



# FieldPoint LabVIEW™ Interface Help

November 2005, 370480C-01

This help file describes the LabVIEW VIs and functions for FieldPoint hardware.

For more information about this help file, refer to the following topics:

[Using Help](#)

[Related Documentation](#)

[Important Information](#)

[Technical Support and Professional Services](#)

To comment on National Instruments documentation, refer to the [National Instruments Web site](#).

© 2000–2005 National Instruments Corporation. All rights reserved.

## FieldPoint-Related Documentation

The following documents contain information that you may find helpful as you use this help file:

- *LabVIEW Help*—Use this help file to learn more about developing and using VIs and projects.
- *Measurement & Automation Explorer Help for FieldPoint*—Use this help file to learn about configuring and testing FieldPoint I/O in Measurement & Automation Explorer (MAX).
- FieldPoint network module manuals installed at [*FieldPoint folder*]\documentation\Manuals\Network Modules\<
- FieldPoint I/O module operating instructions and other documents on the FieldPoint software CD in the \docs\ folder

# Using Help

[Conventions](#)

[Navigating Help](#)

[Searching Help](#)

[Printing Help File Topics](#)

# Conventions

This help file uses the following formatting and typographical conventions:

- [ ] Square brackets enclose optional items—for example, [response].
- » The » symbol leads you through nested menu items and dialog box options to a final action. The sequence **File»Page Setup»Options** directs you to pull down the **File** menu, select the **Page Setup** item, and select **Options** from the last dialog box.
-  This icon denotes a note, which alerts you to important information.
-  This icon denotes a caution, which advises you of precautions to take to avoid injury, data loss, or a system crash.
- bold** Bold text denotes items that you must select or click in the software, such as menu items and dialog box options. Bold text also denotes parameter names.
- dark red** Text in this color denotes a caution.
- green Underlined text in this color denotes a link to a help topic, help file, or Web address.
- italic* Italic text denotes variables, emphasis, cross references, or an introduction to a key concept. Italic text also denotes text that is a placeholder for a word or value that you must supply.
- monospace Text in this font denotes text or characters that you should enter from the keyboard, sections of code, programming examples, and syntax examples. This font is also used for the proper names of disk drives, paths, directories, programs, subprograms, subroutines, device names, functions, operations, variables, filenames, and extensions.
- monospace bold** Bold text in this font denotes the messages and responses that the computer automatically prints to the screen. This font also emphasizes lines of code that are different from the other examples.
- monospace italic* Italic