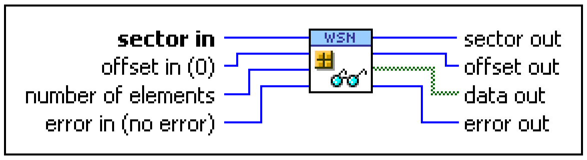
U16 **offset out** returns the byte offset after reading the data.

TF **data out** returns the data read from the user memory sector.

I16 **error out** contains an error code. If **error in** indicates that an error occurred before this VI ran, **error out** contains the same value. Otherwise it contains the error code that this VI produces. The value will be 0 if no error occurred, and non-zero if an error occurred.

This VI can return the LabVIEW WSN Module error codes.

Read from Sector Boolean Array



U8 **sector in** specifies the user memory sector.

U16 **offset in (0)** specifies the byte offset to start reading data from in the user memory sector.

U16 **number of elements** specifies the number of elements in the array to read.

I16 **error in (no error)** describes error conditions that occur before this VI runs. The default is no error. If an error occurred before this VI runs, the VI passes the **error in** value to **error out**. This VI runs normally only if no error occurred before this VI runs. If an error occurs while this VI runs, it runs normally and sets its own error status in **error out**. Use **error in** and **error out** to check errors and to specify execution order by wiring **error out** from one node to **error in** of the next node.

U8 **sector out** returns the address of the specified user memory sector.

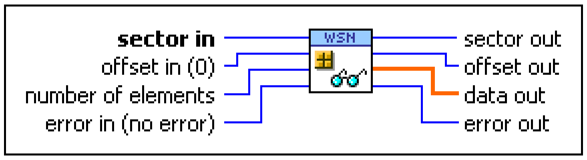
U16 **offset out** returns the byte offset after reading the data.

[TF] **data out** returns the data read from the user memory sector.

I16 **error out** contains an error code. If **error in** indicates that an error occurred before this VI ran, **error out** contains the same value. Otherwise it contains the error code that this VI produces. The value will be 0 if no error occurred, and non-zero if an error occurred.

This VI can return the LabVIEW WSN Module error codes.

Read from Sector Complex Single Array



U8 **sector in** specifies the user memory sector.

U16 **offset in (0)** specifies the byte offset to start reading data from in the user memory sector.

U16 **number of elements** specifies the number of elements in the array to read.

I16 **error in (no error)** describes error conditions that occur before this VI runs. The default is no error. If an error occurred before this VI runs, the VI passes the **error in** value to **error out**. This VI runs normally only if no error occurred before this VI runs. If an error occurs while this VI runs, it runs normally and sets its own error status in **error out**. Use **error in** and **error out** to check errors and to specify execution order by wiring **error out** from one node to **error in** of the next node.

sector out returns the address of the specified user memory sector.

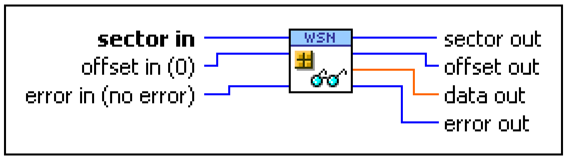
offset out returns the byte offset after reading the data.

data out returns the data read from the user memory sector.

error out contains an error code. If **error in** indicates that an error occurred before this VI ran, **error out** contains the same value. Otherwise it contains the error code that this VI produces. The value will be 0 if no error occurred, and non-zero if an error occurred.

This VI can return the LabVIEW WSN Module error codes.

Read from Sector Complex Single



sector in specifies the user memory sector.

offset in (0) specifies the byte offset to start reading data from in the user memory sector.

error in (no error) describes error conditions that occur before this VI runs. The default is no error. If an error occurred before this VI runs, the VI passes the **error in** value to **error out**. This VI runs normally only if no error occurred before this VI runs. If an error occurs while this VI runs, it runs normally and sets its own error status in **error out**. Use **error in** and **error out** to check errors and to specify execution order by wiring **error out** from one node to **error in** of the next node.

sector out returns the address of the specified user memory sector.

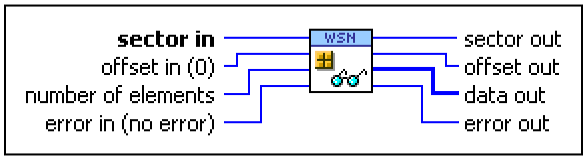
offset out returns the byte offset after reading the data.

data out returns the data read from the user memory sector.

error out contains an error code. If **error in** indicates that an error occurred before this VI ran, **error out** contains the same value. Otherwise it contains the error code that this VI produces. The value will be 0 if no error occurred, and non-zero if an error occurred.

This VI can return the LabVIEW WSN Module error codes.

Read from Sector Int8 Array



U8 **sector in** specifies the user memory sector.

U16 **offset in (0)** specifies the byte offset to start reading data from in the user memory sector.

U16 **number of elements** specifies the number of elements in the array to read.

I16 **error in (no error)** describes error conditions that occur before this VI runs. The default is no error. If an error occurred before this VI runs, the VI passes the **error in** value to **error out**. This VI runs normally only if no error occurred before this VI runs. If an error occurs while this VI runs, it runs normally and sets its own error status in **error out**. Use **error in** and **error out** to check errors and to specify execution order by wiring **error out** from one node to **error in** of the next node.

U8 **sector out** returns the address of the specified user memory sector.

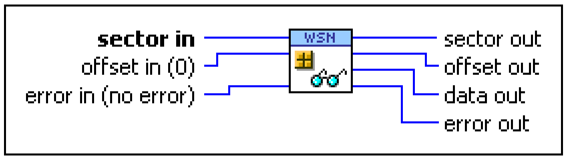
U16 **offset out** returns the byte offset after reading the data.

[I8] **data out** returns the data read from the user memory sector.

I16 **error out** contains an error code. If **error in** indicates that an error occurred before this VI ran, **error out** contains the same value. Otherwise it contains the error code that this VI produces. The value will be 0 if no error occurred, and non-zero if an error occurred.

This VI can return the LabVIEW WSN Module error codes.

Read from Sector Int8



U8 **sector in** specifies the user memory sector.

U16 **offset in (0)** specifies the byte offset to start reading data from in the user memory sector.

I16 **error in (no error)** describes error conditions that occur before this VI runs. The default is no error. If an error occurred before this VI runs, the VI passes the **error in** value to **error out**. This VI runs normally only if no error occurred before this VI runs. If an error occurs while this VI runs, it runs normally and sets its own error status in **error out**. Use **error in** and **error out** to check errors and to specify execution order by wiring **error out** from one node to **error in** of the next node.

U8 **sector out** returns the address of the specified user memory sector.

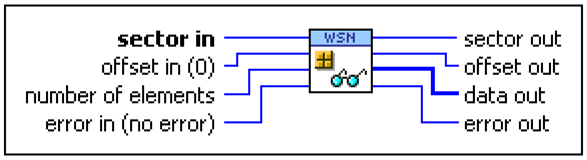
U16 **offset out** returns the byte offset after reading the data.

I8 **data out** returns the data read from the user memory sector.

I16 **error out** contains an error code. If **error in** indicates that an error occurred before this VI ran, **error out** contains the same value. Otherwise it contains the error code that this VI produces. The value will be 0 if no error occurred, and non-zero if an error occurred.

This VI can return the LabVIEW WSN Module error codes.

Read from Sector Int16 Array



U8 **sector in** specifies the user memory sector.

U16 **offset in (0)** specifies the byte offset to start reading data from in the user memory sector.

U16 **number of elements** specifies the number of elements in the array to read.

I16 **error in (no error)** describes error conditions that occur before this VI runs. The default is no error. If an error occurred before this VI runs, the VI passes the **error in** value to **error out**. This VI runs normally only if no error occurred before this VI runs. If an error occurs while this VI runs, it runs normally and sets its own error status in **error out**. Use **error in** and **error out** to check errors and to specify execution order by wiring **error out** from one node to **error in** of the next node.

U8 **sector out** returns the address of the specified user memory sector.

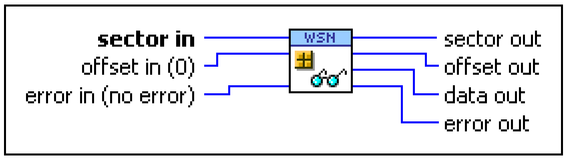
U16 **offset out** returns the byte offset after reading the data.

[I16] **data out** returns the data read from the user memory sector.

I16 **error out** contains an error code. If **error in** indicates that an error occurred before this VI ran, **error out** contains the same value. Otherwise it contains the error code that this VI produces. The value will be 0 if no error occurred, and non-zero if an error occurred.

This VI can return the LabVIEW WSN Module error codes.

Read from Sector Int16



U8 **sector in** specifies the user memory sector.

U16 **offset in (0)** specifies the byte offset to start reading data from in the user memory sector.

I16 **error in (no error)** describes error conditions that occur before this VI runs. The default is no error. If an error occurred before this VI runs, the VI passes the **error in** value to **error out**. This VI runs normally only if no error occurred before this VI runs. If an error occurs while this VI runs, it runs normally and sets its own error status in **error out**. Use **error in** and **error out** to check errors and to specify execution order by wiring **error out** from one node to **error in** of the next node.

U8 **sector out** returns the address of the specified user memory sector.

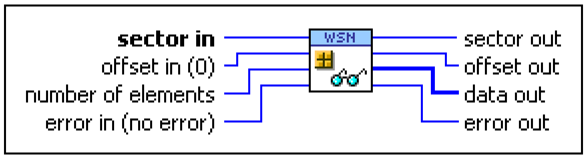
U16 **offset out** returns the byte offset after reading the data.

I16 **data out** returns the data read from the user memory sector.

I16 **error out** contains an error code. If **error in** indicates that an error occurred before this VI ran, **error out** contains the same value. Otherwise it contains the error code that this VI produces. The value will be 0 if no error occurred, and non-zero if an error occurred.

This VI can return the LabVIEW WSN Module error codes.

Read from Sector Int32 Array



U8 **sector in** specifies the user memory sector.

U16 **offset in (0)** specifies the byte offset to start reading data from in the user memory sector.

U16 **number of elements** specifies the number of elements in the array to read.

I16 **error in (no error)** describes error conditions that occur before this VI runs. The default is no error. If an error occurred before this VI runs, the VI passes the **error in** value to **error out**. This VI runs normally only if no error occurred before this VI runs. If an error occurs while this VI runs, it runs normally and sets its own error status in **error out**. Use **error in** and **error out** to check errors and to specify execution order by wiring **error out** from one node to **error in** of the next node.

U8 **sector out** returns the address of the specified user memory sector.

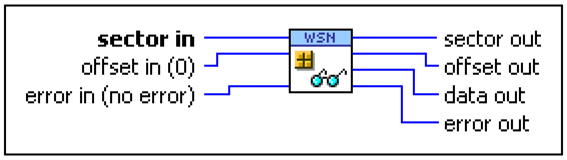
U16 **offset out** returns the byte offset after reading the data.

[I32] **data out** returns the data read from the user memory sector.

I16 **error out** contains an error code. If **error in** indicates that an error occurred before this VI ran, **error out** contains the same value. Otherwise it contains the error code that this VI produces. The value will be 0 if no error occurred, and non-zero if an error occurred.

This VI can return the LabVIEW WSN Module error codes.

Read from Sector Int32



U8 **sector in** specifies the user memory sector.

U16 **offset in (0)** specifies the byte offset to start reading data from in the user memory sector.

I16 **error in (no error)** describes error conditions that occur before this VI runs. The default is no error. If an error occurred before this VI runs, the VI passes the **error in** value to **error out**. This VI runs normally only if no error occurred before this VI runs. If an error occurs while this VI runs, it runs normally and sets its own error status in **error out**. Use **error in** and **error out** to check errors and to specify execution order by wiring **error out** from one node to **error in** of the next node.

U8 **sector out** returns the address of the specified user memory sector.

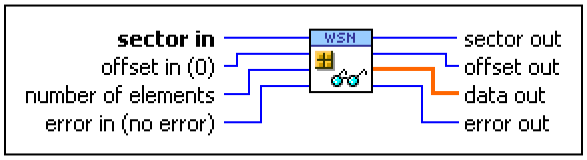
offset out returns the byte offset after reading the data.

data out returns the data read from the user memory sector.

error out contains an error code. If **error in** indicates that an error occurred before this VI ran, **error out** contains the same value. Otherwise it contains the error code that this VI produces. The value will be 0 if no error occurred, and non-zero if an error occurred.

This VI can return the LabVIEW WSN Module error codes.

Read from Sector Single Array



sector in specifies the user memory sector.

offset in (0) specifies the byte offset to start reading data from in the user memory sector.

number of elements specifies the number of elements in the array to read.

error in (no error) describes error conditions that occur before this VI runs. The default is no error. If an error occurred before this VI runs, the VI passes the **error in** value to **error out**. This VI runs normally only if no error occurred before this VI runs. If an error occurs while this VI runs, it runs normally and sets its own error status in **error out**. Use **error in** and **error out** to check errors and to specify execution order by wiring **error out** from one node to **error in** of the next node.

sector out returns the address of the specified user memory sector.

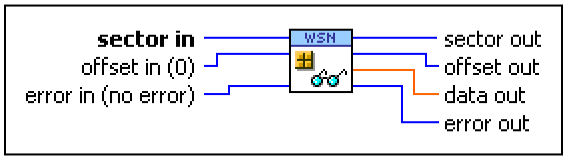
offset out returns the byte offset after reading the data.

data out returns the data read from the user memory sector.

error out contains an error code. If **error in** indicates that an error occurred before this VI ran, **error out** contains the same value. Otherwise it contains the error code that this VI produces. The value will be 0 if no error occurred, and non-zero if an error occurred.

This VI can return the LabVIEW WSN Module error codes.

Read from Sector Single



U8 **sector in** specifies the user memory sector.

U16 **offset in (0)** specifies the byte offset to start reading data from in the user memory sector.

I16 **error in (no error)** describes error conditions that occur before this VI runs. The default is no error. If an error occurred before this VI runs, the VI passes the **error in** value to **error out**. This VI runs normally only if no error occurred before this VI runs. If an error occurs while this VI runs, it runs normally and sets its own error status in **error out**. Use **error in** and **error out** to check errors and to specify execution order by wiring **error out** from one node to **error in** of the next node.

U8 **sector out** returns the address of the specified user memory sector.

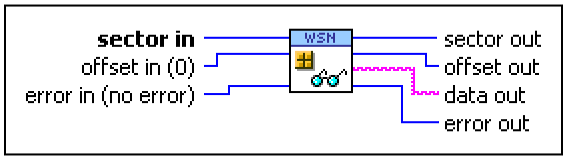
U16 **offset out** returns the byte offset after reading the data.

SGL **data out** returns the data read from the user memory sector.

I16 **error out** contains an error code. If **error in** indicates that an error occurred before this VI ran, **error out** contains the same value. Otherwise it contains the error code that this VI produces. The value will be 0 if no error occurred, and non-zero if an error occurred.

This VI can return the LabVIEW WSN Module error codes.

Read from Sector String



U8 **sector in** specifies the user memory sector.

U16 **offset in (0)** specifies the byte offset to start reading data from in the user memory sector.

I16 **error in (no error)** describes error conditions that occur before this VI runs. The default is no error. If an error occurred before this VI runs, the VI passes the **error in** value to

error out. This VI runs normally only if no error occurred before this VI runs. If an error occurs while this VI runs, it runs normally and sets its own error status in **error out**. Use **error in** and **error out** to check errors and to specify execution order by wiring **error out** from one node to **error in** of the next node.

sector out returns the address of the specified user memory sector.

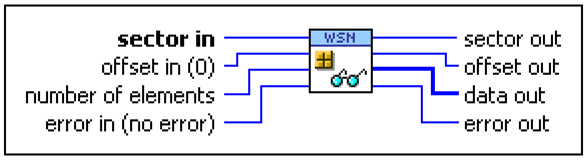
offset out returns the byte offset after reading the data.

data out returns the data read from the user memory sector.

error out contains an error code. If **error in** indicates that an error occurred before this VI ran, **error out** contains the same value. Otherwise it contains the error code that this VI produces. The value will be 0 if no error occurred, and non-zero if an error occurred.

This VI can return the LabVIEW WSN Module error codes.

Read from Sector UInt8 Array



sector in specifies the user memory sector.

offset in (0) specifies the byte offset to start reading data from in the user memory sector.

number of elements specifies the number of elements in the array to read.

error in (no error) describes error conditions that occur before this VI runs. The default is no error. If an error occurred before this VI runs, the VI passes the **error in** value to **error out**. This VI runs normally only if no error occurred before this VI runs. If an error occurs while this VI runs, it runs normally and sets its own error status in **error out**. Use **error in** and **error out** to check errors and to specify execution order by wiring **error out** from one node to **error in** of the next node.

sector out returns the address of the specified user memory sector.

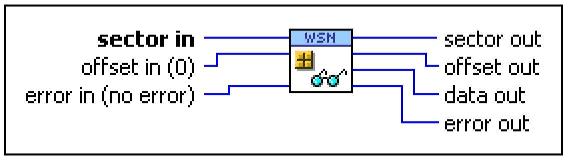
offset out returns the byte offset after reading the data.

data out returns the data read from the user memory sector.

error out contains an error code. If **error in** indicates that an error occurred before this VI ran, **error out** contains the same value. Otherwise it contains the error code that this VI produces. The value will be 0 if no error occurred, and non-zero if an error occurred.

This VI can return the LabVIEW WSN Module error codes.

Read from Sector UInt8



U8 **sector in** specifies the user memory sector.

U16 **offset in (0)** specifies the byte offset to start reading data from in the user memory sector.

I16 **error in (no error)** describes error conditions that occur before this VI runs. The default is no error. If an error occurred before this VI runs, the VI passes the **error in** value to **error out**. This VI runs normally only if no error occurred before this VI runs. If an error occurs while this VI runs, it runs normally and sets its own error status in **error out**. Use **error in** and **error out** to check errors and to specify execution order by wiring **error out** from one node to **error in** of the next node.

U8 **sector out** returns the address of the specified user memory sector.

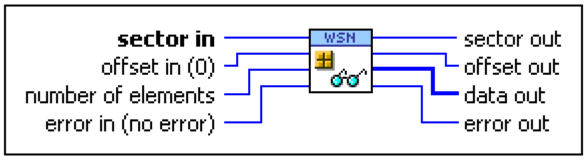
U16 **offset out** returns the byte offset after reading the data.

U8 **data out** returns the data read from the user memory sector.

I16 **error out** contains an error code. If **error in** indicates that an error occurred before this VI ran, **error out** contains the same value. Otherwise it contains the error code that this VI produces. The value will be 0 if no error occurred, and non-zero if an error occurred.

This VI can return the LabVIEW WSN Module error codes.

Read from Sector UInt16 Array



U8 **sector in** specifies the user memory sector.

U16 **offset in (0)** specifies the byte offset to start reading data from in the user memory sector.

U16 **number of elements** specifies the number of elements in the array to read.

I16 **error in (no error)** describes error conditions that occur before this VI runs. The default is no error. If an error occurred before this VI runs, the VI passes the **error in** value to **error out**. This VI runs normally only if no error occurred before this VI runs. If an error occurs while this VI runs, it runs normally and sets its own error status in **error out**. Use **error in** and **error out** to check errors and to specify execution order by wiring **error out** from one node to **error in** of the next node.

U8 **sector out** returns the address of the specified user memory sector.

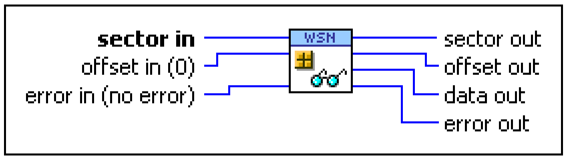
U16 **offset out** returns the byte offset after reading the data.

[U16] **data out** returns the data read from the user memory sector.

I16 **error out** contains an error code. If **error in** indicates that an error occurred before this VI ran, **error out** contains the same value. Otherwise it contains the error code that this VI produces. The value will be 0 if no error occurred, and non-zero if an error occurred.

This VI can return the LabVIEW WSN Module error codes.

Read from Sector UInt16



U8 **sector in** specifies the user memory sector.

U16 **offset in (0)** specifies the byte offset to start reading data from in the user memory sector.

I16 **error in (no error)** describes error conditions that occur before this VI runs. The default is no error. If an error occurred before this VI runs, the VI passes the **error in** value to **error out**. This VI runs normally only if no error occurred before this VI runs. If an error occurs while this VI runs, it runs normally and sets its own error status in **error out**. Use **error in** and **error out** to check errors and to specify execution order by wiring **error out** from one node to **error in** of the next node.

U8 **sector out** returns the address of the specified user memory sector.

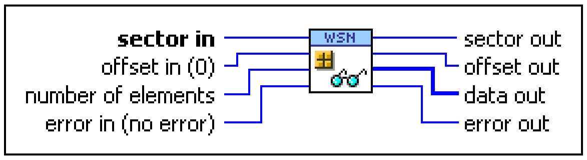
U16 **offset out** returns the byte offset after reading the data.

U16 **data out** returns the data read from the user memory sector.

error out contains an error code. If **error in** indicates that an error occurred before this VI ran, **error out** contains the same value. Otherwise it contains the error code that this VI produces. The value will be 0 if no error occurred, and non-zero if an error occurred.

This VI can return the LabVIEW WSN Module error codes.

Read from Sector UInt32 Array



sector in specifies the user memory sector.

offset in (0) specifies the byte offset to start reading data from in the user memory sector.

number of elements specifies the number of elements in the array to read.

error in (no error) describes error conditions that occur before this VI runs. The default is no error. If an error occurred before this VI runs, the VI passes the **error in** value to **error out**. This VI runs normally only if no error occurred before this VI runs. If an error occurs while this VI runs, it runs normally and sets its own error status in **error out**. Use **error in** and **error out** to check errors and to specify execution order by wiring **error out** from one node to **error in** of the next node.

sector out returns the address of the specified user memory sector.

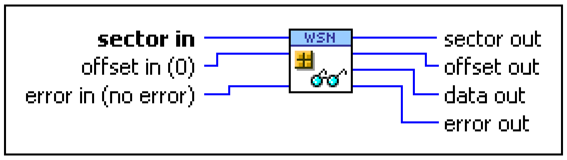
offset out returns the byte offset after reading the data.

data out returns the data read from the user memory sector.

error out contains an error code. If **error in** indicates that an error occurred before this VI ran, **error out** contains the same value. Otherwise it contains the error code that this VI produces. The value will be 0 if no error occurred, and non-zero if an error occurred.

This VI can return the LabVIEW WSN Module error codes.

Read from Sector UInt32



U8 **sector in** specifies the user memory sector.

U16 **offset in (0)** specifies the byte offset to start reading data from in the user memory sector.

I16 **error in (no error)** describes error conditions that occur before this VI runs. The default is no error. If an error occurred before this VI runs, the VI passes the **error in** value to **error out**. This VI runs normally only if no error occurred before this VI runs. If an error occurs while this VI runs, it runs normally and sets its own error status in **error out**. Use **error in** and **error out** to check errors and to specify execution order by wiring **error out** from one node to **error in** of the next node.

U8 **sector out** returns the address of the specified user memory sector.

U16 **offset out** returns the byte offset after reading the data.

U32 **data out** returns the data read from the user memory sector.

I16 **error out** contains an error code. If **error in** indicates that an error occurred before this VI ran, **error out** contains the same value. Otherwise it contains the error code that this VI produces. The value will be 0 if no error occurred, and non-zero if an error occurred.

This VI can return the LabVIEW WSN Module error codes.

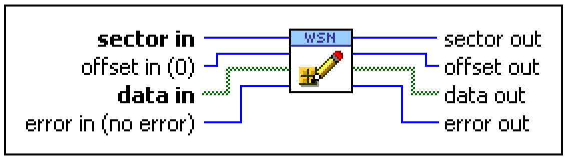
Write to Sector VI

Owning Palette: User Memory VIs

Requires: LabVIEW WSN Module

Writes data into a user memory sector starting at the specified byte offset. The offset is advanced by the length of the data. After writing the data, the data is read to verify that the write succeeded. An error code is returned if the write or verify fails. You must manually select the polymorphic instance you want to use.
Details

Write to Sector Boolean Array



U8 **sector in** specifies the user memory sector.

U16 **offset in (0)** specifies the byte offset to start writing data to in the user memory sector.

[TF] **data in** contains the data you want write to the user memory sector.

I16 **error in (no error)** describes error conditions that occur before this VI runs. The default is no error. If an error occurred before this VI runs, the VI passes the **error in** value to **error out**. This VI runs normally only if no error occurred before this VI runs. If an error occurs while this VI runs, it runs normally and sets its own error status in **error out**. Use **error in** and **error out** to check errors and to specify execution order by wiring **error out** from one node to **error in** of the next node.

U8 **sector out** returns the address of the specified user memory sector.

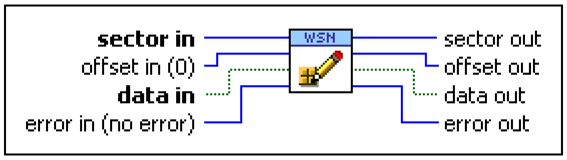
U16 **offset out** returns the byte offset after reading the data.

[TF] **data out** returns the data written to the user memory sector.

I16 **error out** contains an error code. If **error in** indicates that an error occurred before this VI ran, **error out** contains the same value. Otherwise it contains the error code that this VI produces. The value will be 0 if no error occurred, and non-zero if an error occurred.

This VI can return the LabVIEW WSN Module error codes.

Write to Sector Boolean



U8 **sector in** specifies the user memory sector.

U16 **offset in (0)** specifies the byte offset to start writing data to in the user memory sector.

TF **data in** contains the data you want write to the user memory sector.

error in (no error) describes error conditions that occur before this VI runs. The default is no error. If an error occurred before this VI runs, the VI passes the **error in** value to **error out**. This VI runs normally only if no error occurred before this VI runs. If an error occurs while this VI runs, it runs normally and sets its own error status in **error out**. Use **error in** and **error out** to check errors and to specify execution order by wiring **error out** from one node to **error in** of the next node.

sector out returns the address of the specified user memory sector.

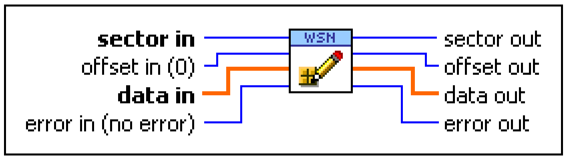
offset out returns the byte offset after reading the data.

data out returns the data written to the user memory sector.

error out contains an error code. If **error in** indicates that an error occurred before this VI ran, **error out** contains the same value. Otherwise it contains the error code that this VI produces. The value will be 0 if no error occurred, and non-zero if an error occurred.

This VI can return the LabVIEW WSN Module error codes.

Write to Sector Complex Single Array



sector in specifies the user memory sector.

offset in (0) specifies the byte offset to start writing data to in the user memory sector.

data in contains the data you want write to the user memory sector.

error in (no error) describes error conditions that occur before this VI runs. The default is no error. If an error occurred before this VI runs, the VI passes the **error in** value to **error out**. This VI runs normally only if no error occurred before this VI runs. If an error occurs while this VI runs, it runs normally and sets its own error status in **error out**. Use **error in** and **error out** to check errors and to specify execution order by wiring **error out** from one node to **error in** of the next node.

sector out returns the address of the specified user memory sector.

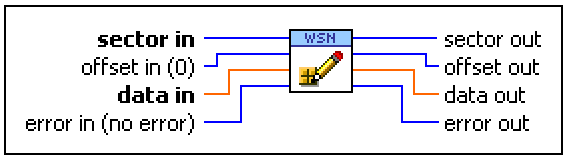
offset out returns the byte offset after reading the data.

data out returns the data written to the user memory sector.

error out contains an error code. If **error in** indicates that an error occurred before this VI ran, **error out** contains the same value. Otherwise it contains the error code that this VI produces. The value will be 0 if no error occurred, and non-zero if an error occurred.

This VI can return the LabVIEW WSN Module error codes.

Write to Sector Complex Single



sector in specifies the user memory sector.

offset in (0) specifies the byte offset to start writing data to in the user memory sector.

data in contains the data you want write to the user memory sector.

error in (no error) describes error conditions that occur before this VI runs. The default is no error. If an error occurred before this VI runs, the VI passes the **error in** value to **error out**. This VI runs normally only if no error occurred before this VI runs. If an error occurs while this VI runs, it runs normally and sets its own error status in **error out**. Use **error in** and **error out** to check errors and to specify execution order by wiring **error out** from one node to **error in** of the next node.

sector out returns the address of the specified user memory sector.

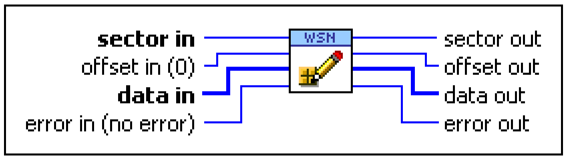
offset out returns the byte offset after reading the data.

data out returns the data written to the user memory sector.

error out contains an error code. If **error in** indicates that an error occurred before this VI ran, **error out** contains the same value. Otherwise it contains the error code that this VI produces. The value will be 0 if no error occurred, and non-zero if an error occurred.

This VI can return the LabVIEW WSN Module error codes.

Write to Sector Int8 Array



U8 **sector in** specifies the user memory sector.

U16 **offset in (0)** specifies the byte offset to start writing data to in the user memory sector.

[I8] **data in** contains the data you want write to the user memory sector.

I16 **error in (no error)** describes error conditions that occur before this VI runs. The default is no error. If an error occurred before this VI runs, the VI passes the **error in** value to **error out**. This VI runs normally only if no error occurred before this VI runs. If an error occurs while this VI runs, it runs normally and sets its own error status in **error out**. Use **error in** and **error out** to check errors and to specify execution order by wiring **error out** from one node to **error in** of the next node.

U8 **sector out** returns the address of the specified user memory sector.

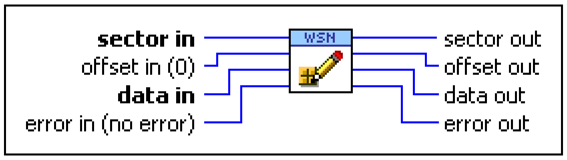
U16 **offset out** returns the byte offset after reading the data.

[I8] **data out** returns the data written to the user memory sector.

I16 **error out** contains an error code. If **error in** indicates that an error occurred before this VI ran, **error out** contains the same value. Otherwise it contains the error code that this VI produces. The value will be 0 if no error occurred, and non-zero if an error occurred.

This VI can return the LabVIEW WSN Module error codes.

Write to Sector Int8



U8 **sector in** specifies the user memory sector.

U16 **offset in (0)** specifies the byte offset to start writing data to in the user memory sector.

I8 **data in** contains the data you want write to the user memory sector.

error in (no error) describes error conditions that occur before this VI runs. The default is no error. If an error occurred before this VI runs, the VI passes the **error in** value to **error out**. This VI runs normally only if no error occurred before this VI runs. If an error occurs while this VI runs, it runs normally and sets its own error status in **error out**. Use **error in** and **error out** to check errors and to specify execution order by wiring **error out** from one node to **error in** of the next node.

sector out returns the address of the specified user memory sector.

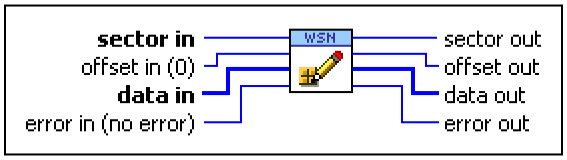
offset out returns the byte offset after reading the data.

data out returns the data written to the user memory sector.

error out contains an error code. If **error in** indicates that an error occurred before this VI ran, **error out** contains the same value. Otherwise it contains the error code that this VI produces. The value will be 0 if no error occurred, and non-zero if an error occurred.

This VI can return the LabVIEW WSN Module error codes.

Write to Sector Int16 Array



sector in specifies the user memory sector.

offset in (0) specifies the byte offset to start writing data to in the user memory sector.

data in contains the data you want write to the user memory sector.

error in (no error) describes error conditions that occur before this VI runs. The default is no error. If an error occurred before this VI runs, the VI passes the **error in** value to **error out**. This VI runs normally only if no error occurred before this VI runs. If an error occurs while this VI runs, it runs normally and sets its own error status in **error out**. Use **error in** and **error out** to check errors and to specify execution order by wiring **error out** from one node to **error in** of the next node.

sector out returns the address of the specified user memory sector.

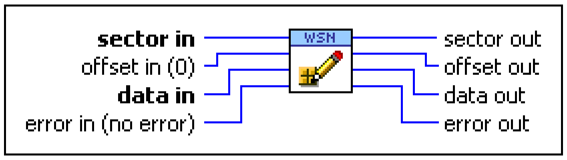
offset out returns the byte offset after reading the data.

data out returns the data written to the user memory sector.

error out contains an error code. If **error in** indicates that an error occurred before this VI ran, **error out** contains the same value. Otherwise it contains the error code that this VI produces. The value will be 0 if no error occurred, and non-zero if an error occurred.

This VI can return the LabVIEW WSN Module error codes.

Write to Sector Int16



sector in specifies the user memory sector.

offset in (0) specifies the byte offset to start writing data to in the user memory sector.

data in contains the data you want write to the user memory sector.

error in (no error) describes error conditions that occur before this VI runs. The default is no error. If an error occurred before this VI runs, the VI passes the **error in** value to **error out**. This VI runs normally only if no error occurred before this VI runs. If an error occurs while this VI runs, it runs normally and sets its own error status in **error out**. Use **error in** and **error out** to check errors and to specify execution order by wiring **error out** from one node to **error in** of the next node.

sector out returns the address of the specified user memory sector.

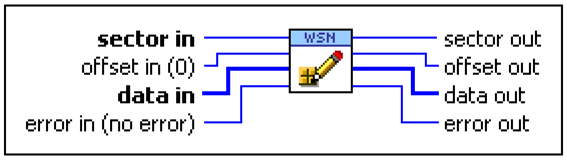
offset out returns the byte offset after reading the data.

data out returns the data written to the user memory sector.

error out contains an error code. If **error in** indicates that an error occurred before this VI ran, **error out** contains the same value. Otherwise it contains the error code that this VI produces. The value will be 0 if no error occurred, and non-zero if an error occurred.

This VI can return the LabVIEW WSN Module error codes.

Write to Sector Int32 Array



U8 **sector in** specifies the user memory sector.

U16 **offset in (0)** specifies the byte offset to start writing data to in the user memory sector.

[I32] **data in** contains the data you want write to the user memory sector.

I16 **error in (no error)** describes error conditions that occur before this VI runs. The default is no error. If an error occurred before this VI runs, the VI passes the **error in** value to **error out**. This VI runs normally only if no error occurred before this VI runs. If an error occurs while this VI runs, it runs normally and sets its own error status in **error out**. Use **error in** and **error out** to check errors and to specify execution order by wiring **error out** from one node to **error in** of the next node.

U8 **sector out** returns the address of the specified user memory sector.

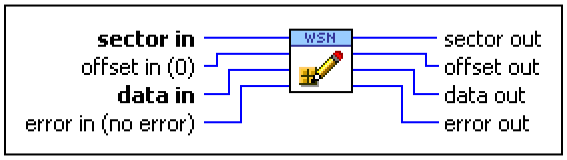
U16 **offset out** returns the byte offset after reading the data.

[I32] **data out** returns the data written to the user memory sector.

I16 **error out** contains an error code. If **error in** indicates that an error occurred before this VI ran, **error out** contains the same value. Otherwise it contains the error code that this VI produces. The value will be 0 if no error occurred, and non-zero if an error occurred.

This VI can return the LabVIEW WSN Module error codes.

Write to Sector Int32



U8 **sector in** specifies the user memory sector.

U16 **offset in (0)** specifies the byte offset to start writing data to in the user memory sector.

I32 **data in** contains the data you want write to the user memory sector.

I16 **error in (no error)** describes error conditions that occur before this VI runs. The default is no error. If an error occurred before this VI runs, the VI passes the **error in** value to **error out**. This VI runs normally only if no error occurred before this VI runs. If an error occurs while this VI runs, it runs normally and sets its own error status in **error out**. Use **error in** and **error out** to check errors and to specify execution order by wiring **error out** from one node to **error in** of the next node.

U8 **sector out** returns the address of the specified user memory sector.

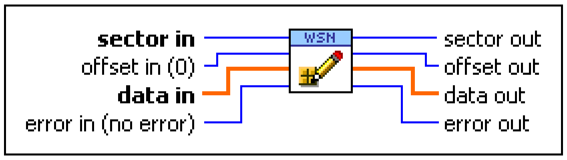
U16 **offset out** returns the byte offset after reading the data.

I32 **data out** returns the data written to the user memory sector.

I16 **error out** contains an error code. If **error in** indicates that an error occurred before this VI ran, **error out** contains the same value. Otherwise it contains the error code that this VI produces. The value will be 0 if no error occurred, and non-zero if an error occurred.

This VI can return the LabVIEW WSN Module error codes.

Write to Sector Single Array



U8 **sector in** specifies the user memory sector.

U16 **offset in (0)** specifies the byte offset to start writing data to in the user memory sector.

[SGL] **data in** contains the data you want write to the user memory sector.

I16 **error in (no error)** describes error conditions that occur before this VI runs. The default is no error. If an error occurred before this VI runs, the VI passes the **error in** value to **error out**. This VI runs normally only if no error occurred before this VI runs. If an error occurs while this VI runs, it runs normally and sets its own error status in **error out**. Use **error in** and **error out** to check errors and to specify execution order by wiring **error out** from one node to **error in** of the next node.

U8 **sector out** returns the address of the specified user memory sector.

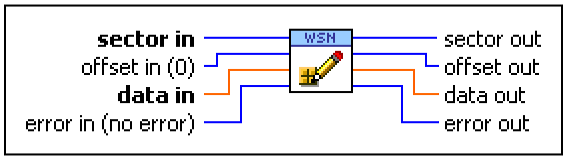
U16 **offset out** returns the byte offset after reading the data.

[SGL] **data out** returns the data written to the user memory sector.

I16 **error out** contains an error code. If **error in** indicates that an error occurred before this VI ran, **error out** contains the same value. Otherwise it contains the error code that this VI produces. The value will be 0 if no error occurred, and non-zero if an error occurred.

This VI can return the LabVIEW WSN Module error codes.

Write to Sector Single



U8 **sector in** specifies the user memory sector.

U16 **offset in (0)** specifies the byte offset to start writing data to in the user memory sector.

SGL **data in** contains the data you want write to the user memory sector.

I16 **error in (no error)** describes error conditions that occur before this VI runs. The default is no error. If an error occurred before this VI runs, the VI passes the **error in** value to **error out**. This VI runs normally only if no error occurred before this VI runs. If an error occurs while this VI runs, it runs normally and sets its own error status in **error out**. Use **error in** and **error out** to check errors and to specify execution order by wiring **error out** from one node to **error in** of the next node.

U8 **sector out** returns the address of the specified user memory sector.

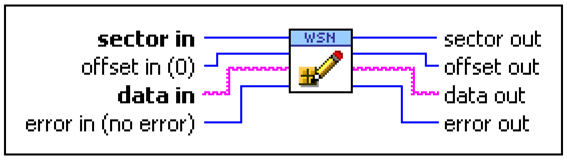
U16 **offset out** returns the byte offset after reading the data.

SGL **data out** returns the data written to the user memory sector.

I16 **error out** contains an error code. If **error in** indicates that an error occurred before this VI ran, **error out** contains the same value. Otherwise it contains the error code that this VI produces. The value will be 0 if no error occurred, and non-zero if an error occurred.

This VI can return the LabVIEW WSN Module error codes.

Write to Sector String



U8 **sector in** specifies the user memory sector.

U16 **offset in (0)** specifies the byte offset to start writing data to in the user memory sector.

abc **data in** contains the data you want write to the user memory sector.

I16 **error in (no error)** describes error conditions that occur before this VI runs. The default is no error. If an error occurred before this VI runs, the VI passes the **error in** value to **error out**. This VI runs normally only if no error occurred before this VI runs. If an error occurs while this VI runs, it runs normally and sets its own error status in **error out**. Use **error in** and **error out** to check errors and to specify execution order by wiring **error out** from one node to **error in** of the next node.

U8 **sector out** returns the address of the specified user memory sector.

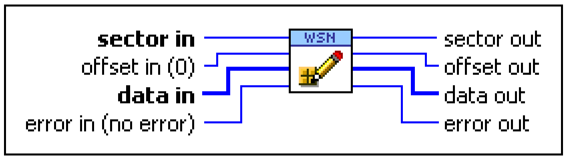
U16 **offset out** returns the byte offset after reading the data.

abc **data out** returns the data written to the user memory sector.

I16 **error out** contains an error code. If **error in** indicates that an error occurred before this VI ran, **error out** contains the same value. Otherwise it contains the error code that this VI produces. The value will be 0 if no error occurred, and non-zero if an error occurred.

This VI can return the LabVIEW WSN Module error codes.

Write to Sector UInt8 Array



U8 **sector in** specifies the user memory sector.

U16 **offset in (0)** specifies the byte offset to start writing data to in the user memory sector.

[U8] **data in** contains the data you want write to the user memory sector.

I16 **error in (no error)** describes error conditions that occur before this VI runs. The default is no error. If an error occurred before this VI runs, the VI passes the **error in** value to **error out**. This VI runs normally only if no error occurred before this VI runs. If an error occurs while this VI runs, it runs normally and sets its own error status in **error out**. Use **error in** and **error out** to check errors and to specify execution order by wiring **error out** from one node to **error in** of the next node.

U8 **sector out** returns the address of the specified user memory sector.

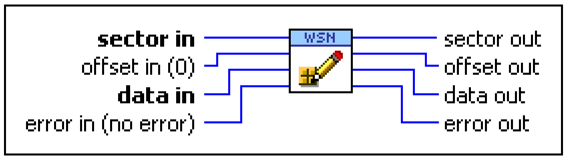
U16 **offset out** returns the byte offset after reading the data.

[U8] **data out** returns the data written to the user memory sector.

I16 **error out** contains an error code. If **error in** indicates that an error occurred before this VI ran, **error out** contains the same value. Otherwise it contains the error code that this VI produces. The value will be 0 if no error occurred, and non-zero if an error occurred.

This VI can return the LabVIEW WSN Module error codes.

Write to Sector UInt8



U8 **sector in** specifies the user memory sector.

U16 **offset in (0)** specifies the byte offset to start writing data to in the user memory sector.

U8 **data in** contains the data you want write to the user memory sector.

I16 **error in (no error)** describes error conditions that occur before this VI runs. The default is no error. If an error occurred before this VI runs, the VI passes the **error in** value to **error out**. This VI runs normally only if no error occurred before this VI runs. If an error occurs while this VI runs, it runs normally and sets its own error status in **error out**. Use **error in** and **error out** to check errors and to specify execution order by wiring **error out** from one node to **error in** of the next node.

U8 **sector out** returns the address of the specified user memory sector.

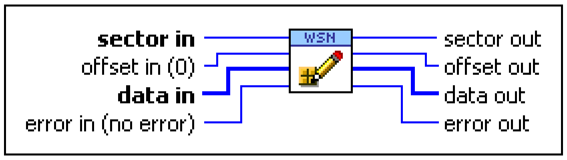
U16 **offset out** returns the byte offset after reading the data.

U8 **data out** returns the data written to the user memory sector.

error out contains an error code. If **error in** indicates that an error occurred before this VI ran, **error out** contains the same value. Otherwise it contains the error code that this VI produces. The value will be 0 if no error occurred, and non-zero if an error occurred.

This VI can return the LabVIEW WSN Module error codes.

Write to Sector UInt16 Array



sector in specifies the user memory sector.

offset in (0) specifies the byte offset to start writing data to in the user memory sector.

data in contains the data you want write to the user memory sector.

error in (no error) describes error conditions that occur before this VI runs. The default is no error. If an error occurred before this VI runs, the VI passes the **error in** value to **error out**. This VI runs normally only if no error occurred before this VI runs. If an error occurs while this VI runs, it runs normally and sets its own error status in **error out**. Use **error in** and **error out** to check errors and to specify execution order by wiring **error out** from one node to **error in** of the next node.

sector out returns the address of the specified user memory sector.

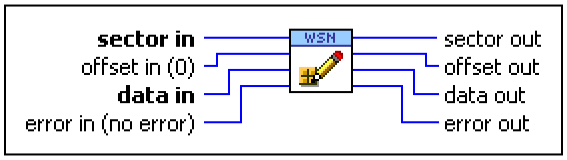
offset out returns the byte offset after reading the data.

data out returns the data written to the user memory sector.

error out contains an error code. If **error in** indicates that an error occurred before this VI ran, **error out** contains the same value. Otherwise it contains the error code that this VI produces. The value will be 0 if no error occurred, and non-zero if an error occurred.

This VI can return the LabVIEW WSN Module error codes.

Write to Sector UInt16



U8 **sector in** specifies the user memory sector.

U16 **offset in (0)** specifies the byte offset to start writing data to in the user memory sector.

U16 **data in** contains the data you want write to the user memory sector.

I16 **error in (no error)** describes error conditions that occur before this VI runs. The default is no error. If an error occurred before this VI runs, the VI passes the **error in** value to **error out**. This VI runs normally only if no error occurred before this VI runs. If an error occurs while this VI runs, it runs normally and sets its own error status in **error out**. Use **error in** and **error out** to check errors and to specify execution order by wiring **error out** from one node to **error in** of the next node.

U8 **sector out** returns the address of the specified user memory sector.

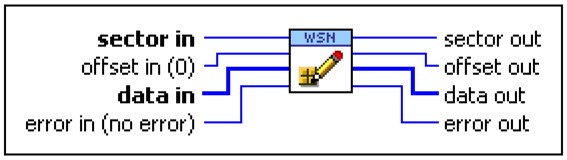
U16 **offset out** returns the byte offset after reading the data.

U16 **data out** returns the data written to the user memory sector.

I16 **error out** contains an error code. If **error in** indicates that an error occurred before this VI ran, **error out** contains the same value. Otherwise it contains the error code that this VI produces. The value will be 0 if no error occurred, and non-zero if an error occurred.

This VI can return the LabVIEW WSN Module error codes.

Write to Sector UInt32 Array



U8 **sector in** specifies the user memory sector.

U16 **offset in (0)** specifies the byte offset to start writing data to in the user memory sector.

[U32] **data in** contains the data you want write to the user memory sector.

I16 **error in (no error)** describes error conditions that occur before this VI runs. The default is no error. If an error occurred before this VI runs, the VI passes the **error in** value to **error out**. This VI runs normally only if no error occurred before this VI runs. If an error occurs while this VI runs, it runs normally and sets its own error status in **error out**. Use **error in** and **error out** to check errors and to specify execution order by wiring **error out** from one node to **error in** of the next node.

U8 **sector out** returns the address of the specified user memory sector.

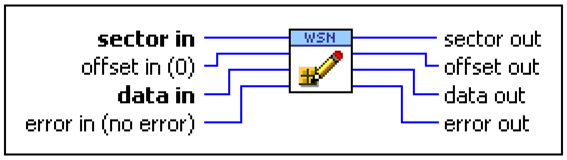
U16 **offset out** returns the byte offset after reading the data.

U32 **data out** returns the data written to the user memory sector.

I16 **error out** contains an error code. If **error in** indicates that an error occurred before this VI ran, **error out** contains the same value. Otherwise it contains the error code that this VI produces. The value will be 0 if no error occurred, and non-zero if an error occurred.

This VI can return the LabVIEW WSN Module error codes.

Write to Sector UInt32



U8 **sector in** specifies the user memory sector.

U16 **offset in (0)** specifies the byte offset to start writing data to in the user memory sector.

U32 **data in** contains the data you want write to the user memory sector.

I16 **error in (no error)** describes error conditions that occur before this VI runs. The default is no error. If an error occurred before this VI runs, the VI passes the **error in** value to **error out**. This VI runs normally only if no error occurred before this VI runs. If an error occurs while this VI runs, it runs normally and sets its own error status in **error out**. Use **error in** and **error out** to check errors and to specify execution order by wiring **error out** from one node to **error in** of the next node.

U8 **sector out** returns the address of the specified user memory sector.

U16 **offset out** returns the byte offset after reading the data.

U32 **data out** returns the data written to the user memory sector.

error out contains an error code. If **error in** indicates that an error occurred before this VI ran, **error out** contains the same value. Otherwise it contains the error code that this VI produces. The value will be 0 if no error occurred, and non-zero if an error occurred.

This VI can return the LabVIEW WSN Module error codes.

Write to Sector Details

You can write to the same location in user memory multiple times, but you must erase the entire user memory sector before you can write to the same location again. You can write to the same user memory sector multiple times as long as you don't overwrite a location in user memory that has already been written to.

User RAM VIs

Owning Palette: WSN VIs

Requires: LabVIEW WSN Module. This topic might not match its corresponding palette in LabVIEW depending on your operating system, licensed product(s), and target.

Use the User RAM VIs to access data in the user RAM on WSN nodes.

Not all WSN nodes have user RAM. Refer to the WSN node documentation to determine if the node has user RAM. Use the Memory Configuration page to configure the amount of RAM available as user RAM.

Palette Object	Description
User RAM Allocate	Allocates the specified number of elements in the user RAM.
User RAM BlockCopy	Copies data to the specified block in the user RAM.
User RAM Free	Frees the specified block of memory in the user RAM.
User RAM Property Node	Gets (reads) and/or sets (writes) properties of a reference to a block of memory in the user RAM.
User RAM Read	Reads data from the specified block of memory in the user RAM.

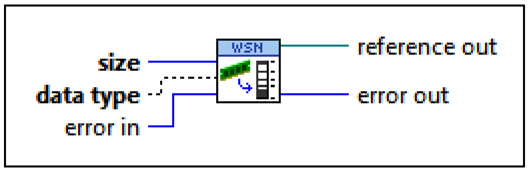
User RAM Write	Writes data to the specified block of memory in the user RAM.

User RAM Allocate VI

Owning Palette: User RAM VIs

Requires: LabVIEW WSN Module

Allocates the specified number of elements in the user RAM.
Details



I32	**size** specifies the number of elements to allocate in the user RAM.
	data type specifies the data type of the elements to allocate in the user RAM. You can allocate waveforms of single-precision, floating-point numbers, signed 8-bit, 16-bit, and 32-bit integers, and unsigned 8-bit, 16-bit, and 32-bit integers in the user RAM.
I16	**error in** describes error conditions that occur before this VI runs. The default is no error. If an error occurred before this VI runs, the VI passes the **error in** value to **error out**. This VI runs normally only if no error occurred before this VI runs. If an error occurs while this VI runs, it runs normally and sets its own error status in **error out**. Use **error in** and **error out** to check errors and to specify execution order by wiring **error out** from one node to **error in** of the next node.
	reference out is a reference to a block of memory in the user RAM.
I16	**error out** contains an error code. If **error in** indicates that an error occurred before this VI ran, **error out** contains the same value. Otherwise it contains the error code that this VI produces. The value will be 0 if no error occurred, and non-zero if an error occurred. This VI can return the LabVIEW WSN Module error codes.

User RAM Allocate Details

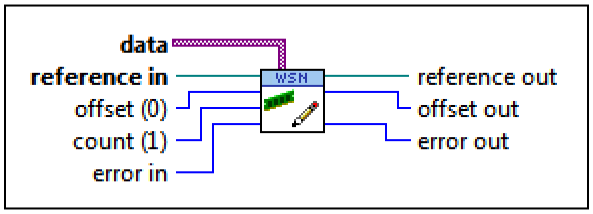
If an error occurs when you use this VI, allocate more memory in the user RAM.

User RAM Write VI

Owning Palette: User RAM VIs

Requires: LabVIEW WSN Module

Writes data to the specified block of memory in the user RAM.



data contains the data to write to the specified block of memory.

reference in is a reference to a block of memory in the user RAM.

offset (0) specifies the element offset to start writing data to the specified block of memory. The default value is `0`.

count (1) specifies the number of elements to write. The default value is `1`.

error in describes error conditions that occur before this VI runs. The default is no error. If an error occurred before this VI runs, the VI passes the **error in** value to **error out**. This VI runs normally only if no error occurred before this VI runs. If an error occurs while this VI runs, it runs normally and sets its own error status in **error out**. Use **error in** and **error out** to check errors and to specify execution order by wiring **error out** from one node to **error in** of the next node.

reference out returns **reference in**.

offset out returns the element offset after writing the data.

error out contains an error code. If **error in** indicates that an error occurred before this VI ran, **error out** contains the same value. Otherwise it contains the error code that this VI produces. The value will be 0 if no error occurred, and non-zero if an error occurred.

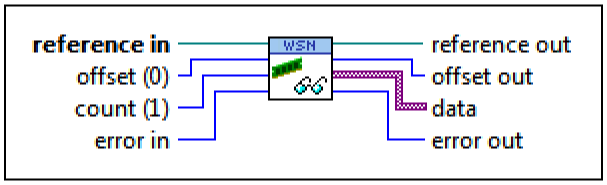
This VI can return the LabVIEW WSN Module error codes.

User RAM Read VI

Owning Palette: User RAM VIs

Requires: LabVIEW WSN Module

Reads data from the specified block of memory in the user RAM.



reference in is a reference to a block of memory in the user RAM.

offset (0) specifies the element offset to start reading data from the specified block of memory. The default value is `0`.

count (1) specifies the number of elements to read. The default value is `1`.

error in describes error conditions that occur before this VI runs. The default is no error. If an error occurred before this VI runs, the VI passes the **error in** value to **error out**. This VI runs normally only if no error occurred before this VI runs. If an error occurs while this VI runs, it runs normally and sets its own error status in **error out**. Use **error in** and **error out** to check errors and to specify execution order by wiring **error out** from one node to **error in** of the next node.

reference out returns **reference in**.

offset out returns the element offset after reading the data.

data returns the data read from the specified block of memory.

error out contains an error code. If **error in** indicates that an error occurred before this VI ran, **error out** contains the same value. Otherwise it contains the error code that this VI produces. The value will be 0 if no error occurred, and non-zero if an error occurred.

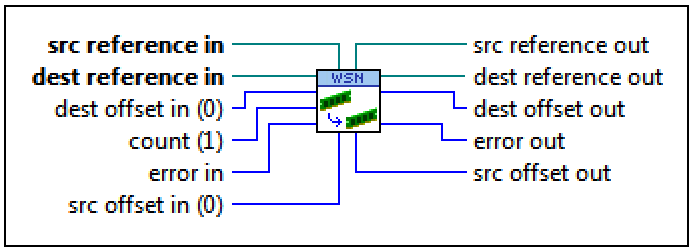
This VI can return the LabVIEW WSN Module error codes.

User RAM BlockCopy VI

Owning Palette: User RAM VIs

Requires: LabVIEW WSN Module

Copies data to the specified block in the user RAM. This VI allocates a block of memory if you do not wire **dest reference in**.



src reference in is a reference to the data to copy to the user RAM.

dest reference in is a reference to the block of memory to which to copy the data referenced by **src reference in**.

dest offset in (0) specifies the element offset to start copying data to in the block of memory. The default value is `0`.

count (1) specifies the number of elements to copy. The default value is `1`.

error in describes error conditions that occur before this VI runs. The default is no error. If an error occurred before this VI runs, the VI passes the **error in** value to **error out**. This VI runs normally only if no error occurred before this VI runs. If an error occurs while this VI runs, it runs normally and sets its own error status in **error out**. Use **error in** and **error out** to check errors and to specify execution order by wiring **error out** from one node to **error in** of the next node.

src offset in (0) specifies the element offset to start copy data from in the **src reference in**. The default value is `0`.

src reference out returns **src reference in**.

dest reference out returns **dest reference in**.

dest offset out returns the element offset after copying data to the block of user RAM.

error out contains an error code. If **error in** indicates that an error occurred before this VI ran, **error out** contains the same value. Otherwise it contains the error code that this VI produces. The value will be 0 if no error occurred, and non-zero if an error occurred.

This VI can return the LabVIEW WSN Module error codes.

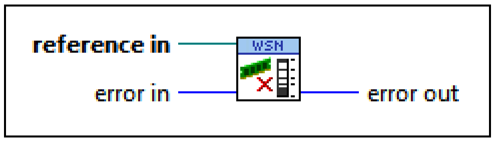
src offset out returns the element offset of **src reference in** after copying the data.

User RAM Free VI

Owning Palette: User RAM VIs

Requires: LabVIEW WSN Module

Frees the specified block of memory in the user RAM.



reference in is a reference to a block of memory in the user RAM.

error in describes error conditions that occur before this VI runs. The default is no error. If an error occurred before this VI runs, the VI passes the **error in** value to **error out**. This VI runs normally only if no error occurred before this VI runs. If an error occurs while this VI runs, it runs normally and sets its own error status in **error out**. Use **error in** and **error out** to check errors and to specify execution order by wiring **error out** from one node to **error in** of the next node.

error out contains an error code. If **error in** indicates that an error occurred before this VI ran, **error out** contains the same value. Otherwise it contains the error code that this VI produces. The value will be 0 if no error occurred, and non-zero if an error occurred.

This VI can return the LabVIEW WSN Module error codes.

User RAM Property Node

Owning Palette: User RAM VIs

Requires: LabVIEW WSN Module

Gets (reads) and/or sets (writes) properties of a reference to a block of memory in the user RAM.

count returns the number of elements in the block of memory referenced.

dt is the time interval between data points in the waveform. To specify whether this parameter is a control or an indicator, right-click an element in the **Property** section of the User RAM Property Node and select **Change To Read** or **Change To Write** from the shortcut menu.

t0 is the time the first sample in the waveform was acquired. To specify whether this parameter is a control or an indicator, right-click an element in the **Property** section of the User RAM Property Node and select **Change To Read** or **Change To Write** from the shortcut menu.

Configuration & Communication VIs

Owning Palette: WSN VIs

Requires: LabVIEW WSN Module. This topic might not match its corresponding palette in LabVIEW depending on your operating system, licensed product(s), and target.

Use the Configuration & Communication VIs to configure and communicate with LabVIEW WSN nodes.

Palette Object	Description
Config Node	Configures the WSN node.
Radio Messages	Sends and receives data between the host comp uter for the WSN network and the WSN nodes.

Config Node VI

Owning Palette: Configuration & Communication VIs

Requires: LabVIEW WSN Module

Configures the WSN node.

If you are using an NI LabVIEW WSN target, NI WSN nodes include the following Config Node properties that you can use with the Elemental I/O Property Node:

- NI WSN-3202 Config Node Properties
- NI WSN-3212 Config Node Properties
- NI WSN-3214 Config Node Properties
- NI WSN-3226 Config Node Properties
- NI WSN-3230/3231 Config Node Properties

Con! g Node VI for the NI WSN-3202 Node

Owning Palette: Configuration & Communication VIs

Requires: LabVIEW WSN Module

Configures the WSN node.

Parameter	Description
Sample Interval Mode	Specifies whether the host computer for the WS N network or the VI on the WSN node controls th e rate at which samples are taken. The default v alue is **VI Driven**. Possible inputs are: - **Host Driven**—Specifies that the host c omputer for the WSN network controls th e sample interval. - **VI Driven**—Specifies that the VI on the WSN node controls the sample interval.
Sample Interval	Sets or returns the interval in seconds at which t he sample case is executed. Default is **5**. The new value is immediately applied.
Flash.SectorSize	Returns the size of the user memory sector.
Flash.NumberofSectors	Returns the number of user memory sectors ava ilable.
Maximum Debug Message Size	Returns the maximum allowable size of debug messages. If you send a debug message that exc eeds the maximum size, the message is truncat ed to the maximum size.
Maximum User Message Size	Returns the maximum allowable size of user me ssages. If you send a user message that exceeds the maximum size, the message is truncated to t he maximum size.
Memory.Current Heap Usage	Returns the amount of heap memory currently allocated for the application.

	The heap memory is located in RAM. An applicat ion running on a WSN node uses the heap mem ory to store strings, arrays, and clusters. The ap plication uses statically allocated RAM to store n umeric values. Right-click the build specificatio n for the application in the **Project Explorer** wi ndow and select **Application Information** fro m the shortcut menu to view how much static m emory is allocated for the application.
Memory.Maximum Heap Usage	Returns the maximum amount of heap memory that the application has allocated. The heap memory is located in RAM. An applicat ion running on a WSN node uses the heap mem ory to store strings, arrays, and clusters. The ap plication uses statically allocated RAM to store n umeric values. Right-click the build specificatio n for the application in the **Project Explorer** wi ndow and select **Application Information** fro m the shortcut menu to view how much static m emory is allocated for the application.
Memory.Available Heap	Returns the amount of free heap memory. The heap memory is located in RAM. An applicat ion running on a WSN node uses the heap mem ory to store strings, arrays, and clusters. The ap plication uses statically allocated RAM to store n umeric values. Right-click the build specificatio n for the application in the **Project Explorer** wi ndow and select **Application Information** fro m the shortcut menu to view how much static m emory is allocated for the application.
Memory.Total Heap	Returns the size of heap memory. The heap memory is located in RAM. An applicat ion running on a WSN node uses the heap mem ory to store strings, arrays, and clusters. The ap plication uses statically allocated RAM to store n umeric values. Right-click the build specificatio n for the application in the **Project Explorer** wi

	ndow and select **Application Information** fro m the shortcut menu to view how much static m emory is allocated for the application.
Memory.Current Stack Usage	Returns the amount of stack memory currently allocated to the application. The stack memory is located in RAM. Stack mem ory stores the parameters and local variables of subVIs. If an application uses more stack memor y than is available, the application might crash.
Memory.Maximum Stack Usage	Returns the maximum amount of stack memory that the application has used. The stack memory is located in RAM. Stack mem ory stores the parameters and local variables of subVIs. If an application uses more stack memor y than is available, the application might crash.
Memory.Available Stack	Returns the amount of free stack memory. The stack memory is located in RAM. Stack mem ory stores the parameters and local variables of subVIs. If an application uses more stack memor y than is available, the application might crash.
Memory.Total Stack	Returns the size of stack memory. The stack memory is located in RAM. Stack mem ory stores the parameters and local variables of subVIs. If an application uses more stack memor y than is available, the application might crash.

Con! g Node VI for the NI WSN-3212 Node

Owning Palette: Configuration & Communication VIs

Requires: LabVIEW WSN Module

Configures the WSN node.

Parameter	Description
Sample Interval Mode	Specifies whether the host computer for the WS N network or the VI on the WSN node controls th

	e rate at which samples are taken. The default value is **VI Driven**. Possible inputs are: ▪ **Host Driven**—Specifies that the host computer for the WSN network controls the sample interval. ▪ **VI Driven**—Specifies that the VI on the WSN node controls the sample interval.
Sample Interval	Sets or returns the interval in seconds at which the sample case is executed. Default is **5**. The new value is immediately applied.
Flash.SectorSize	Returns the size of the user memory sector.
Flash.NumberofSectors	Returns the number of user memory sectors available.
Maximum Debug Message Size	Returns the maximum allowable size of debug messages. If you send a debug message that exceeds the maximum size, the message is truncated to the maximum size.
Maximum User Message Size	Returns the maximum allowable size of user messages. If you send a user message that exceeds the maximum size, the message is truncated to the maximum size.
Memory.Current Heap Usage	Returns the amount of heap memory currently allocated for the application. The heap memory is located in RAM. An application running on a WSN node uses the heap memory to store strings, arrays, and clusters. The application uses statically allocated RAM to store numeric values. Right-click the build specification for the application in the **Project Explorer** window and select **Application Information** from the shortcut menu to view how much static memory is allocated for the application.

Memory.Maximum Heap Usage	Returns the maximum amount of heap memory that the application has allocated. The heap memory is located in RAM. An applicat ion running on a WSN node uses the heap mem ory to store strings, arrays, and clusters. The ap plication uses statically allocated RAM to store n umeric values. Right-click the build specificatio n for the application in the **Project Explorer** wi ndow and select **Application Information** fro m the shortcut menu to view how much static m emory is allocated for the application.
Memory.Available Heap	Returns the amount of free heap memory. The heap memory is located in RAM. An applicat ion running on a WSN node uses the heap mem ory to store strings, arrays, and clusters. The ap plication uses statically allocated RAM to store n umeric values. Right-click the build specificatio n for the application in the **Project Explorer** wi ndow and select **Application Information** fro m the shortcut menu to view how much static m emory is allocated for the application.
Memory.Total Heap	Returns the size of heap memory. The heap memory is located in RAM. An applicat ion running on a WSN node uses the heap mem ory to store strings, arrays, and clusters. The ap plication uses statically allocated RAM to store n umeric values. Right-click the build specificatio n for the application in the **Project Explorer** wi ndow and select **Application Information** fro m the shortcut menu to view how much static m emory is allocated for the application.
Memory.Current Stack Usage	Returns the amount of stack memory currently allocated to the application. The stack memory is located in RAM. Stack mem ory stores the parameters and local variables of

	subVIs. If an application uses more stack memor y than is available, the application might crash.
Memory.Maximum Stack Usage	Returns the maximum amount of stack memory that the application has used. The stack memory is located in RAM. Stack mem ory stores the parameters and local variables of subVIs. If an application uses more stack memor y than is available, the application might crash.
Memory.Available Stack	Returns the amount of free stack memory. The stack memory is located in RAM. Stack mem ory stores the parameters and local variables of subVIs. If an application uses more stack memor y than is available, the application might crash.
Memory.Total Stack	Returns the size of stack memory. The stack memory is located in RAM. Stack mem ory stores the parameters and local variables of subVIs. If an application uses more stack memor y than is available, the application might crash.

Con! g Node VI for the NI WSN-3214 Node

Owning Palette: Configuration & Communication VIs

Requires: LabVIEW WSN Module

Configures the WSN node.

Parameter	Description
Sample Interval	Sets or returns the waveform interval. This is th e time between successive waveform acquisitio ns, in seconds. If set, the new value is immediat ely applied. The default value is **30**. Possible inputs are: ▪ 0.1 to 100000 seconds

	For a diagram, see NI WSN-3214 Waveform Parameters.
Sample Interval Mode	Specifies whether the host computer for the WSN network or the VI on the WSN node controls the rate at which samples are taken. The default value is **VI Driven**. Possible inputs are: ▪ **Host Driven**—Specifies that the host computer for the WSN network controls the sample interval. ▪ **VI Driven**—Specifies that the VI on the WSN node controls the sample interval.
Waveform Sample Rate	Specifies the sampling rate in samples per channel per second (SPS/ch)during each waveform. The default value is **10**. Possible inputs are: ▪ 1 to max rate that hardware supports (SPS/ch). Sample rates depend on filtering and aperture settings. See sample rates. For a diagram, see NI WSN-3214 Waveform Parameters.
Samples to Read	Specifies the number of samples to read for each waveform, for each configured analog channel. The default value is **10**. For a diagram, see NI WSN-3214 Waveform Parameters.
Powerline Filtering	Sets or returns powerline filtering for the node. The default value is **Host Driven**. Possible inputs are: ▪ **Host Driven**—For this attribute, use the setting from the host computer. ▪ **None**—Do not reject powerline noise. ▪ **50 Hz**—Rejects 50 Hz powerline noise.

	▪ **60 Hz**—Rejects 60 Hz powerline noise. ▪ **50/60 Hz**—Rejects 50 Hz and 60 Hz po werline noise.
Filtering Strength	**Note** Powerline Filtering must be enabled to use Filtering Strength. Sets or returns the filtering strength for the nod e. The default value is **Host Driven**. Possible inputs are: ▪ **Host Driven**—For this attribute, use th e setting from the host computer. ▪ **High Rejection**—Rejects powerline no ise more, at the expense of higher energy consumption. ▪ **Low Power**—Rejects powerline noise, but as energy efficiently as possible.
Aperture Time	Specifies the period during which the ADC is rea ding the input signal. Default is **1.4** ms. Possible inputs are: ▪ 250 us ▪ 400 us ▪ 750 us ▪ 1.4 ms ▪ 2.8 ms ▪ 5.5 ms ▪ 10.8 ms ▪ 21.5 ms
Flash.SectorSize	Returns the size of the user memory sector.
Flash.NumberofSectors	Returns the number of user memory sectors ava ilable.
Maximum Debug Message Size	Returns the maximum allowable size of debug messages. If you send a debug message that exc

	eeds the maximum size, the message is truncat ed to the maximum size.
Maximum User Message Size	Returns the maximum allowable size of user me ssages. If you send a user message that exceeds the maximum size, the message is truncated to t he maximum size.
Memory.Current Heap Usage	Returns the amount of heap memory currently allocated for the application. The heap memory is located in RAM. An applicat ion running on a WSN node uses the heap mem ory to store strings, arrays, and clusters. The ap plication uses statically allocated RAM to store n umeric values. Right-click the build specificatio n for the application in the **Project Explorer** wi ndow and select **Application Information** fro m the shortcut menu to view how much static m emory is allocated for the application.
Memory.Maximum Heap Usage	Returns the maximum amount of heap memory that the application has allocated. The heap memory is located in RAM. An applicat ion running on a WSN node uses the heap mem ory to store strings, arrays, and clusters. The ap plication uses statically allocated RAM to store n umeric values. Right-click the build specificatio n for the application in the **Project Explorer** wi ndow and select **Application Information** fro m the shortcut menu to view how much static m emory is allocated for the application.
Memory.Available Heap	Returns the amount of free heap memory. The heap memory is located in RAM. An applicat ion running on a WSN node uses the heap mem ory to store strings, arrays, and clusters. The ap plication uses statically allocated RAM to store n umeric values. Right-click the build specificatio n for the application in the **Project Explorer** wi ndow and select **Application Information** fro

	m the shortcut menu to view how much static m emory is allocated for the application.
Memory.Total Heap	Returns the size of heap memory. The heap memory is located in RAM. An applicat ion running on a WSN node uses the heap mem ory to store strings, arrays, and clusters. The ap plication uses statically allocated RAM to store n umeric values. Right-click the build specificatio n for the application in the **Project Explorer** wi ndow and select **Application Information** fro m the shortcut menu to view how much static m emory is allocated for the application.
Memory.Current Stack Usage	Returns the amount of stack memory currently allocated to the application. The stack memory is located in RAM. Stack mem ory stores the parameters and local variables of subVIs. If an application uses more stack memor y than is available, the application might crash.
Memory.Maximum Stack Usage	Returns the maximum amount of stack memory that the application has used. The stack memory is located in RAM. Stack mem ory stores the parameters and local variables of subVIs. If an application uses more stack memor y than is available, the application might crash.
Memory.Available Stack	Returns the amount of free stack memory. The stack memory is located in RAM. Stack mem ory stores the parameters and local variables of subVIs. If an application uses more stack memor y than is available, the application might crash.
Memory.Total Stack	Returns the size of stack memory. The stack memory is located in RAM. Stack mem ory stores the parameters and local variables of subVIs. If an application uses more stack memor y than is available, the application might crash.

Con! g Node VI for the NI WSN-3226 Node

Owning Palette: Configuration & Communication VIs

Requires: LabVIEW WSN Module

Configures the WSN node.

Parameter	Description
Sample Interval Mode	Specifies whether the host computer for the WS N network or the VI on the WSN node controls th e rate at which samples are taken. The default v alue is **VI Driven**. Possible inputs are: ▪ **Host Driven**—Specifies that the host c omputer for the WSN network controls th e sample interval. ▪ **VI Driven**—Specifies that the VI on the WSN node controls the sample interval.
Sample Interval	Sets or returns the interval in seconds at which s ampling occurs. The new value is immediately applied.
Sensor Power	Sets or returns the sensor power timing for the node. Sensor power turns off after the acquisitio n (except for Always On). The default value is **Ho st Driven**. Possible inputs are: ▪ **Host Driven**—For this attribute, use th e setting from the host computer. ▪ **0 ms before sampling**—Turns on the Sensor Power immediately when the anal og acquisition starts. ▪ **25 ms before sampling**—Turns on Se nsor Power 25 ms before the analog acqui sition starts.

	▪ **100 ms before sampling**—Turns on Sensor 100 ms before the analog acquisition starts. ▪ **250 ms before sampling**—Turns on Sensor Power 250 ms before the analog acquisition starts. ▪ **Always On**—Turns on Sensor Power when the next analog acquisition starts and leaves it on indefinitely. ▪ **Always Off**—Sensor Power never turns on.
Powerline Filtering	Sets or returns powerline filtering for the node. The default value is **Host Driven**. Possible inputs are: ▪ **Host Driven**—For this attribute, use the setting from the host computer. ▪ **None**—Do not reject powerline noise. ▪ **50 Hz**—Rejects 50 Hz powerline noise. ▪ **60 Hz**—Rejects 60 Hz powerline noise. ▪ **50/60 Hz**—Rejects 50 Hz and 60 Hz powerline noise.
Filtering Strength	**Note** Powerline Filtering must be enabled to use Filtering Strength. Sets or returns the filtering strength for the node. The default value is **Host Driven**. Possible inputs are: ▪ **Host Driven**—For this attribute, use the setting from the host computer. ▪ **High Rejection**—Rejects powerline noise more, at the expense of higher energy consumption.

	▪ **Low Power**—Rejects powerline noise, but as energy efficiently as possible.
RTD Resistance Range	Sets or returns the RTD/Resistance range for the node. This attribute sets the excitation current a nd scaling for analog input channels which use a Measurement Type of Resistance or RTD. The default value is **Host Driven**. Possible inputs are: ▪ **Host Driven**—For this attribute, use th e setting from the host computer. ▪ **400 ohms/Pt100**—Sets the excitation current for a resistor up to 400 ohms or a Pt100 RTD. ▪ **4 kiloohms/Pt1000**—Sets the excitati on current for a resistor up to 4 kiloohms or a Pt1000 RTD. ▪ **100 kiloohms**—Sets the excitation cur rent for a resistor up to 100 kiloohms.
Flash.SectorSize	Returns the size of the user memory sector.
Flash.NumberofSectors	Returns the number of user memory sectors ava ilable.
Maximum Debug Message Size	Returns the maximum allowable size of debug messages. If you send a debug message that exc eeds the maximum size, the message is truncat ed to the maximum size.
Maximum User Message Size	Returns the maximum allowable size of user me ssages. If you send a user message that exceeds the maximum size, the message is truncated to t he maximum size.
Memory.Current Heap Usage	Returns the amount of heap memory currently allocated for the application. The heap memory is located in RAM. An applicat ion running on a WSN node uses the heap mem ory to store strings, arrays, and clusters. The ap plication uses statically allocated RAM to store n

	umeric values. Right-click the build specification for the application in the **Project Explorer** window and select **Application Information** from the shortcut menu to view how much static memory is allocated for the application.
Memory.Maximum Heap Usage	Returns the maximum amount of heap memory that the application has allocated. The heap memory is located in RAM. An application running on a WSN node uses the heap memory to store strings, arrays, and clusters. The application uses statically allocated RAM to store numeric values. Right-click the build specification for the application in the **Project Explorer** window and select **Application Information** from the shortcut menu to view how much static memory is allocated for the application.
Memory.Available Heap	Returns the amount of free heap memory. The heap memory is located in RAM. An application running on a WSN node uses the heap memory to store strings, arrays, and clusters. The application uses statically allocated RAM to store numeric values. Right-click the build specification for the application in the **Project Explorer** window and select **Application Information** from the shortcut menu to view how much static memory is allocated for the application.
Memory.Total Heap	Returns the size of heap memory. The heap memory is located in RAM. An application running on a WSN node uses the heap memory to store strings, arrays, and clusters. The application uses statically allocated RAM to store numeric values. Right-click the build specification for the application in the **Project Explorer** window and select **Application Information** from the shortcut menu to view how much static memory is allocated for the application.

Memory.Current Stack Usage	Returns the amount of stack memory currently allocated to the application. The stack memory is located in RAM. Stack mem ory stores the parameters and local variables of subVIs. If an application uses more stack memor y than is available, the application might crash.
Memory.Maximum Stack Usage	Returns the maximum amount of stack memory that the application has used. The stack memory is located in RAM. Stack mem ory stores the parameters and local variables of subVIs. If an application uses more stack memor y than is available, the application might crash.
Memory.Available Stack	Returns the amount of free stack memory. The stack memory is located in RAM. Stack mem ory stores the parameters and local variables of subVIs. If an application uses more stack memor y than is available, the application might crash.
Memory.Total Stack	Returns the size of stack memory. The stack memory is located in RAM. Stack mem ory stores the parameters and local variables of subVIs. If an application uses more stack memor y than is available, the application might crash.

Con! g Node VI for the NI WSN-3230/3231 Node

Owning Palette: Configuration & Communication VIs

Requires: LabVIEW WSN Module

Configures the WSN node.

Parameter	Description
Sample Interval Mode	Specifies whether the host computer for the WS N network or the VI on the WSN node controls th

	e rate at which samples are taken. The default v alue is **VI Driven**. Possible inputs are: ▪ **Host Driven**—Specifies that the host c omputer for the WSN network controls th e sample interval. ▪ **VI Driven**—Specifies that the VI on the WSN node controls the sample interval.
Sample Interval	Sets or returns the interval in seconds at which s ampling occurs. The new value is immediately applied.
Flash.SectorSize	Returns the size of the user memory sector.
Flash.NumberofSectors	Returns the number of user memory sectors ava ilable.
Maximum Debug Message Size	Returns the maximum allowable size of debug messages. If you send a debug message that exc eeds the maximum size, the message is truncat ed to the maximum size.
Maximum User Message Size	Returns the maximum allowable size of user me ssages. If you send a user message that exceeds the maximum size, the message is truncated to t he maximum size.
Memory.Current Heap Usage	Returns the amount of heap memory currently allocated for the application. The heap memory is located in RAM. An applicat ion running on a WSN node uses the heap mem ory to store strings, arrays, and clusters. The ap plication uses statically allocated RAM to store n umeric values. Right-click the build specificatio n for the application in the **Project Explorer** wi ndow and select **Application Information** fro m the shortcut menu to view how much static m emory is allocated for the application.
Memory.Maximum Heap Usage	Returns the maximum amount of heap memory that the application has allocated.

	The heap memory is located in RAM. An applicat ion running on a WSN node uses the heap mem ory to store strings, arrays, and clusters. The ap plication uses statically allocated RAM to store n umeric values. Right-click the build specificatio n for the application in the **Project Explorer** wi ndow and select **Application Information** fro m the shortcut menu to view how much static m emory is allocated for the application.
Memory.Available Heap	Returns the amount of free heap memory. The heap memory is located in RAM. An applicat ion running on a WSN node uses the heap mem ory to store strings, arrays, and clusters. The ap plication uses statically allocated RAM to store n umeric values. Right-click the build specificatio n for the application in the **Project Explorer** wi ndow and select **Application Information** fro m the shortcut menu to view how much static m emory is allocated for the application.
Memory.Total Heap	Returns the size of heap memory. The heap memory is located in RAM. An applicat ion running on a WSN node uses the heap mem ory to store strings, arrays, and clusters. The ap plication uses statically allocated RAM to store n umeric values. Right-click the build specificatio n for the application in the **Project Explorer** wi ndow and select **Application Information** fro m the shortcut menu to view how much static m emory is allocated for the application.
Memory.Current Stack Usage	Returns the amount of stack memory currently allocated to the application. The stack memory is located in RAM. Stack mem ory stores the parameters and local variables of subVIs. If an application uses more stack memor y than is available, the application might crash.
Memory.Maximum Stack Usage	Returns the maximum amount of stack memory that the application has used.

	The stack memory is located in RAM. Stack mem ory stores the parameters and local variables of subVIs. If an application uses more stack memor y than is available, the application might crash.
Memory.Available Stack	Returns the amount of free stack memory. The stack memory is located in RAM. Stack mem ory stores the parameters and local variables of subVIs. If an application uses more stack memor y than is available, the application might crash.
Memory.Total Stack	Returns the size of stack memory. The stack memory is located in RAM. Stack mem ory stores the parameters and local variables of subVIs. If an application uses more stack memor y than is available, the application might crash.

Con! g Node VI for the NI WSN-3230/3231 Node

Owning Palette: Configuration & Communication VIs

Requires: LabVIEW WSN Module

Configures the WSN node.

Parameter	Description
Sample Interval Mode	Specifies whether the host computer for the WS N network or the VI on the WSN node controls th e rate at which samples are taken. The default v alue is **VI Driven**. Possible inputs are: ▪ **Host Driven**—Specifies that the host c omputer for the WSN network controls th e sample interval. ▪ **VI Driven**—Specifies that the VI on the WSN node controls the sample interval.

Sample Interval	Sets or returns the interval in seconds at which s ampling occurs. The new value is immediately applied.
Flash.SectorSize	Returns the size of the user memory sector.
Flash.NumberofSectors	Returns the number of user memory sectors ava ilable.
Maximum Debug Message Size	Returns the maximum allowable size of debug messages. If you send a debug message that exc eeds the maximum size, the message is truncat ed to the maximum size.
Maximum User Message Size	Returns the maximum allowable size of user me ssages. If you send a user message that exceeds the maximum size, the message is truncated to t he maximum size.
Memory.Current Heap Usage	Returns the amount of heap memory currently allocated for the application. The heap memory is located in RAM. An applicat ion running on a WSN node uses the heap mem ory to store strings, arrays, and clusters. The ap plication uses statically allocated RAM to store n umeric values. Right-click the build specificatio n for the application in the **Project Explorer** wi ndow and select **Application Information** fro m the shortcut menu to view how much static m emory is allocated for the application.
Memory.Maximum Heap Usage	Returns the maximum amount of heap memory that the application has allocated. The heap memory is located in RAM. An applicat ion running on a WSN node uses the heap mem ory to store strings, arrays, and clusters. The ap plication uses statically allocated RAM to store n umeric values. Right-click the build specificatio n for the application in the **Project Explorer** wi ndow and select **Application Information** fro m the shortcut menu to view how much static m emory is allocated for the application.

Memory.Available Heap	Returns the amount of free heap memory. The heap memory is located in RAM. An applicat ion running on a WSN node uses the heap mem ory to store strings, arrays, and clusters. The ap plication uses statically allocated RAM to store n umeric values. Right-click the build specificatio n for the application in the **Project Explorer** wi ndow and select **Application Information** fro m the shortcut menu to view how much static m emory is allocated for the application.
Memory.Total Heap	Returns the size of heap memory. The heap memory is located in RAM. An applicat ion running on a WSN node uses the heap mem ory to store strings, arrays, and clusters. The ap plication uses statically allocated RAM to store n umeric values. Right-click the build specificatio n for the application in the **Project Explorer** wi ndow and select **Application Information** fro m the shortcut menu to view how much static m emory is allocated for the application.
Memory.Current Stack Usage	Returns the amount of stack memory currently allocated to the application. The stack memory is located in RAM. Stack mem ory stores the parameters and local variables of subVIs. If an application uses more stack memor y than is available, the application might crash.
Memory.Maximum Stack Usage	Returns the maximum amount of stack memory that the application has used. The stack memory is located in RAM. Stack mem ory stores the parameters and local variables of subVIs. If an application uses more stack memor y than is available, the application might crash.
Memory.Available Stack	Returns the amount of free stack memory. The stack memory is located in RAM. Stack mem ory stores the parameters and local variables of

	subVIs. If an application uses more stack memor y than is available, the application might crash.
Memory.Total Stack	Returns the size of stack memory. The stack memory is located in RAM. Stack mem ory stores the parameters and local variables of subVIs. If an application uses more stack memor y than is available, the application might crash.

Radio Messages VI

Owning Palette: Configuration & Communication VIs

Requires: LabVIEW WSN Module

Sends and receives messages between the WSN nodes and the host computer for the WSN network.

Block Diagram Inputs
Block Diagram Outputs

Block Diagram Inputs

Parameter	Description
Send User Message	Sends a message from the WSN node to the gat eway. Use the WSN Host API VIs to read user messages from the gateway.
Send Debug Message	Sends a debug message from the WSN node to t he gateway. Use the WSN Host API VIs to read debug messag es from the gateway.

Block Diagram Outputs

Parameter	Description
Network Status	Shows whether the WSN node is connected to t he network.

Received User Message	Returns a message received by the WSN node. Works only if called from the receive state of the state machine.
Sample Interval	Represents the sampling interval sent to the WS N node by the host computer. If you use the Con fig Node VI and set the **Sample Interval Mode** input to **VI Driven**, the **Sample Interval** input might not represent the actual sample interval of the WSN node.

Serial VIs

Owning Palette: WSN VIs

Requires: LabVIEW WSN Module. This topic might not match its corresponding palette in LabVIEW depending on your operating system, licensed product(s), and target.

Use the Serial VIs to configure and communicate with the LabVIEW WSN serial nodes.

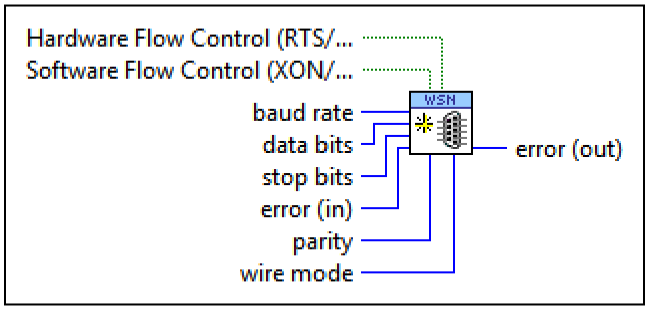
Palette Object	Description
WSN Serial Port Init VI	Initializes and powers on the serial port to the s ettings you specify.
WSN Serial Port Receiver Power VI	Enables or disables power to the serial receiver.
WSN Serial Port Read VI	Reads the number of characters specified by **re quested byte count** from the serial port.
WSN Serial Port Close VI	Closes and powers down the serial port.
WSN Serial Port Bytes At Port VI	Returns the number of bytes in the input buffer f or the serial port.
WSN Serial Port Break VI	Sends a break on the serial port for a period of ti me based on the **duration** input.
WSN Serial Port Write VI	Writes the data in **string to write** to the serial p ort.

WSN Serial Port Init VI

Owning Palette: Serial

Requires: LabVIEW WSN Module

Initializes and powers on the serial port to the settings you specify.



Software Flow Control (XON/XOFF) is a software handshaking protocol you can use to avoid overflowing the serial port buffers. The default is FALSE.

Hardware Flow Control (RTS/CTS) corresponds to Request To Send (RTS) and Clear to Send (CTS) handshaking. The default is FALSE..

baud rate is the rate of transmission. The default is 9600.

data bits is the number of bits in the incoming data and can be 7 or 8. The default is 8.

stop bits specifies the number of stop bits used to indicate the end of a frame. (enumerated values)

- 0 is 1 stop bit (default)
- 2 is 2 stop bits

parity specifies the parity used to transmit or receive every frame. (enumerated values)

- 0 is no parity (default)
- 1 is odd parity
- 2 is even parity

wire mode specifies the transceiver operating mode. (enumerated values)

- 0 is 2-wire mode
- 1 is 4-wire mode

error in describes error conditions that occur before this VI runs. The default is no error. If an error occurred before this VI runs, the VI passes the **error in** value to **error out**. This VI runs normally only if no error occurred before this VI runs. If an error occurs while this VI runs, it runs normally and sets its own error status in **error out**. Use **error in** and **error out** to check errors and to specify execution order by wiring **error out** from one node to **error in** of the next node.

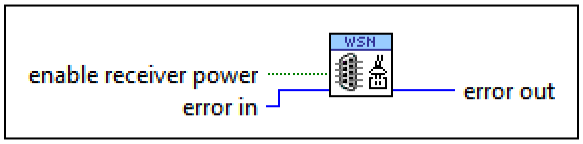
error out contains an error code. If **error in** indicates that an error occurred before this VI ran, **error out** contains the same value. Otherwise it contains the error code that this VI produces. The value will be 0 if no error occurred and non-zero if an error occurred.

WSN Serial Port Receiver Power VI

Owning Palette: Serial

Requires: LabVIEW WSN Module

Enables or disables power to the serial receiver.



enable receiver power enables or disables power to the serial receiver.

error in describes error conditions that occur before this VI runs. The default is no error. If an error occurred before this VI runs, the VI passes the **error in** value to **error out**. This VI runs normally only if no error occurred before this VI runs. If an error occurs while this VI runs, it runs normally and sets its own error status in **error out**. Use **error in** and **error out** to check errors and to specify execution order by wiring **error out** from one node to **error in** of the next node.

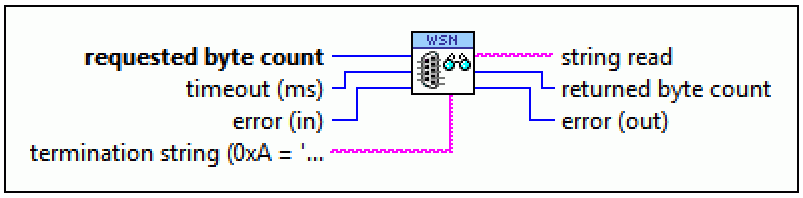
error out contains an error code. If **error in** indicates that an error occurred before this VI ran, **error out** contains the same value. Otherwise it contains the error code that this VI produces. The value will be 0 if no error occurred and non-zero if an error occurred.

WSN Serial Port Read VI

Owning Palette: Serial

Requires: LabVIEW WSN Module

Reads the number of characters specified by **requested byte count** from the serial port.



requested byte count specifies the number of characters to read. If you want to read the characters currently at the serial port, first execute the WSN Serial Port Bytes At Port VI to determine the exact number of bytes ready to read. Use the byte count output of the WSN Serial Port Bytes At Port VI as the requested byte count input to the Serial Port Read VI.

timeout (ms) specifies the amount of time, in milliseconds, to wait for this function to read the **requested byte count** or the **termination string**. The default value is 2000.

termination string calls for termination of the read operation. The read operation terminates when the termination string is read from the serial device. 0xA is the hex equivalent of a linefeed character (\n). Change the termination string to 0xD for message strings that terminate with a carriage return (\r). The default is the linefeed character (0xA).

string read is the data read from the serial port.

returned byte count is the number of bytes read from the serial port. This value is the lesser of the requested byte count and the number of bytes up to and including the first termination character found.

error in describes error conditions that occur before this VI runs. The default is no error. If an error occurred before this VI runs, the VI passes the **error in** value to **error out**. This VI runs normally only if no error occurred before this VI runs. If an error occurs while this VI runs, it runs normally and sets its own error status in **error out**. Use **error in** and **error out** to check errors and to specify execution order by wiring **error out** from one node to **error in** of the next node.

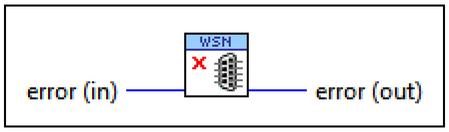
error out contains an error code. If **error in** indicates that an error occurred before this VI ran, **error out** contains the same value. Otherwise it contains the error code that this VI produces. The value will be 0 if no error occurred and non-zero if an error occurred.

WSN Serial Port Close VI

Owning Palette: Serial

Requires: LabVIEW WSN Module

Closes and powers down the serial port.



I32 **error in** describes error conditions that occur before this VI runs. The default is no error. If an error occurred before this VI runs, the VI passes the **error in** value to **error out**. This VI runs normally only if no error occurred before this VI runs. If an error occurs while this VI runs, it runs normally and sets its own error status in **error out**. Use **error in** and **error out** to check errors and to specify execution order by wiring **error out** from one node to **error in** of the next node.

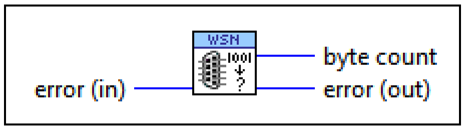
I32 **error out** contains an error code. If **error in** indicates that an error occurred before this VI ran, **error out** contains the same value. Otherwise it contains the error code that this VI produces. The value will be 0 if no error occurred and non-zero if an error occurred.

WSN Serial Port Bytes At Port VI

Owning Palette: Serial

Requires: LabVIEW WSN Module

Returns the number of bytes in the input buffer for the serial port.



U32 **byte count** indicates the number of bytes which were found available to read.

I32 **error in** describes error conditions that occur before this VI runs. The default is no error. If an error occurred before this VI runs, the VI passes the **error in** value to **error out**. This VI runs normally only if no error occurred before this VI runs. If an error occurs while this VI runs, it runs normally and sets its own error status in **error out**. Use **error in** and **error out** to check errors and to specify execution order by wiring **error out** from one node to **error in** of the next node.

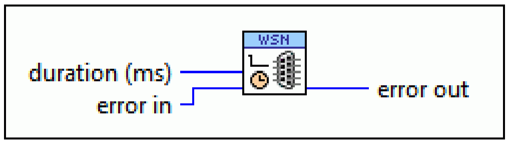
error out contains an error code. If **error in** indicates that an error occurred before this VI ran, **error out** contains the same value. Otherwise it contains the error code that this VI produces. The value will be 0 if no error occurred and non-zero if an error occurred.

WSN Serial Port Break VI

Owning Palette: Serial

Requires: LabVIEW WSN Module

Sends a break on the serial port for a period of time based on the **duration** input.



duration (ms) specifies the amount of time, in milliseconds, to suspend character transmission and place the transmission line in a break state.

error in describes error conditions that occur before this VI runs. The default is no error. If an error occurred before this VI runs, the VI passes the **error in** value to **error out**. This VI runs normally only if no error occurred before this VI runs. If an error occurs while this VI runs, it runs normally and sets its own error status in **error out**. Use **error in** and **error out** to check errors and to specify execution order by wiring **error out** from one node to **error in** of the next node.

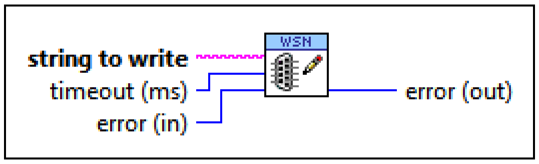
error out contains an error code. If **error in** indicates that an error occurred before this VI ran, **error out** contains the same value. Otherwise it contains the error code that this VI produces. The value will be 0 if no error occurred and non-zero if an error occurred.

WSN Serial Port Write VI

Owning Palette: Serial

Requires: LabVIEW WSN Module

Writes the data in **string to write** to the serial port.



string to write is the data in string to write to the serial port.

timeout (ms) specifies the amount of time, in milliseconds, to wait for this function to finish outputting the data in **string to write**. The default value is 2000.

error in describes error conditions that occur before this VI runs. The default is no error. If an error occurred before this VI runs, the VI passes the **error in** value to **error out**. This VI runs normally only if no error occurred before this VI runs. If an error occurs while this VI runs, it runs normally and sets its own error status in **error out**. Use **error in** and **error out** to check errors and to specify execution order by wiring **error out** from one node to **error in** of the next node.

error out contains an error code. If **error in** indicates that an error occurred before this VI ran, **error out** contains the same value. Otherwise it contains the error code that this VI produces. The value will be 0 if no error occurred and non-zero if an error occurred.

User Calibration VIs

Owning Palette: WSN VIs

Requires: LabVIEW WSN Module.

Use the User Calibration VIs to calibrate LabVIEW WSN strain nodes.

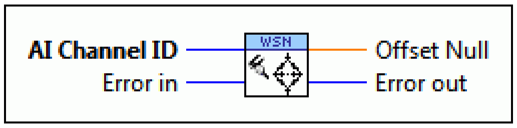
Palette Object	Description
Shunt Calibration VI	Obtains the shunt calibration gain value for Qua rter Bridge gages.
Offset Nulling VI	Obtains the offset value in millivolts that the bri dge is away from 0.

Offset Nulling VI

Owning Palette: User Calibration

Requires: LabVIEW WSN Module

Obtains the offset value in mV/V that the bridge is away from 0.



AI Channel ID is the channel that will be calibrated (0-3).

Offset Null is the value in mV/V that the bridge is away from 0. The offset null value is added to the calibrated ratiometric reading. This value is applied in all bridge configurations and can be also be set or returned using the Analog Input Properties for NI WSN-3214 Strain Node.

error in describes error conditions that occur before this VI runs. The default is no error. If an error occurred before this VI runs, the VI passes the **error in** value to **error out**. This VI runs normally only if no error occurred before this VI runs. If an error occurs while this VI runs, it runs normally and sets its own error status in **error out**. Use **error in** and **error out** to check errors and to specify execution order by wiring **error out** from one node to **error in** of the next node.

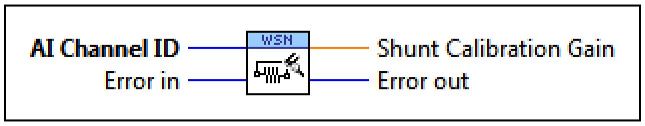
error out contains an error code. If **error in** indicates that an error occurred before this VI ran, **error out** contains the same value. Otherwise it contains the error code that this VI produces. The value will be 0 if no error occurred and non-zero if an error occurred.

Shunt Calibration VI

Owning Palette: User Calibration

Requires: LabVIEW WSN Module

Obtains the shunt calibration gain value by applying a shunt calibration resistor in parallel with the gage and measuring the deflection. This can only be used in Quarter Bridge configurations.



AI Channel ID is the channel that will be calibrated (0-3).

Shunt Calibration Gain is a gain that can be applied to the post-ADC calibrated ratiometric value. Apply this gain using AI**x** Elemental I/O Property Nodes, where **x** is the value of the channel. See also Analog Input Properties for NI WSN-3214 Strain Node.

error in describes error conditions that occur before this VI runs. The default is no error. If an error occurred before this VI runs, the VI passes the **error in** value to **error out**. This VI runs normally only if no error occurred before this VI runs. If an error occurs while this VI runs, it runs normally and sets its own error status in **error out**. Use **error in** and

error out to check errors and to specify execution order by wiring **error out** from one node to **error in** of the next node.

I32 **error out** contains an error code. If **error in** indicates that an error occurred before this VI ran, **error out** contains the same value. Otherwise it contains the error code that this VI produces. The value will be 0 if no error occurred and non-zero if an error occurred.

NI-WSN

September 2012, 372839F-01

This help file contains information about using LabVIEW with the National Instruments Wireless Sensor Network gateways and nodes.

To view related topics, click the **Locate** button, shown at left, in the toolbar at the top of this window. The **LabVIEW Help** highlights this topic in the **Contents** tab so you can navigate the related topics.

To comment on National Instruments documentation, refer to the National Instruments Web site.

NI-WSN Related Documentation

Refer to the following documents for information about using the NI WSN gateway and NI WSN-32**xx** nodes.

Note For an up-to-date list of WSN documentation, go to `ni.com/info` and enter rdwsnrd.

Help Resources

- **LabVIEW Help**—Use these help files to learn how to create and download LabVIEW applications to WSN nodes over a wireless connection and for information about developing and using Vis and projects.
- **Measurement & Automation Explorer Help for WSN**—This help file contains information about configuring your Wireless Sensor Network system using NI Measurement & Automation Explorer (MAX).Configuring WSN in MAX,

available in MAX from **Start»Programs» National Instruments»NI-WSN»Configuring WSN in MAX**.

Refer to the National Instruments Product Manuals Library for updated documentation resources.

PDF Documents

These documents are available as PDFs in the `Documentation\Manuals` directory. The latest versions of these documents are online at ni.com/manuals. You must have Adobe Reader with Search and Accessibility 5.0.5 or later installed to view the PDFs. You must have Adobe Reader with Search and Accessibility 6.**x** or later installed to search PDF versions of these manuals. Refer to the Adobe Systems Incorporated Web site to download Adobe Reader.

- **NI Wireless Sensor Network Devices Getting Started Guide**—This document describes how to install the WSN gateway and nodes, how to use MAX to configure the devices, and how to get started using LabVIEW.
- User guide/specifications for your gateway—This documentation contains system-level information about the WSN gateway, including device features, mounting information, and specifications.
- **NI WSN-32xx User Guide and Specifications**—These documents contain information about WSN nodes, including features, mounting information, and specifications.

Readme

- **NI-WSN Readme**—This document contains last-minute information about WSN devices, including a list of known issues. Access the **NI-WSN Readme** on the NI-WSN installation CD.

Searching PDF Versions of WSN Manuals

Use Adobe Reader with Search and Accessibility 6.**x** or later to search PDF versions of all the WSN manuals. Refer to the Adobe Systems Incorporated Web site to download Adobe Reader.

Complete the following steps to search all the PDF versions of WSN manuals.

1. In Adobe Reader, select **Edit»Search** to display the **Search PDF** window.
2. Enter a word or phrase in the **What word or phrase would you like to search for** text box.
3. Click the **All PDF Documents in** button and select **Browse for Location** from the drop-down list. The **Browse for Folder** dialog box appears.
    1. Navigate to the `WSN\manuals` directory.
    2. Click the **OK** button to close the dialog box and return to the **Search PDF** window.
4. Click the **Search** button.

Refer to the Adobe Reader Help for more information about searching all the PDF documents in a directory for a word or phrase.

Supported WSN Hardware

WSN Gateways

- NI WSN-9791 Ethernet Gateway
- NI 9792 Programmable WSN Gateway
- NI WSN-9795 C Series WSN Gateway

WSN Nodes

- NI WSN-3202 ±10 V, 4-Channel, 16-Bit Analog Input Node
- NI WSN-3212 4-Channel, 24-Bit Thermocouple Input Node
- NI WSN-3226 4-Channel, 20-Bit Voltage/RTD Node
- NI WSN-3214 Strain Node
- NI WSN-3230/3231 Serial Node

NI WSN-3202 Analog Input Node

WSN ±10 V, 4-Channel, 16-Bit Analog Input Node

Node I/O Variables

To use I/O from this node in a VI, drag and drop I/O variables from the **Project Explorer** window to the VI block diagram. The I/O variables for the channels return calibrated floating-point voltage, in volts.

Node Channels

The NI WSN-3202 has the following channels.

Channel	Type	Description
AI**x**	Read	Analog input channel **x**, where **x** is the number of the channel. For the NI WSN-3202, **x** is 0 to 3.
DIO**x**	Read/Write	Digital input/output channel **x**, where **x** is the number of the ch annel. For the NI WSN-3202, **x** is 0 to 3.
Battery Voltage	Read	Returns the voltage, in volts, of the batteries installed in the de vice.
Link Quality	Read	Returns the link quality percent age of the radio link.
VI Deployed	Read	Returns **True** if a LabVIEW WSN application has been deployed to the node.
External Power	Read	Returns **True** if external power i s connected to the device.
Mesh Router	Read	Returns **True** if the device is co nfigured as a Mesh Router Node . Returns **False** if the device is c onfigured as an End Node.

NI WSN-3202 Settings

Right-click the node in the **Project Explorer** window and select **Properties**. Select the **Channels** or **Node** tab.

To upload the serial number of a connected node to the **Project Explorer** window, right-click the gateway and select **Refresh Node Serial Numbers**.

Channels Tab: Analog Input

This dialog box includes the following components:

- **Channels**—Specifies the channel or channels to configure.
- **Channel Configuration**
  - **Range**—Specifies the voltage range of the selected channel(s):
    - -10 to 10 Volts
    - -5 to 5 Volts
    - -2 to 2 Volts
    - -0.5 to 0.5 Volts
- **Channel Attributes**
  - **Attribute**—You cannot change this setting.
  - **Value**—You cannot change this setting

Channels Tab: Digital Input/Output

This dialog box includes the following components:

- **Channels**—Specifies the channel or channels to configure.
- **Channel Configuration**
  - **Range**—You cannot change this setting.
- **Channel Attributes:**

Attribute	Value
DIO Drive Mode	Specifies the channel state: **Tristate**—The line is not driven. **Drive Low Only**—When the DIO line is off, the output line drives low. When the DIO line is on, the output line tristates.

	Drive High Only—When the DIO line is on, the output line drives high. When the DIO line is off, the output line tristates. **Drive High and Low**—When the DIO line is on, the output line drives high. When the DIO line is off, the output line drives low.

Node Tab: Analog Input and Digital Input/Output

This dialog box includes the following components:

- **Device Name**—Specifies the name of the WSN node, which appears in the **Project Explorer** window. LabVIEW assigns a default name to the node based on the Hardware Configuration ID. You can use this field to give the node a descriptive name.
- **Device Type**—Specifies the type of WSN node. You cannot change this value.
- **Hardware Configuration**—Specifies the hardware configuration ID and the sample interval (seconds).
  - **ID**—Allows you to map a node in the property tree to a node of that ID on the gateway.
  - **Sample Interval (Seconds)**—Specifies how often the node samples all inputs, in seconds. For End Nodes, the nodes sleep during this interval.
  - **Sensor Power—**Specifies the sensor excitation delay time:
    - **0 ms before sampling**—turns on the Sensor Power immediately when the analog acquisition starts.
    - **25 ms before sampling**—turns on Sensor Power 25 ms before the analog acquisition starts.
    - **100 ms before sampling**—turns on Sensor 100 ms before the analog acquisition starts.
    - **250 ms before sampling**—turns on Sensor Power 250 ms before the analog acquisition starts.
    - **Always On**—turns on Sensor Power when the next analog acquisition starts and leaves it on indefinitely.

Note Enabling Sensor Power diminishes battery life.

NI WSN-3212 Thermocouple Input Node

WSN 4-Channel, 24-Bit Thermocouple Input Node

Node I/O Variables

To use I/O from this node in a VI, drag and drop I/O variables from the **Project Explorer** window to the VI block diagram. The I/O variables for the channels return data in units as specified in the **WSN Node Properties** dialog box.

Node Channels

The NI WSN-3212 has the following channels.

Channel	Type	Description
TC**x**	Read	Thermocouple input channel **x**, where **x** is the number of the ch annel. For the NI WSN-3212, **x** is 0 to 3.
DIO**x**	Read/Write	Digital input/output channel **x**, where **x** is the number of the ch annel. For the NI WSN-3212, **x** is 0 to 3.
Battery Voltage	Read	Returns the voltage, in volts, of the batteries installed in the de vice.
Link Quality	Read	Returns the link quality percent age of the radio link.
VI Deployed	Read	Returns **True** if a LabVIEW WSN application has been deployed to the node.
External Power	Read	Returns **True** if external power i s connected to the device.
Mesh Router	Read	Returns **True** if the device is co nfigured as a Mesh Router Node

		. Returns **False** if the device is c onfigured as an End Node.

NI WSN-3212 Settings

Right-click the node in the **Project Explorer** window and select **Properties**. Select the **Channels** or **Node** tab.

To upload the serial number of a connected node to the **Project Explorer** window, right-click the gateway and select **Refresh Node Serial Numbers**.

Channels Tab: Analog Input

This dialog box includes the following components:

- **Channels**—Specifies the channel or channels to configure.
- **Channel Configuration**
  - **Range**—Specifies the temperature or voltage range:
    - -273.15 to 1,820 Celsius
    - 0 to 2,093 Kelvin
    - -459.67 to 3,308 Fahrenheit
    - -0.073 to 0.073 Volts
- **Channel Attributes**

Attribute	Value
Thermocouple Type	J, K, T, E, R, S, N, or B
CJC (Cold Junction Compensation) Source	Internal CJC 0 C 25 C

Channels Tab: Digital Input/Output

This dialog box includes the following components:

- **Channels**—Specifies the channel or channels to configure.

- **Channel Configuration**
  - **Range**—You cannot change this setting.
- **Channel Attributes:**

Attribute	Value
DIO Drive M ode	Specifies the channel state: **Tristate**—The line is not driven. **Drive Low Only**—When the DIO line is off, the output line drives low. When the DIO line is on, the output line tristates. **Drive High Only**—When the DIO line is on, the output line drives high. When the DI O line is off, the output line tristates. **Drive High and Low**—When the DIO line is on, the output line drives high. When th e DIO line is off, the output line drives low.

Node Tab: Analog Input and Digital Input/Output

This dialog box includes the following components:

- **Device Name**—Specifies the name of the WSN node, which appears in the **Project Explorer** window. LabVIEW assigns a default name to the node based on the Hardware Configuration ID. You can use this field to give the node a descriptive name.
- **Device Type**—Specifies the type of WSN node. You cannot change this value.
- **Hardware Configuration**—Specifies the hardware configuration ID and the sample interval (seconds).
  - **ID**—Allows you to map a node in the property tree to a node of that ID on the gateway.
  - **Sample Interval (Seconds)**—Specifies how often the node samples all inputs, in seconds. For End Nodes, the nodes sleep during this interval.

NI WSN-3214 Strain Node

WSN Full/Half/Quarter Bridge, 4-Channel, 20-Bit Strain Input Node

Node I/O Variables

To use I/O from this node in a VI, drag and drop I/O variables from the **Project Explorer** window to the VI block diagram.

Node Channels

The NI WSN-3214 has the following channels.

Channel	Type	Description
AI**x**	Read	Waveform analog input channe l **x**, where **x** is the number of th e channel. For the NI WSN-3214 , **x** is 0 to 3.
DIO**x**	Read/Write	Digital input/output channel **x**, where **x** is the number of the ch annel. For the NI WSN-3214, **x** is 0 to 1.
Battery Voltage	Read	Returns the voltage, in volts, of the batteries installed in the de vice.
Link Quality	Read	Returns the link quality percent age of the radio link.
VI Deployed	Read	Returns **True** if a LabVIEW WSN application has been deployed to the node.
External Power	Read	Returns **True** if external power i s connected to the device.
Mesh Router	Read	Returns **True** if the device is co nfigured as a Mesh Router Node . Returns **False** if the device is c onfigured as an End Node.

NI WSN-3214 Settings

Right-click the node in the **Project Explorer** window and select **Properties**.

To upload the serial number of a connected node to the **Project Explorer** window, right-click the gateway and select **Refresh Node Serial Numbers**.

Waveform Analog Input Properties

Digital Input/Output Properties

Node Properties

NI WSN-3214 Waveform Analog Input Properties

Select the channel or channels to configure. This dialog box includes the following components:

Channel Configuration

Default = Full Bridge – Strain Type I

Bridge Type	Description
Full Bridge – Strain Type I	Four active strain gage elements. Two are moun ted in the direction of bending strain on the top side of the strain specimen and the other two m ounted in the direction of bending strain on the bottom side. See also Strain Gages and Strain G age Bridge Configurations.
Full Bridge – Strain Type II	Four active strain gage elements. Two are moun ted in the direction of **bending** strain with one o n the top side of the strain specimen and the ot her on the bottom side. The other two act toget her as a Poisson gage and are mounted transver se, or perpendicular, to the principal axis of strai n with one on the top side of the strain specime n and the other on the bottom side. See also Str ain Gages and Strain Gage Bridge Configuration s.

Full Bridge – Strain Type III	Four active strain gage elements. Two are mounted in the direction of **axial** strain with one on the top side of the strain specimen and the other on the bottom side. The other two act together as a Poisson gage and are mounted transverse, or perpendicular, to the principal axis of strain with one on the top side of the strain specimen and the other on the bottom side. See also Strain Gages and Strain Gage Bridge Configurations.
Half Bridge – Strain Type I	Two active strain gage elements, one mounted in the direction of axial strain and the other acting as a Poisson gage and mounted transverse, or perpendicular, to the principal axis of strain. See also Strain Gages and Strain Gage Bridge Configurations.
Half Bridge – Strain Type II	Two active strain gage elements, one mounted in the direction of axial strain on the top side of the strain specimen and the other mounted in the direction of axial strain on the bottom side. See also Strain Gages and Strain Gage Bridge Configurations.
Quarter Bridge – Strain Type I	A single active strain gage element mounted in the principle direction of axial or bending strain. See also Strain Gages and Strain Gage Bridge Configurations.
Full Bridge – Ratiometric	Four active bridge elements with an output in millivolts per volt (mV/V). See also Strain Gages and Strain Gage Bridge Configurations.
Half Bridge – Ratiometric	Two active bridge elements with an output in mV/V. See also Strain Gages and Strain Gage Bridge Configurations.
Quarter Bridge – Ratiometric	One active bridge element with an output in mV/V. See also Strain Gages and Strain Gage Bridge Configurations.
Channel Disabled	Disabled channels are not sampled and are not sent to the gateway.

Sensor Properties

Property	Value/Description

Range	Microstrain (με) when Strain, mV/V when Ratiometric. The default value is **Microstrain (με)**.
Gage Factor	Returns or sets the Gage Factor. The Gage factor specifies the sensitivity of the strain gages and relates the change in electrical resistance to the change in strain. Each gage in the bridge must have the same gage factor. Refer to the sensor documentation to determine this value. The default value is **2**. Possible inputs are: ▪ 0 < Gage Factor < 800
Calibration, Offset Null (mV/V)	The Offset Null value is added to the calibrated ratiometric reading. It can be measured by clicking **Run** in the node properties window for each channel. This value is applied automatically in all bridge configurations. The default value is **0**. Possible inputs are: ▪ -25 to +25 mV/V
Poisson Ratio	Sets the Poisson Ratio. Poisson ratio is the ratio of lateral strain to axial strain in the material you are measuring. The default value is **0**. Possible inputs are: ▪ -1 to +1
Quarter Bridge Gage Resistance	Sets the Quarter Bridge Gage Resistance. Quarter bridge gage resistance is the resistance in ohms of the gage in an unstrained position. The default value is **350**. Possible inputs are: ▪ 350 Ohms ▪ 1000 Ohms
Shunt Calibration (1/Actual Gain)	The Shunt Calibration value is a gain applied to the post-ADC calibrated ratiometric value. This value will always be applied regardless of bridge

	configuration, as it can be calculated and/or me asured through external means. This value is ap plied in all configurations. The default value is **1** . **Note** The hardware can only perform a sh unt calibration in Quarter Bridge mode. It can be measured by clicking **Run** in the n ode properties window for each channel. Shunt calibration must be done before an offset null calibration. Possible inputs are: ▪ 0 – 2

NI WSN-3214 Digital Input/Output Properties

Select the channel or channels to configure. This dialog box includes the following components:

Property	Value/Description
DIO Mode	Returns or sets the mode for the corresponding DIO lines. The default value is **Host Driven**. Thi s property can contain the following values: ▪ **DI - 24V Sinking**—Use this mode to co nnect to industrial 24V sourcing output de vices. The input has a low impedance to g round and input thresholds compatible wi th 24 V signaling. ▪ **DI - 24V Sinking with Power Manag ement**—This mode is similar to DI-24V Si nking, but the low impedance path to gro und is removed when the inputs are not b eing actively read. This may reduce power consumption of the output device. It shou ld only be used with output devices that h ave valid output states within 100 µs of ha

ving a low impedance path presented to t he output.

- **DI - TTL Logic**—High impedance input s with TTL compatible thresholds; suitabl e for most 3 V or 5 V logic signals.
- **DI - Contact Closure**—The input has T TL compatible thresholds and provides a pull-up resistor to 3 V when the input is ac tively being read. This is suitable for conn ections to contact switches wired betwee n the input and ground. If a contact to gro und remains closed in this setting while DI O Notifications are not set to Disabled, the current into the contact switch increases power consumption and can reduce batte ry life.
- **DO - Drive Low (Sinking)**—For DIO o utput value 1, this mode does not drive th e DIO line. For DIO output value 0, this mo de drives the DIO line low. This setting do es not require a supply voltage on DIO_P WR.
- **DO - Drive High (Sourcing)**—For DIO output value 1, this mode drives the DIO li ne high to the DIO_PWR voltage. For DIO o utput value 0, this mode does not drive th e DIO line. This setting requires a supply v oltage connected to the DIO_PWR pin.
- **DO - 3V TTL Logic** —For DIO output va lue 1, this mode pulls-up the DIO line to 3 V through a pull-up resistor. For DIO outpu t value 0, this mode drives the DIO line lo w. Use this mode to connect to 3 V and 5 V logic inputs with TTL compatible input thr esholds. This setting does not require a su pply voltage on DIO_PWR.
- **DO - Drive High and Low** —For DIO o utput value 1, this mode drives the DIO lin e high to the DIO_PWR voltage. For DIO ou tput value 0, this mode drives the DIO line

	low. This setting requires a supply voltage connected to the DIO_PWR pin in order to drive high. If no supply is connected to DIO_PWR, this mode will operate as if it were set to the DO-Drive Low (Sinking) mode.

NI WSN-3214 Node Properties

This dialog box includes the following components:

WSN Configuration

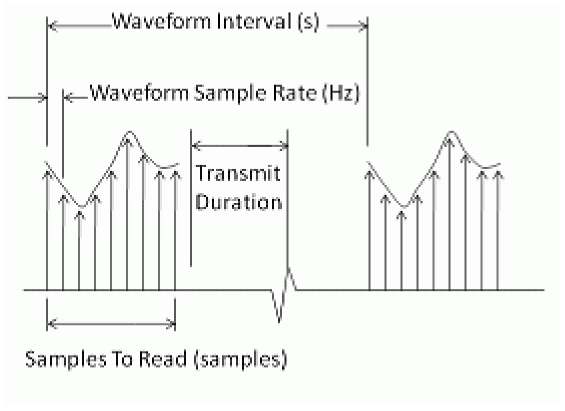
Property	Value/Description
Device Name	Specifies the name of the WSN node, which appears in the **Project Explorer** window. LabVIEW assigns a default name to the node based on the Hardware Configuration ID. You can use this field to give the node a descriptive name.
Device Type	Specifies the type of WSN node. You cannot change this value.
ID	Allows you to map a node in the property tree to a node of that ID on the gateway.

Waveform Acquisition

Property	Value/Description
Waveform Interval (Seconds)	Waveform interval is time between successive waveform acquisitions, in seconds. The default value is **5**. Possible inputs are: ▪ 0.1 to 100000 seconds
Waveform Sample Rate (Hz)	Waveform sample rate is the rate at which samples are taken within a single waveform. The default value is **10**. Possible inputs are:

	▪ 1 to max rate that hardware supports (S PS/ch). Sample rates depend on filtering s ettings. See sample rates.
Samples to Read	Samples to read is the number of samples withi n a given waveform. The default value is **10**. Possible inputs are: ▪ 1 to 8192 Samples

The following diagram shows how these three properties work together:



For each sample, one sample is taken for each channel that is active. This means that the waveform shown in the diagram can be replicated up to 4 times, depending on which channels are enabled. These waveforms overlap in time, with a small offset between them (the settling time for each channel).

Waveform Filtering

Property	Value/Description
Powerline Filtering	When set to None, Powerline Filtering will avera ge a series of ADC values to arrive at a single sa mple value. The time spent doing this can be set using the Aperture Time attribute. The default v alue is **50/60 Hz**. Possible inputs are: ▪ 50 Hz ▪ 60 Hz ▪ 50/60 Hz

	▪ None **Note** Filtering has an effect on maximum waveform sampling rate. See sample rates .
Filtering Strength	Sets or returns the filtering strength for the nod e. The default is **High Rejection**. **Note** Powerline Filtering must be enabled for filtering strength to take effect. Possible inputs are: ▪ High Rejection—Rejects more powerlin e noise at the expense of higher energy co nsumption. ▪ Low Power—Rejects powerline noise, b ut as energy efficiently as possible. **Note** Filtering strength has an effect on m aximum waveform sampling rate. See sam ple rates.
Aperture Time	Specifies the period during which the ADC is rea ding the input signal. The default value is **1.4 m s**. **Note** Aperture Time can only be edited w hen Powerline Filtering is set to None. Possible inputs are: ▪ 250 us ▪ 400 us ▪ 750 us ▪ 1.4 ms ▪ 2.8 ms ▪ 5.5 ms ▪ 10.8 ms ▪ 21.5 ms

Note Aperture time has an effect on maximum waveform sampling rate. See sample rates.

NI WSN-3214 Sample Rates

The following tables show sample rates for the NI WSN-3214 based on the powerline filtering setting or the aperture time setting.

Powerline Filtering

If you use the powerline filtering setting, the following waveform sample rates apply.

Filter	Max Sample Rate (Hz) (N number of channels)
60 Hz Low Power	59
50 Hz Low Power	49
60 Hz High Rejection	30
50 Hz High Rejection	25
50/60 Hz Low Power	27
50/60 Hz High Rejection	13

Aperture Time

If you use the aperture time setting, the following waveform sample rates apply.

Note Aperture time is only used if powerline filtering is set to None.

Channels	1	2	3	4
Aperture (us)	Max Sample Rate (Hz)	Max Sample Rate (Hz)	Max Sample Rate (Hz)	Max Sample Rate (Hz)
250	4096	1560	1092	862
400	2340	992	697	537
750	1260	585	404	306
1400	668	321	217	164
2800	344	168	113	85

5500	175	86	57	43
10800	88	43	29	22
21500	44	22	14	11

Strain Gages

You can measure strain with a strain gage, which is a device with electrical resistance that varies in proportion to the amount of strain in the device, and with signal conditioning. When using a strain gage, you bond the strain gage to the device under test, apply force, and measure the strain by detecting changes in resistance (Ω). Strain gages return varying voltages in response to stress or vibrations in materials. Resistance changes in parts of the strain gage to indicate deformation of the material. Strain gages require excitation, generally voltage excitation, and linearization of the voltage measurements.

Strain measurements rarely involve quantities larger than a few microstrain (µε). Therefore, measuring strain requires accurate measurements of very small changes in resistance. For example, if a test specimen undergoes a substantial strain of 500 µε, a strain gage with a gage factor of 2 exhibits a change in electrical resistance of only $2 \times (500 \times 10^{-6}) = 0.1\%$. For 120 Ω, this is a change of only 0.12 Ω.

To measure such small changes in resistance and to compensate for temperature sensitivity, strain gages often use a Wheatstone bridge with a voltage or current excitation source, arranged in one of several bridge configurations. The gage is the collection of all of the active elements of the Wheatstone bridge.

WSN supports measuring axial strain, bending strain, or both. While you can use some similar configuration types to measure torsional strain, NI software scaling does not support these configuration types. It is possible to use NI products to measure torsional strain, but to properly scale these configuration types you must use a custom scale with a bridge (V/V) or a custom voltage with excitation channel.

Gage Factor

A fundamental parameter of the strain gage is its sensitivity to strain, expressed quantitatively as the gage factor (GF). Gage factor is the ratio of the fractional change in electrical resistance to the fractional change in length, or strain. The gage factor must be the same for each gage in the bridge.

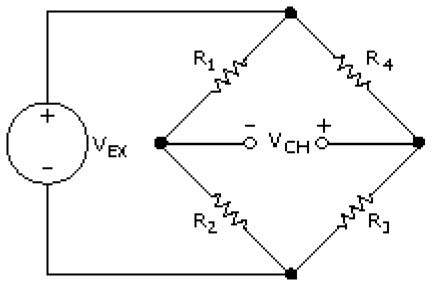
The gage factor for metallic strain gages is usually around 2. You can obtain the actual gage factor of a particular strain gage from the sensor vendor or sensor documentation.

Nominal Gage Resistance

Nominal gage resistance is the resistance of a strain gage in an unstrained position. You can obtain the nominal gage resistance of a particular gage from the sensor vendor or sensor documentation.

Bridge-Based Sensors

Bridge-based sensors operate by correlating a physical phenomena, such as strain, temperature, or force, to a change in resistance in one or more legs of a Wheatstone bridge. The general Wheatstone bridge, shown in the following figure, is a network of four resistive legs with an excitation voltage, V_{EX}, that is applied across the bridge. One or more of these legs can be active sensing elements.



The Wheatstone bridge is the electrical equivalent of two parallel voltage divider circuits. R1 and R2 compose one voltage divider circuit, and R4 and R3 compose the second voltage divider circuit. You measure the output of a Wheatstone bridge between the middle nodes of the two voltage dividers.

A physical phenomena, such as a temperature shift or a change in strain applied to a specimen, changes the resistance of the sensing elements in the Wheatstone bridge. You can use the Wheatstone bridge configuration to help measure the small variations in resistance that the sensing elements produce corresponding to a physical change in the specimen.

Bridge Sensor Scaling

Measurements from a bridge-based sensor are based on a ratio of measured voltage to excitation voltage. The following equation is used to calculate that ratio:

$$V_R = \frac{(V - V_{IB}) \times G}{V_{EX}}$$

where V_R is the voltage ratio; V is the voltage output from the bridge; V_{IB} is the initial bridge voltage, as determined by offset nulling; V_{EX} is the excitation voltage supplied to the bridge; and G is the gain adjustment from shunt calibration.

Ratiometric devices divide the voltage output from the bridge by the excitation voltage in hardware. Therefore, V/V_{EX} must be within the device range of the device. On voltage devices, the voltage output from the bridge must be within the device range. The initial bridge voltage and gain adjustment affect the association between device range and input limits. For example, on a device that can measure ±5 V, an initial bridge voltage of 1 V means that the minimum and maximum input limits must correspond to -6 V to 4 V.

Strain Gage Bridge Configurations

Connect strain gages in a variation on a generic Wheatstone bridge configuration. These configurations vary based on the placement of strain gages within the bridge; their location and orientation on the material you want to measure; and whether the gages are active sensing gages or dummy gages, used for temperature compensation.

- Full-Bridge Ratiometric
- Full-Bridge Type I
- Full-Bridge Type II
- Full-Bridge Type III
- Half-Bridge Ratiometric
- Half-Bridge Type I
- Half-Bridge Type II

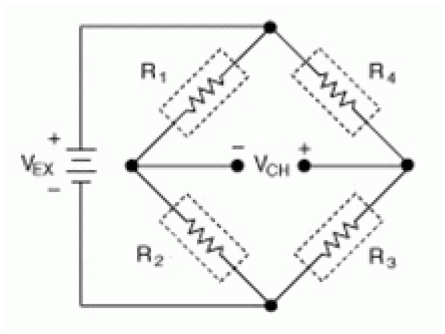
- Quarter-Bridge Ratiometric
- Quarter-Bridge Type I

Full-Bridge Ratiometric

Full-bridge Ratiometric sensors have the following characteristics:

- Four active resistive elements.
- Compensation for lead resistance.
- Output is in mV/V.

Full-Bridge Ratiometric Circuit Diagram

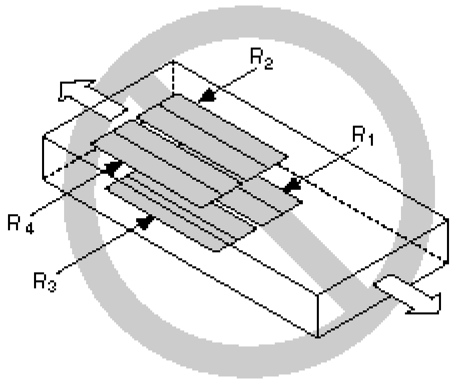


The following symbols apply to the circuit diagram:

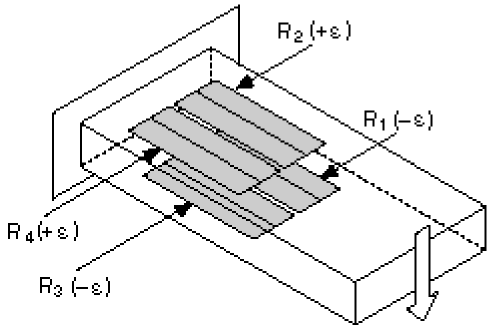
- R_1 is the active resistive element.
- R_2 is the active resistive element.
- R_3 is the active resistive element.
- R_4 is the active resistive element.
- V_{EX} is the excitation voltage.
- V_{CH} is the measured voltage.

Full-Bridge Type I

The full-bridge type I configuration only measures the bending strain.



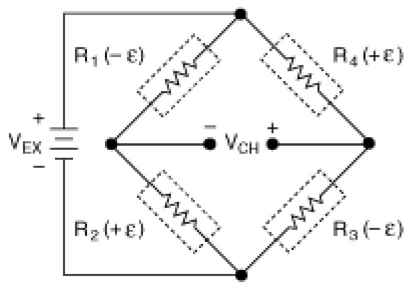
The following figure shows how to position strain gage resistors in a bending configuration for the full-bridge type I.



Full-bridge type I strain gage configurations have the following characteristics:

- Four active strain gage elements, two mounted in the direction of bending strain on the top side of the strain specimen and the other two mounted in the direction of bending strain on the bottom side.
- High sensitivity to bending strain.
- Rejection of axial strain.
- Compensation for temperature.
- Compensation for lead resistance.
- Sensitivity at 1000 με is ~ 2.0 mV_{out} / V_{EX} input.

Full-Bridge Type I Circuit Diagram



The following symbols apply to the circuit diagram:

- R_1 is the active strain gage element measuring compressive strain (–ε).
- R_2 is the active strain gage element measuring tensile strain (+ε).
- R_3 is the active strain gage element measuring compressive strain (–ε).
- R_4 is the active strain gage element measuring tensile strain (+ε).
- V_{EX} is the excitation voltage.
- R_L is the lead resistance.
- V_{CH} is the measured voltage.

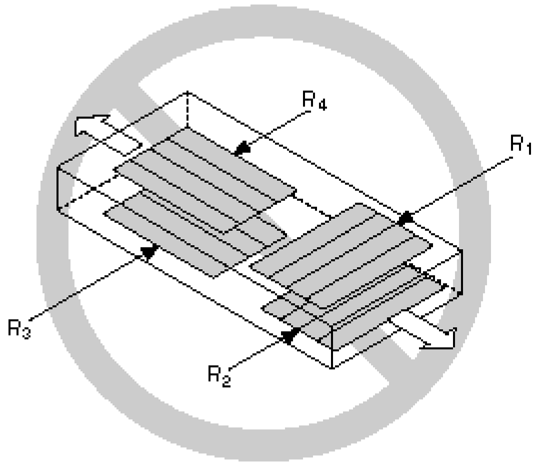
The following equation converts voltage ratios to strain units for full-bridge type I configurations.

$$\text{Strain}(\varepsilon) = \frac{-V_r}{GF}$$

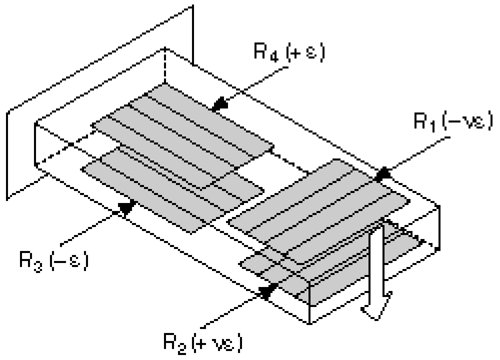
where V_r is the voltage ratio that virtual channels use in the voltage-to-strain conversion equation, and GF is the gage factor.

Full-Bridge Type II

The full-bridge type II configuration only measures bending strain.



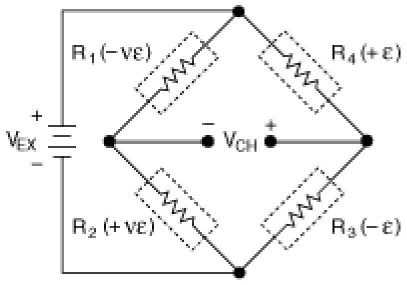
The following figure shows how to position strain gage elements in a bending configuration for the full-bridge type II.



Full-bridge type II strain gage configurations have the following characteristics:

- Four active strain gage elements. Two are mounted in the direction of bending strain with one on the top side of the strain specimen and the other on the bottom side. The other two act together as a Poisson gage and are mounted transverse, or perpendicular, to the principal axis of strain with one on the top side of the strain specimen and the other on the bottom side.
- Rejection of axial strain.
- Compensation for temperature.
- Compensation for the aggregate effect on the principle strain measurement due to the Poisson's ratio of the material.
- Compensation for lead resistance.
- Sensitivity at 1000 με is ~ 1.3 mV_{out} / V_{EX} input.

Full-Bridge Type II Circuit Diagram



The following symbols apply to the circuit diagram:

- R_1 is the active strain gage element measuring compressive Poisson effect (–ε).
- R_2 is the active strain gage element measuring tensile Poisson effect (+ε).
- R_3 is the active strain gage element measuring compressive strain (–ε).
- R_4 is the active strain gage element measuring tensile strain (+ε).
- V_{EX} is the excitation voltage.
- R_L is the lead resistance.
- V_{CH} is the measured voltage.

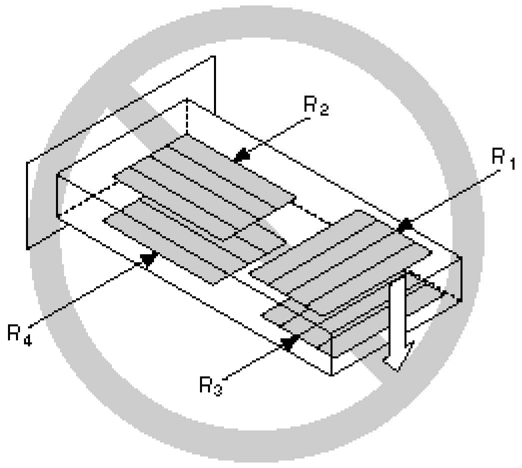
The following equation converts voltage ratios to strain units for full-bridge type II configurations.

$$\text{Strain}(\varepsilon) = \frac{-2V_r}{GF(v+1)}$$

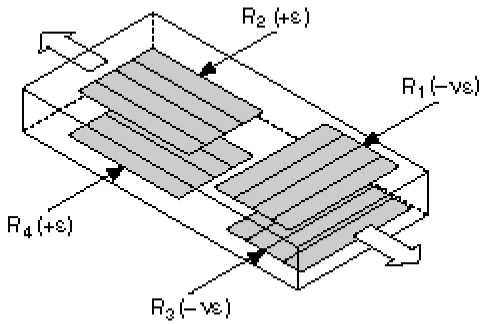
where V_r is the voltage ratio that virtual channels use in the voltage-to-strain conversion equation, GF is the gage factor, and v is the Poisson's ratio.

Full-Bridge Type III

The full-bridge type III configuration only measures the axial configuration.



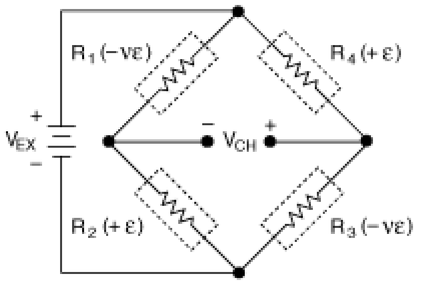
The following figure shows how to position strain gage resistors in an axial configuration for the full-bridge type III.



Full-bridge type III strain gage configurations have the following characteristics:

- Four active strain gage elements. Two are mounted in the direction of axial strain with one on the top side of the strain specimen and the other on the bottom side. The other two act together as a Poisson gage and are mounted transverse, or perpendicular, to the principal axis of strain with one on the top side of the strain specimen and the other on the bottom side.
- Compensation for temperature.
- Rejection of bending strain.
- Compensation for the aggregate effect on the principle strain measurement due to the Poisson's ratio of the material.
- Compensation for lead resistance.
- Sensitivity at 1000 με is ~ 1.3 mV_{out} / V_{EX} input.

Full-Bridge Type III Circuit Diagram



The following symbols apply to the circuit diagram:

- R_1 is the active strain gage element measuring compressive Poisson effect (–ε).
- R_2 is the active strain gage element measuring tensile strain (+ε).
- R_3 is the active strain gage element measuring compressive Poisson effect (–ε).
- R_4 is the active strain gage element measuring the tensile strain (+ε).
- V_{EX} is the excitation voltage.
- R_L is the lead resistance.
- V_{CH} is the measured voltage.

The following equation converts voltage ratios to strain units for full-bridge type III configurations.

$$\text{Strain}(\varepsilon) = \frac{-2V_r}{GF[(v+1)-V_r(v-1)]}$$

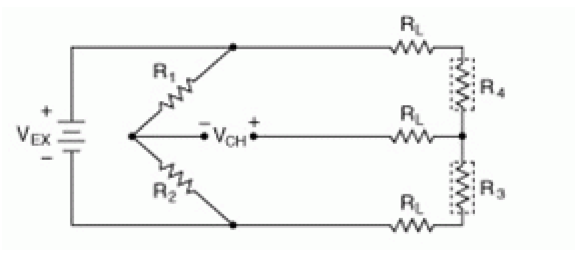
where V_r is the voltage ratio that virtual channels use in the voltage-to-strain conversion equation, GF is the gage factor, and v is the Poisson's ratio.

Half-Bridge Ratiometric

Half-bridge Ratiometric sensors have the following characteristics:

- Two active strain gage elements.
- Completion resistors which provide half-bridge completion.
- Output is in mV/V.

Half-Bridge Ratiometric Circuit Diagram

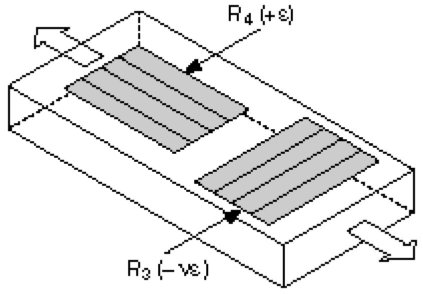


The following symbols apply to the circuit diagram:

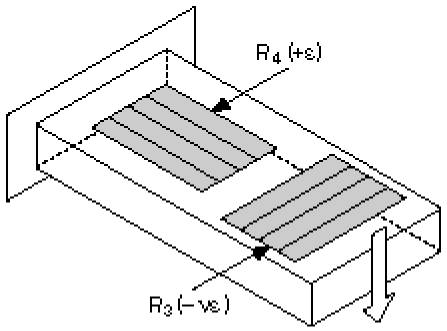
- R_1 is the half-bridge completion resistor.
- R_2 is the half-bridge completion resistor.
- R_3 is the active resistive element.
- R_4 is the active resistive element.
- V_{EX} is the excitation voltage.
- R_L is the lead resistance.
- V_{CH} is the measured voltage.

Half-Bridge Type I

The following figure shows how to position strain gage resistors in an axial configuration for the half-bridge type I.



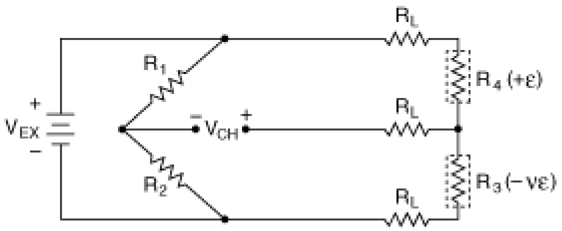
The following figure shows how to position strain gage resistors in a bending configuration for the half-bridge type I.



Half-bridge type I strain gage configurations have the following characteristics:

- Two active strain gage elements, one mounted in the direction of axial strain and the other acting as a Poisson gage and mounted transverse, or perpendicular, to the principal axis of strain.
- Completion resistors which provide half-bridge completion.
- Sensitivity to both axial and bending strain.
- Compensation for temperature.
- Compensation for the aggregate effect on the principle strain measurement due to the Poisson's ratio of the material.
- Sensitivity at 1000 με is ~ 0.65 mV_{out}/ V_{EX} input.

Half-Bridge Type I Circuit Diagram



The following symbols apply to the circuit diagram:

- R_1 is the half-bridge completion resistor.
- R_2 is the half-bridge completion resistor.
- R_3 is the active strain gage element measuring compression due to the Poisson effect (−ε).
- R_4 is the active strain gage element measuring tensile strain (+ε).

- V_{EX} is the excitation voltage.
- R_L is the lead resistance.
- V_{CH} is the measured voltage.

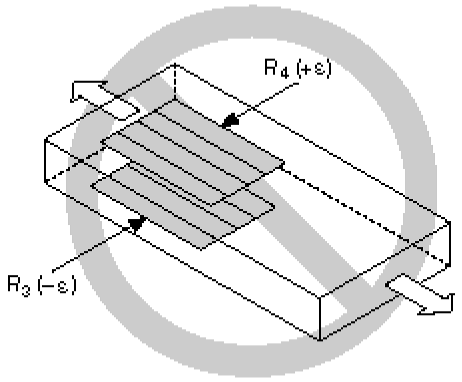
The following equation converts voltage ratios to strain units for half-bridge type I configurations.

$$\text{Strain}(\varepsilon) = \frac{-4V_r}{GF[(1+v)-2V_r(v-1)]} \cdot \left(1+\frac{R_L}{R_g}\right)$$

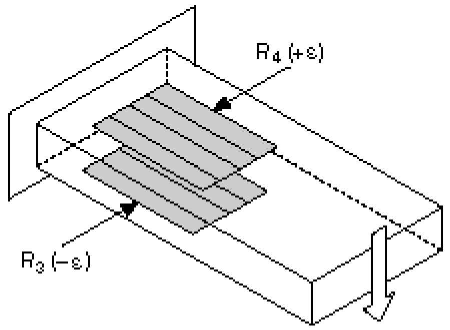
where V_r is the voltage ratio that virtual channels use in the voltage-to-strain conversion equation, GF is the gage factor, v is the Poisson's ratio, R_L is the lead resistance, and R_g is the nominal gage resistance.

Half-Bridge Type II

The half-bridge type II configuration only measures bending strain.



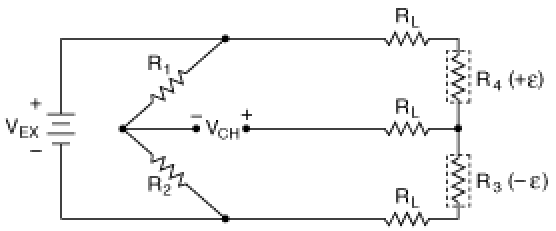
The following figure shows how to position strain gage resistors in a bending configuration for the half-bridge type II.



Half-bridge type II strain gage configurations have the following characteristics:

- Two active strain gage elements, one mounted in the direction of axial strain on the top side of the strain specimen and the other mounted in the direction of axial strain on the bottom side.
- Completion resistors which provide half-bridge completion.
- Sensitivity to bending strain.
- Rejection of axial strain.
- Compensation for temperature.
- Sensitivity at 1000 με is ~ 1 mV_{out} / V_{EX} input.

Half-Bridge Type II Circuit Diagram



The following symbols apply to the circuit diagram:

- R_1 is the half-bridge completion resistor.
- R_2 is the half-bridge completion resistor.
- R_3 is the active strain gage element measuring compressive strain (–ε).
- R_4 is the active strain gage resistor measuring tensile strain (+ε).
- V_{EX} is the excitation voltage.
- R_L is the lead resistance.
- V_{CH} is the measured voltage.

The following equation converts voltage ratios to strain units for half-bridge type II configurations.

$$\text{Strain}(\varepsilon) = \frac{-2V_r}{GF} \bullet \left(1 + \frac{R_L}{R_g}\right)$$

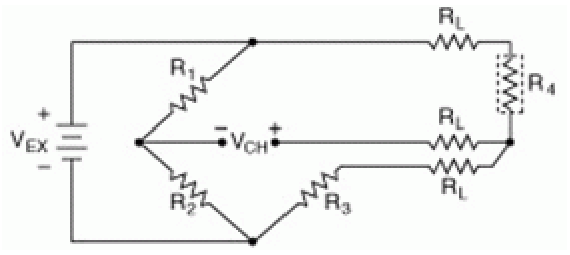
where V_r is the voltage ratio that virtual channels use in the voltage-to-strain conversion equation, GF is the gage factor, R_L is the lead resistance, and R_g is the nominal gage resistance.

Quarter-Bridge Ratiometric

Quarter-bridge Ratiometric sensors have the following characteristics:

- A single active resistive element.
- A passive quarter-bridge completion resistor, known as a dummy resistor, in addition to half-bridge completion.
- Output is in mV/V.

Quarter-Bridge Ratiometric Circuit Diagram

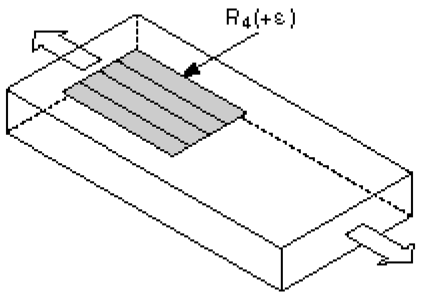


The following symbols apply to the circuit diagram:

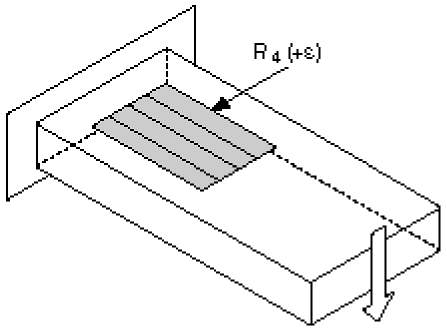
- R_1 is the half-bridge completion resistor.
- R_2 is the half-bridge completion resistor.
- R_3 is the quarter-bridge completion resistor, known as a dummy resistor.
- R_4 is the active resistive element.
- V_{EX} is the excitation voltage.
- R_L is the lead resistance.
- V_{CH} is the measured voltage.

Quarter-Bridge Type I

The following figure shows how to position a strain gage resistor in an axial configuration for the quarter-bridge type I.



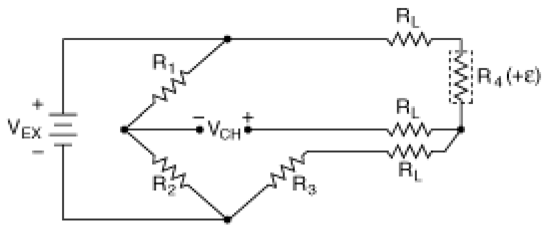
The following figure shows how to position a strain gage resistor in a bending configuration for the quarter-bridge type I.



Quarter-bridge type I strain gage configurations have the following characteristics:

- A single active strain gage element mounted in the principle direction of axial or bending strain.
- A passive quarter-bridge completion resistor, known as a dummy resistor, in addition to half-bridge completion.
- Temperature variation decreasing the accuracy of the measurements.
- Sensitivity at 1000 με is ~ 0.5 mV_{out} / V_{EX} input.

Quarter-Bridge Type I Circuit Diagram



The following symbols apply to the circuit diagram:

- R_1 is the half-bridge completion resistor.

- R_2 is the half-bridge completion resistor.
- R_3 is the quarter-bridge completion resistor, known as a dummy resistor.
- R_4 is the active strain gage element measuring tensile strain (+ε).
- V_{EX} is the excitation voltage.
- R_L is the lead resistance.
- V_{CH} is the measured voltage.

The following equation converts voltage ratios to strain units for quarter-bridge configurations.

$$\text{strain}(\varepsilon) = \frac{-4V_r}{GF(1+2V_r)} \cdot \left(1 + \frac{R_L}{R_g}\right)$$

where V_r is the voltage ratio that virtual channels use in the voltage-to-strain conversion equation, GF is the gage factor, R_L is the lead resistance, and R_g is the nominal gage resistance.

NI WSN-3226 Voltage/RTD Node

WSN 4-Channel, 20-Bit Voltage/RTD Node

Node I/O Variables

To use I/O from this node in a VI, drag and drop I/O variables from the **Project Explorer** window to the VI block diagram. The I/O variables for the channels return calibrated floating-point values.

Node Channels

The NI WSN-3226 has the following channels.

Channel	Type	Description
AI**x**	Read	Analog input channel **x**, where **x** is the number of the channel. For the NI WSN-3226, **x** is 0 to 3.
DIO**x**	Read/Write	Digital input/output channel **x**, where **x** is the number of the ch

		annel. For the NI WSN-3226, **x** is 0 to 1.
Battery Voltage	Read	Returns the voltage, in volts, of the batteries installed in the de vice.
Link Quality	Read	Returns the link quality percent age of the radio link.
VI Deployed	Read	Returns **True** if a LabVIEW WSN application has been deployed to the node.
External Power	Read	Returns **True** if external power i s connected to the device.
Mesh Router	Read	Returns **True** if the device is co nfigured as a Mesh Router Node . Returns **False** if the device is c onfigured as an End Node.

NI WSN-3226 Settings

Right-click the node in the **Project Explorer** window and select **Properties**. Select the **Channels** or **Node** tab.

To upload the serial number of a connected node to the **Project Explorer** window, right-click the gateway and select **Refresh node serial numbers**.

Channels Tab: Analog Input

This dialog box includes the following components:

- **Channels**—Specifies the channel or channels to configure.
- **Channel Configuration**
  - **Range**—Specifies the voltage range of the selected channel(s). You cannot change this setting.

Note Set the input range by selecting the **Measurement Type** on the **Channels** tab and the **RTD/Resistance Range** on the **Node** tab.

- **Channel Attributes:**

Attribute	Value
Measurement Type	Voltage, Resistance, or RTD
RTD Temperature Scale*	Celsius, Fahrenheit, or Kelvin
RTD Coefficient* (per degree Celsius)	▪ **3750**—Measures the temperature of an RTD with a=.003750. ▪ **3851**—Measures the temperature of an RTD with a=.003851. ▪ **3911**—Measures the temperature of an RTD with a=.003911. ▪ **3916**—Measures the temperature of an RTD with a=.003916. ▪ **3920**—Measures the temperature of an RTD with a=.003920. ▪ **3928**—Measures the temperature of an RTD with a=.003928.
*Applicable only if you select **RTD** as the **Measurement Type**	

Channels Tab: Digital Input/Output

- **Channels**—Specifies the channel or channels to configure.
- **Channel Configuration**
  - **Range**—You cannot change this setting.
- **Channel Attributes:**

Attribute	Value
DIO Mode	Returns or sets the mode for the corresponding DIO lines. This property can contain the following values: ▪ **DI - 24V Sinking**—Use this mode to connect to industrial 24V sourcing output devices. The input has a low impedance to ground and input thresholds compatible with 24 V signaling. ▪ **DI - 24V Sinking with Power Management**—This mode is similar to DI-24V Sinking, but the low impedance path to ground is removed when the inputs are not being actively read. This may reduce power consumption of the output device. This mode should only be used with output devices that have vali

d output states within 100 µs of having a low impedance path presented to th e output.

- **DI - TTL Logic**—High impedance inputs with TTL compatible thresholds; s uitable for most 3V or 5V logic signals.
- **DI - Contact Closure** —The input has TTL compatible thresholds and prov ides a pull-up resistor to 3V when the input is actively being read. This is suita ble for connections to contact switches wired between the input and ground. I f a contact to ground remains closed in this setting while DIO Notifications are not set to Disabled, the current into the contact switch increases power consu mption and can reduce battery life.
- **DO - Drive Low (Sinking)**—For DIO output value 1, do not drive the DIO li ne. For DIO output value 0, drive the DIO line low. This setting does not requir e a supply voltage on DIO_PWR.
- **DO - Drive High (Sourcing)**—For DIO output value 1, drive the DIO line hi gh to the DIO_PWR voltage. For DIO output value 0, do not drive the DIO line. This setting requires a supply voltage connected to the DIO_PWR pin.
- **DO - 3V TTL Logic (Open-Collector with Pull-Up)**—For DIO output val ue 1, pull-up the DIO line to 3V through a pull-up resistor. For DIO output value 0, drive the DIO line low. When read, the DIO line is read with TTL input thresh olds. This mode allows for connection to 3V and 5V logic inputs with TTL com patible input thresholds. This setting does not require a supply voltage on DIO _PWR.
- **Drive High and Low (Sourcing and Sinking)**—For DIO output value 1, t his mode drives the DIO line high to the DIO_PWR voltage. For DIO output valu e 0, drive the DIO line low. This setting requires a supply voltage connected to the DIO_PWR pin in order to drive high. If no supply is connected to DIO_PWR, this mode will operate as if it were set to the DO-Drive Low (Sinking) mode.

Node Tab (Analog Input and Digital Input/Output)

This dialog box includes the following components:

- **Device Name**—Specifies the name of the WSN node, which appears in the **Project Explorer** window. LabVIEW assigns a default name to the node based on the Hardware Configuration ID. You can use this field to give the node a descriptive name.
- **Device Type**—Specifies the type of WSN node. You cannot change this value.

- **Hardware Configuration**—Specifies the hardware configuration ID and the sample interval (seconds).
  - **ID**—Allows you to map a node in the property tree to a node of that ID on the gateway.
  - **Sample Interval**—Specifies how often the node samples all inputs, in seconds. For End Nodes, the nodes sleep during this interval.
  - **RTD/Resistance Range—**Specifies the resistance value, depending on the sensor type:
    - 400 ohms/Pt100
    - 4 kiloohms/Pt1000
    - 100 kiloohms

Note The range you select applies to all channels only if you select **RTD** or **Resistance** as the **Measurement Type**.

- **Sensor Power**—Specifies when to apply power to the 12 V SENPWR terminal (and not to the RTD sensor). Sensor power turns off after the acquisition (except for Always On). Possible inputs are:
  - **0 ms before sampling**—turns on the Sensor Power immediately when the analog acquisition starts.
  - **25 ms before sampling**—turns on Sensor Power 25 ms before the analog acquisition starts.
  - **100 ms before sampling**—turns on Sensor 100 ms before the analog acquisition starts.
  - **250 ms before sampling**—turns on Sensor Power 250 ms before the analog acquisition starts.
  - **Always On**—turns on Sensor Power when the next analog acquisition starts and leaves it on indefinitely.
  - **Always Off**—never turns on Sensor Power

Note Enabling Sensor Power (other than Always Off) diminishes battery life.

- **Powerline Filtering**—Sets or returns powerline filtering for the node. Possible inputs are:
  - **None**—do not reject powerline noise.
  - **50 Hz**—rejects 50 Hz powerline noise.
  - **60 Hz**—rejects 60 Hz powerline noise.
  - **50/60 Hz**—rejects 50 Hz and 60 Hz powerline noise.
- **Filtering Strength**—Specifies the strength of the filter versus the power consumption. Possible inputs are:
  - **High Rejection**—rejects powerline noise more, at the expense of higher energy consumption.
  - **Low Power**—rejects powerline noise, but as energy efficiently as possible.

NI WSN-3230/3231 Serial Node

WSN 1-Channel, 7 or 8-Bit Serial Input Node

Note The NI WSN-3230/3231 Serial Node is only supported with the LabVIEW Wireless Sensor Network Module.

Node I/O Variables

To use I/O from this node in a VI, drag and drop I/O variables from the **Project Explorer** window to the VI block diagram.

Node Channels

The NI WSN-3230/3231 has the following channels.

Channel	Type	Description
RS232/RS485	Read/Write	Accessed only by using the NI L abVIEW Wireless Sensor Networ k Module. Gives serial read and write capabilities to sensors. Da ta it transferred from the node t

		o the host using User Defined V ariables (UDVs).
DIO**x**	Read/Write	Digital input/output channel **x**, where **x** is the number of the ch annel. For the NI WSN-3230/323 1, **x** is 0 to 1.
Battery Voltage	Read	Returns the voltage, in volts, of the batteries installed in the de vice.
Link Quality	Read	Returns the link quality percent age of the radio link.
VI Deployed	Read	Returns **True** if a LabVIEW WSN application has been deployed to the node.
External Power	Read	Returns **True** if external power i s connected to the device.
Mesh Router	Read	Returns **True** if the device is co nfigured as a Mesh Router Node . Returns **False** if the device is c onfigured as an End Node.

NI WSN-3230/3231 Settings

Right-click the node in the **Project Explorer** window and select **Properties**. Select the **Channels** or **Node** tab.

To upload the serial number of a connected node to the **Project Explorer** window, right-click the gateway and select **Refresh Node Serial Numbers**.

Channels Tab: Digital Input/Output

This dialog box includes the following components:

- **Channels**—Specifies the channel or channels to configure.
- **Channel Configuration**

Range—Specifies the data type of the selected channel(s). You cannot change this setting.

- **Channel Attributes**

Select the **Attribute** and **Value**.

Attribute	Value
DIO Mod e	Returns or sets the mode for the corresponding DIO lines. This property can contain the following values: - **DI - 24V Sinking**—Use this mode to connect to industrial 24V sourcing output devices. The input has a low impedance to ground and input thresholds compati ble with 24 V signaling. - **DI - 24V Sinking with Power Management**—This mode is similar to DI-24V Sinking, but the low impedance path to ground is removed when the inputs are n ot being actively read. This may reduce power consumption of the output device. This mode should only be used with output devices that have valid output states within 100 µs of having a low impedance path presented to the output. - **DI - TTL Logic**—High impedance inputs with TTL compatible thresholds; suita ble for most 3V or 5V logic signals. - **DI - Contact Closure** —The input has TTL compatible thresholds and provide s a pull-up resistor to 3V when the input is actively being read. This is suitable for connections to contact switches wired between the input and ground. If a contac t to ground remains closed in this setting while DIO Notifications are not set to Di sabled, the current into the contact switch increases power consumption and ca n reduce battery life. - **DO - Drive Low (Sinking)**—For DIO output value 1, do not drive the DIO line. For DIO output value 0, drive the DIO line low. This setting does not require a sup ply voltage on DIO_PWR. - **DO - Drive High (Sourcing)**—For DIO output value 1, drive the DIO line high t o the DIO_PWR voltage. For DIO output value 0, do not drive the DIO line. This set ting requires a supply voltage connected to the DIO_PWR pin. - **DO - 3V TTL Logic (Open-Collector with Pull-Up)**—For DIO output value 1 , pull-up the DIO line to 3V through a pull-up resistor. For DIO output value 0, driv e the DIO line low. When read, the DIO line is read with TTL input thresholds. This mode allows for connection to 3V and 5V logic inputs with TTL compatible input t hresholds. This setting does not require a supply voltage on DIO_PWR. - **Drive High and Low (Sourcing and Sinking)**—For DIO output value 1, this mode drives the DIO line high to the DIO_PWR voltage. For DIO output value 0, dri ve the DIO line low. This setting requires a supply voltage connected to the DIO_P

	WR pin in order to drive high. If no supply is connected to DIO_PWR, this mode wi ll operate as if it were set to the DO-Drive Low (Sinking) mode.

Node Tab

This dialog box includes the following components:

- **Device Name**—Specifies the name of the WSN node, which appears in the **Project Explorer** window. LabVIEW assigns a default name to the node based on the Hardware Configuration ID. You can use this field to give the node a descriptive name.
- **Device Type**—Specifies the type of WSN node. You cannot change this value.
- **Hardware Configuration**—Specifies the hardware configuration ID and the sample interval (seconds).
  - **ID**—Allows you to map a node in the property tree to a node of that ID on the gateway.
  - **Sample Interval (Seconds)**—Specifies how often the node samples all inputs, in seconds. For End Nodes, the nodes sleep during this interval.

Configuring Projects

Use the **Project Explorer** window to create an empty project. You can add Vis, WSN devices, and other items to the project.

After configuration, the gateway, node, and channels appear in the **Project Explorer** window.

Configuring a Project with Offline Hardware

Complete the following steps to configure the project if you do not have hardware installed.

1. Create a new project or open an existing project.

2. Right-click the project root in the **Project Explorer** window and select **New»Targets and Devices** from the shortcut menu to display the **Add Targets and Devices** dialog box.
3. Click the **New target or device** radio button, select the gateway, and click **OK**. LabVIEW adds a target item to the project.
4. Right-click the gateway in the **Project Explorer** window and select **New»Targets and Devices** from the shortcut menu to display the **Add Targets and Devices** dialog box.
5. Click the **New target or device** radio button, select the node to add, and click **OK**.
6. Assign an ID to the node, and click **OK**.
7. Right-click the node in the **Project Explorer** window and select **Properties** to configure node settings. Click the **Help** button for information about the node settings.

Configuring a Project with Connected Hardware

Complete the following steps to configure the project. The gateway must be powered on, connected to the same subnet as the host computer, and configured in MAX. Refer to the device documentation for installation and configuration information.

1. Create a new project or open an existing project.
2. Right-click the project root in the **Project Explorer** window and select **New»Targets and Devices** from the shortcut menu to display the **Add Targets and Devices** dialog box.
3. Select either **Discover an existing target(s) or device(s)** or **Specify a target or device by IP address**.
4. Select the gateway and click **OK**. It may take several seconds to update the target configuration.

5. Right-click the node in the **Project Explorer** window and select **Properties** to configure node settings. Click the **Help** button for information about the node settings.

Comparing a Deployed Node Configuration to One on the Host

Right-click the gateway in the **Project Explorer** window, and select **Utilities»Compare Project & System**.

NI-WSN I/O Variables

NI-WSN uses the NI Publish-Subscribe Protocol (NI-PSP) for both remote and local I/O variable access. Local access in this context refers to I/O variable access from a LabVIEW Real-Time VI running on a WSN RT gateway.

Note The Using I/O Variables (ETS, VxWorks, Windows) topic provides general information about I/O variables. Only sections that refer to NI-PSP I/O variables are applicable to NI-WSN.

Note NI-WSN (LabVIEW 2010 or later) does not support local I/O variable access using the NI Scan Engine.

You can access NI-WSN I/O data using static I/O variables from the LabVIEW Project tree or programmatically using DataSocket Vis or Shared Variable Vis with the following URL format:

```
ni.var.psp://gateway/node/channel
```

- gateway—The IP address of the NI WSN-9791 Ethernet Gateway or NI 9792 WSN Real-Time Gateway. For NI 9792 WSN Real-Time Gateway targets, local I/O variable access may use **localhost**.
- node—The WSN node name. The default node name is **Nodex**, where **x** is the ID assigned to the node when it was added to a LabVIEW project or WSN gateway. You can change node names in the LabVIEW project. Node names become active after the you deploy the project to the gateway.
- channel—The I/O variable name. Default I/O variable names depend on the WSN node type. You can change I/O variables names in the LabVIEW project. I/O variable names become active after you deploy the project to the gateway.

You can explore the I/O variable published by your WSN gateway using any of the following:

- LabVIEW Project
- Distributed System Manager
- Browse Variable Object dialog box (accessible from a shared variable refnum control)

Timestamping

By default, NI-WSN I/O variables do not return timestamps. Use the following instructions to enable timestamps:

1. Right-click on the variable in the project tree, and select **Properties**.
2. Check the **Enable Timestamping** box.
3. Find the instance of the variable on the block diagram and right-click it.
4. Select **Timestamp»Show**. An additional output terminal will appear on the variable in the block diagram. The timestamp represents the time at which the data was acquired and is based on the gateway's clock.

If using gateway and node firmware from NI-WSN 1.2 or later, the timestamp is applied at the node when data is taken. Otherwise, it is applied by the gateway when data is received from the node.

For waveform data type nodes the dt and T0 variables are applied at the node.

NI-WSN I/O Variable Buffering Options

You can configure NI-WSN I/O variables to take advantage of NI-PSP buffering capabilities.

Note I/O buffering is not enabled until you configure it as described in the following instructions.

Use the following instructions to access the NI-WSN I/O variable buffering options in the NI-WSN Shared Variable Properties dialog box:

1. Right-click the I/O variable item in the LabVIEW project tree.

2. Select **Properties**.
3. Select the **Variables** property view.
4. Click the **I/O Buffering** tab.

NI-WSN I/O Variable Buffering Options

- **Enable I/O Buffering**—Enables you to store data from the variable in a first-in-first-out (FIFO) buffer. The I/O variable overwrites the first value in the FIFO if the FIFO reaches capacity.
- **Number of elements**—Specifies the number of shared variables the buffer can contain. The value specifies the number of elements of the variable datatype, not the number of bytes.

WSN Gateway Settings

Use this dialog box to configure the gateway.

Right-click the gateway in the **Project Explorer** window, and select **Properties**.

Using the NI Distributed System Manager

Use the NI Distributed System Manager to monitor WSN I/O and to manage WSN targets.

To launch the Distributed System Manager, click **Tools»Distributed System Manager**.

For more information, refer to the Reading Channel Data without Creating a VI topic of this help file.

NI WSN-9791 Ethernet Gateway Settings

Right-click the gateway in the **Project Explorer** window, and select **Properties**.

Use this dialog box to configure the following:

- **Name**—Specifies the name of the device, which appears in the **Project Explorer** window. You can use this field to give the gateway a descriptive name.
- **IP Address/DNS Name**—Specifies the location of the gateway by its IP address or DNS name.

NI 9792 WSN Real-Time Gateway Settings

Right-click the gateway in the **Project Explorer** window, and select **Properties**.

Use this dialog box to configure network settings for the NI 9792 WSN Real-Time Gateway.

General

Conditional Disable Settings

VI Server

Web Server

User Access

Host Environment

Miscellaneous

Scan Engine

General

- **Name**—Specifies the name of the device, which appears in the **Project Explorer** window. You can use this field to give the gateway a descriptive name.
- **IP Address/DNS Name**—Specifies the location of the gateway by its IP address or DNS name.

Conditional Disable Settings

Use these settings to add or remove symbols to use with the Conditional Disable structure. The symbols you add from the **Project Properties** dialog box are available for all targets in the project. The symbols that you add from the

Properties dialog box for a target are available only with the specific target. For example, if you add symbols from the **My Computer** dialog box, the symbols are available only in Vis under **My Computer** that use the Conditional Disable structure.

- **New Symbol**—Specifies the symbol you want to add.
- **New Value**—Specifies the value of the **New Symbol** you want to add.
- **Add**—Adds the **New Symbol** and **New Value** to the table.
- **Remove Selected Items**—Removes the symbols and values you select from the table.

Note You can rename a symbol or change a value in the listbox by clicking on the symbol or value. When the cursor appears, you can type the name or value to which you want to change.

VI Server

This page includes the following components:

- **Protocols**—Use this section to configure the VI Server. The default VI Server settings are **ActiveX** enabled and **TCP/IP** disabled.
  - **TCP/IP**—Enables VI server support for TCP/IP. If you allow remote applications to connect using TCP/IP, you also should specify which machine addresses can access the VI Server in the **Machine Access** section of this page. This checkbox does not contain a checkmark by default.
    - **Port**—Sets the TCP/IP port at which the VI server listens for requests. From **Tools»Options** this port number is 3363, by default, which is a registered port number reserved for use by LabVIEW. For targets, the default is 0 causing the operating system to dynamically select a port. If you want to run multiple application instances on the machine, each with its own VI Server running, you must have a unique VI Server port number. You also can use the Server:Port property to set the LabVIEW VI Server port programmatically.
    - **Service name**—Sets the service name for the VI Server TCP Instance. To retrieve an application reference without the port number, use

Service name in conjunction with the Open Application Reference function by wiring a **Service name** to the polymorphic port number or **Service name** input.

If you display this page from the **Options** dialog box, this service name is `Main Application Instance/VI Server` by default. If you display this page from the **Properties** dialog box for a target, the service name is **target name**`/VI Server` by default. You can use the Server:Service Name property to set the service name programmatically.

- **Use default**—Sets **Service name** to its default value. This checkbox contains a checkmark by default. To edit **Service name**, remove the checkmark from the checkbox.

- **ActiveX**—(Windows) Enables VI server support for ActiveX Automation. This checkbox is available only from the **Tools»Options** navigation. This checkbox contains a checkmark by default.

- **VI Scripting**—Use this section to enable VI Scripting.

  - **Show VI Scripting functions, properties and methods**—Enables VI Scripting functions on the VI Scripting palette and additional VI Server properties and methods. All functions, properties, and methods you enable through VI Scripting display as blue.

    - **Display additional VI Scripting information in Context Help window**—Displays connector pane terminal numbers in the **Context Help** window. Place a checkmark in the **Show VI Scripting functions, properties and methods** checkbox to enable this option.

- **Accessible Server Resources**—Use this section to indicate the tasks that remote applications can accomplish.

  - **VI calls**—Allows remote applications to call Vis exported through the VI Server. If you allow remote applications access to Vis, specify which Vis can be exported. This checkbox contains a checkmark by default.

- **VI properties and methods**—Allows remote applications to read and set the properties of Vis and to call methods for Vis through the VI Server. If you allow remote applications access to Vis, specify which Vis can be exported. This checkbox contains a checkmark by default.
- **Application properties and methods**—Allows remote applications to read and set the properties of the application instance and to call methods for the application instance through the VI Server. This checkbox contains a checkmark by default.
- **Control properties and methods**—Allows remote applications to read and set the properties of controls and to call methods for controls through the VI Server. This checkbox contains a checkmark by default.

- **Machine Access**—Use this section to control machine access to Vis through the VI Server.
  - **Machine access list**—Lists machines that do and do not have access to the VI Server. You also can use the Server:TCP/IP Access List property to list programmatically the TCP/IP addresses of machines that may access the VI server.

    Note If you change the **Machine access list**, machines that are currently connected to the VI server will not be disconnected even if they are no longer allowed access to the server.
  - **Machine name/address**—Enter the name or IP address of the machine you want to add to the **Machine access list**.
  - **Allow access**—Allows access to the machine(s) selected in **Machine access list**.
  - **Deny access**—Denies access to the machine(s) selected in **Machine access list**.
  - **Add**—Adds a new entry to the **Machine access list**. The new entry appears below the selected entry in the **Machine access list**.
  - **Remove**—Removes the selected entry from the **Machine access list**.
- **Exported Vis**—Use this section to add, edit, and remove Vis from the **Exported Vis** list.

- **Exported Vis list**—Lists the Vis that can be exported. You also can use the Server:VI Access List property to list programmatically the Vis on the VI Server that are accessible by remote clients.
- **Exported VI**—Enter a VI to list in **Exported Vis**. You can use wildcards in the VI name or directory path you enter.
- **Allow access**—Allows access to the VI(s) selected in **Exported Vis**. This option is selected by default.
- **Deny access**—Denies access to the VI(s) selected in **Exported Vis**.
- **Add**—Adds a new entry to **Exported Vis**.
- **Remove**—Removes the selected entry from **Exported Vis**.

Web Server

- **Web Application Server** —Use this section to launch user-created Web services in LabVIEW.
  - **Configure Web Application Server**—Launches the Web Monitoring and Configuration interface.
- **Remote Panel Server**—Use this section to access LabVIEW Vis through your Web browser
  - **Enable Remote Panel Server**—Activates the Remote Panel Server. This checkbox does not contain a checkmark by default. You must restart LabVIEW to apply changes to this option. Any changes are saved and appear the next time you open LabVIEW.
  - **Reset to defaults**—Resets all options on the **Web Server: Configuration** page to the default values.
  - **Root directory**—Indicates the directory where the Web Server HTML files are located. The default path is `labview\www`. You also can use the Web Server:Root Directory Path property to specify the root directory programmatically.
  - **HTTP port**—Indicates the TCP/IP port the Web Server uses for unencrypted communication. If another server already uses the port specified by **HTTP port** on the computer or if you are on a computer where

you do not have permission to use reserved ports, such as `80`, replace the value of **HTTP port** with the port you want to use.

You also can use the Web Server:HTTP Port property to set the port to which the built-in Web Server listens for HTTP requests.

Note If you use a port other than `80`, such as `8000`, you must specify the port on URLs that refer to the server, as shown in the following example:

```
http://hostname:8000/index.htm
```

- **Remote front panels**—Allows you to view and control front panels remotely.
- **Snapshot**—Displays a static image of the front panel of a VI currently in memory on the Web Server.
- **SSL**—Enables SSL support on the Web Server.
  - **SSL port**—Indicates the TCP/IP port the Web Server uses for SSL-encrypted communication. You cannot enable SSL on the port specified by **HTTP Port**. You must use a unique port for **SSL port** to allow encrypted communication.
  - **SSL certificate File**—Specifies the certificate to use for SSL encryption on the Web server. You can leave this component blank to use the default LabVIEW self-signed certificate.
  - **Discovered certificates**—Lists available certificates on the system specified in the **Server address** text box.
  - **Server address**—Specifies the name or IP address of a system that contains certificates. For example, you can enter localhost to view certificates on the local system.
  - **Query**—Queries the system specified in the **Server address** text box for available certificates. Discovered certificates appear in the **Discovered certificates** listbox.

- **Log File**—Use this section to enable the log file.

- **Use log file**—Enables the log file. This checkbox does not contain a checkmark by default. You also can use the Web Server:Logging Enabled property to enable the log file programmatically.
  - **Log file path**—Indicates the path of the file where LabVIEW saves Web connection information. The default path is `labview\resource\webserver\logs\access.log`. You also can use the Web Server:Logging File Path property to determine programmatically where the built-in Web Server places the log file.

- **Visible Vis**—Use this section to configure and edit the list of Vis that are visible on the Web.
You also can use the Web Server:VI Access List property to allow and deny access to Vis programmatically.

  - **Visible Vis**—Lists the Vis that are visible through the Web Server. A green checkmark appears to the left of the item when you allow access, and a red `X` appears when you deny access. If an entry does not have a green checkmark or a red `X` by its name, the syntax for the entry is incorrect.
  - **Add**—Adds a new entry to the **Visible Vis** list. The new entry appears below the selected entry in the **Visible Vis** list.
  - **Remove**—Removes the selected entry from the **Visible Vis** list.
  - **Visible VI**—Allows you to enter a VI to list in **Visible Vis**. You can use wildcards in the VI name or directory path you enter. To specify a VI that is part of a LabVIEW project, you must include the project name, the project library name, and the target in the path of the VI, when applicable. For example, if `MyVI.vi` resides in a project called `MyProject.lvproj` under target `My Computer`, enter the VI name as `MyProject.lvproj/My Computer/MyVI.vi`. If the VI is owned by a project library called `MyLibrary`, also include the project library in the path, as in `MyProject.lvproj/My Computer/MyLibrary.lvlib:MyVI.vi`. If the VI is not in a project or project library, you can enter the VI name without any additional information.
  - **Allow access**—Allows access to the VI selected in the **Visible Vis** list. This option is selected by default.

- **Deny access**—Denies access to the VI selected in the **Visible Vis** list.
- **Control time limit (seconds)**—Specifies the amount of time in seconds a remote client can control a VI in the **Visible Vis** list when multiple clients are waiting to control the VI. The default is 300 seconds. If **Use default** contains a checkmark, you cannot edit this field.

Note LabVIEW does not begin monitoring the time limit set on a particular VI until a second client requests control of the same VI. If another client requests control, LabVIEW begins monitoring the control time limit. If a second client never requests control of the VI, the initial client never loses control of the VI.

  - **Use default**—Sets **Control time limit (seconds)** to its default value of 300 seconds. This checkbox contains a checkmark by default. To edit **Control time limit (seconds)**, remove the checkmark from the checkbox.

- **Browser Access**—This section lists the browser addresses that have access to the Web Server. Use this dialog box to add entries to the **Browser Access List**, remove entries from the list, and set access permissions for items in the list. The **Browser Access List** entries must use correct syntax.
You also can use the Web Server:TCP/IP Access List property to allow and deny access to browser addresses programmatically.

  - **Browser access list**—Lists browser addresses that have access to the Web Server. Two green checkmarks appear to the left of the item when you allow viewing and controlling of the front panel, a single green checkmark appears when you allow only viewing of the front panel, and a red X appears when you deny access. If an entry does not have a green checkmark or a red X by its name, the syntax for the entry is incorrect.
  - **Add**—Adds a new browser address to the **Browser Access List**. The new address appears below the selected address in the **Browser Access List**.
  - **Remove**—Removes the selected browser address from the **Browser Access List**.
  - **Browser address**—Specifies a browser address to list in the **Browser access list**. You can use wildcards in the browser address you enter.

- **Allow viewing and controlling**—Allows the browser address selected in the **Browser access list** access to the Web Server for viewing and controlling a VI remotely. This option is selected by default.
- **Allow viewing**—Allows the browser address selected in the **Browser access list** access to the Web Server for viewing Vis and documents.
- **Deny access**—Denies the browser address selected in the **Browser access list** access to the Web Server.

User Access

Use this page to control access to the front panels of Vis running on an RT target.

This page includes the following components:

- **Target Access List**—Lists the IP address of computers that have access to the RT target.
- **IP Address**—Specifies an IP address or symbolic IP address that you want to allow or deny access to the RT target. For example, a `*` entry includes all IP addresses. A `123.123.123.123` entry includes only that specific IP address. A `123.*` entry includes all IP addresses that begin with 123.
- **Allow Access**—Allows access to the RT target from the IP address specified in the **IP Address** text box. The **Target Access List** displays a checkmark before all IP addresses that can access the RT target.
- **Deny Access**—Denies access to the RT target from the **IP address** specified in the IP Address text box. The **Target Access List** displays an X before all IP addresses that cannot access the RT target.
- **Add**—Adds a new entry to the **Target Access List**.
- **Remove**—Removes entries from the **Target Access List**.

Host Environment

Use this page to enable polling the status of the front panel connection with RT targets and to set the polling options.

This page includes the following components:

- **Periodically check responsiveness of RT Protocol**—Enables polling the status of the connection to the RT target.
  - **Ping Delay (ms)**—Specifies the amount of time, in milliseconds, that must elapse without receiving any communication from the RT target before the host computer sends a ping to check the responsiveness of the target.
  - **Ping Timeout (ms)**—Specifies the amount of time, in milliseconds, that the host can wait for the ping to return before displaying a dialog box that warns of an unreliable or dropped connection.

Miscellaneous

Use this page to set miscellaneous options for RT targets.

This page includes the following components:

- **Automatically close VISA sessions**—Automatically closes VISA sessions left open by an RT target application when the top-level VI becomes idle.
- **Enable CPU Load Monitoring**—Specifies whether to log CPU usage data on the target. Enabling this option slightly increases target CPU overhead.
- **Downloaded VI Path Alias**—Specifies the path alias for Vis deployed to the RT target. The path alias is the disk path from which deployed Vis appear to load.

  When you deploy a VI from the **Project Explorer** window, you deploy the VI to memory but you do not save the VI to disk on the RT target. However, when you create a stand-alone application or source distribution, you save the executable files to disk on the RT target.

  Use the **Downloaded VI Path** as a path alias while you debug a VI, before you build the VI into a stand-alone application or source distribution. For example, when you use the Open VI Reference function to open a reference to a deployed VI that you plan to build into a stand-alone application or source distribution, use the **Downloaded VI Path** to ensure that the VI appears to load from the same location where you plan to save the stand-alone application or source distribution to disk on the RT target.

When you use the Open VI Reference function to open a VI deployed to an RT target, ensure that the **VI Path** input of the Open VI Reference function matches the **Downloaded VI Path** plus the name of the VI.

When you change the **Downloaded VI Path**, you must right-click the RT target in the **Project Explorer** window and select **Deploy** from the shortcut menu before the change takes effect.

Scan Engine

Use this page to configure NI Scan Engine settings.

Note The settings you configure on this dialog page apply only to targets with the NI Scan Engine installed.

This page includes the following components:

- **Scan Engine Properties**—Contains general NI Scan Engine configuration settings.
  - **Scan Period**—Specifies the period of the NI Scan Engine. Use the pull-down menu to select the time units of the scan period.
  - **Network Publishing Period (ms)**—Specifies how often, in milliseconds, the target updates published values on the network. This rate should not be faster than the scan period.
  - **Scan Engine Priority**—Specifies the priority of the NI Scan Engine thread on the target. Select **Above time critical** if you do not want any other thread to interrupt the NI Scan Engine thread. Select **Below time critical but above timed structures** if you want Vis set to **time-critical** priority to interrupt the NI Scan Engine thread if necessary. The NI Scan Engine thread always runs at a higher priority than Timed Loops and Timed Sequence structures.
- **Fault Configuration**—Contains NI Scan Engine fault configuration options. Advanced fault configuration options are available only if you select

Custom from the pull-down menu in the **Level** column of the **Configurable Faults** table.

Note To restore the default configuration of a fault, select **Unconfigured** from the pull-down menu in the **Level** column of the **Configurable Faults** table.

- **Configurable Faults**—Contains a table that lists configurable NI Scan Engine faults and the level configured for each of the faults. The faults in this table depend on which scanned I/O drivers are installed on the system.

  Use the pull-down menu in the **Level** column to configure the level of a fault. Select **Custom** from the pull-down menu to enable advanced configuration options for a fault. Select **Unconfigured** to restore the default configuration of a fault.
- **Description**—Displays a description of the fault you select from the **Configurable Faults** list.
- **Occurrence Threshold**—Specifies the number of times the selected fault can occur before the level of the fault changes.
- **Use Time Window**—Specifies whether to use the **Time Window** to bound the amount of time required to meet the occurrence threshold.
- **Time Window**—Specifies a time limit for the occurrence threshold. When the time window expires, LabVIEW resets the internal occurrence counter for the specified fault and restarts the time window. This control is available only if you place a checkmark in the **Use Time Window** checkbox.
- **Level Before Threshold**—Specifies the level of the specified fault before the occurrence threshold is met.
- **Level After Threshold**—Specifies the level of the specified fault after the occurrence threshold is met.

NI 9795 C Series WSN Gateway

Settings for the NI 9795 are determined by the CompactRIO chassis and controller. Right-click the CompactRIO controller in the **Project Explorer** window and select **Properties**.

Reading from WSN Channels

Complete the following steps to read data from a WSN channel.

1. Configure the WSN system.
2. In the **Project Explorer** window, right-click **My Computer** and select **New»VI** from the shortcut menu to add a new VI.
3. Place a loop on the block diagram of the VI.
4. Place the I/O variable for the channel in the loop by dragging and dropping the variable from the LabVIEW project tree. You can also place the I/O variable from the palette by right-clicking and selecting **Structures»Shared Variable**.

Note To change the read/write mode for a DIO variable, right-click input/output on the variable and select **Access Mode**.

5. Right-click the channel output terminal of the I/O variable and select **Create»Indicator**.
6. Right-click the **error out** terminal of the I/O variable and select **Create»Indicator**.
7. Run the VI.

Reading Channel Data without Creating a VI

To read channel data without using a VI, right-click the gateway in the **Project Explorer** window, and select **Utilities»View in System Manager**. The gateway appears under **My Systems**. Expand the item in the configuration tree to display the channel data.

Writing to WSN Channels

Complete the following steps to write to a WSN channel.

1. Configure the WSN system.

2. In the **Project Explorer** window, right-click **My Computer** and select **New»VI** from the shortcut menu to add a new VI.
3. Place the I/O variable for the channel in the loop by dragging and dropping the variable from the LabVIEW project tree. You can also place the I/O variable from the palette by right-clicking and selecting **Structures»Shared Variable**.

Note To change the read/write mode for a DIO variable, right-click input/output on the variable and select **Access Mode**.

4. Right-click the channel input terminal of the I/O variable and select **Create»Control**.
5. Right-click the **error out** terminal of the I/O variable and select **Create»Indicator**.
6. Run the VI.

NI-WSN Error Codes

I/O variables can return the following error codes for NI-WSN.

Code	Description
−306028	Node type not supported by this version of the g ateway firmware or NI-WSN driver for LabVIEW RT.
−306027	Cannot deploy because the WSN node is not incl uded in the gateway configuration.
−306026	Unknown node name.
−306025	Unexpected content type posted to the web ser vice. The correct content type for nodes is text/x ml. The correct content type for firmware-image is application/octet-stream.
−306024	The number of files posted to the web service is incorrect. The correct number is two files.
−306023	The data submitted to the web service cannot b e parsed due to being corrupted or malformed.
−306022	Property is read-only.

−306021	This requested property is invalid.
−306020	Property value is out of range.
−306019	Operation is not allowed while the WSN gatewa y is locked.
−306018	This feature is not supported.
−306017	The user/debug message queue is busy. Unable to send user/debug message.
−306016	Cannot complete the operation. The requested node is unavailable.
−306015	The configuration could not be saved.
−306014	The channel has not been configured and canno t be written to or read.
−306013	No node connected to the gateway at this ID.
−306012	The type of node deployed does not match the node connected at this ID.
−306011	The configuration is corrupt and cannot be pars ed.
−306010	The configuration cannot be generated.
−306009	The version of the communication protocol imp lemented by this firmware is not supported by t he gateway.
−306008	No message information is available for this refe rence.
−306007	Unrecognized message type.
−306006	The user/debug message is too long.
−306005	Firmware update could not be completed becau se one or more nodes are offline.
−306004	Request cannot be completed at this time. A no de firmware update is in progress.
−306003	The node firmware image is not valid. The firmw are is not compatible with one of the nodes in th e update request.
−306002	An invalid configuration parameter was submitt ed to the gateway.

−306001	Local copy of the node configuration is out of da te. Refresh configuration to get the latest config uration.
−306000	NI-WSN - Internal Error.
306000	NI-WSN - Internal warning.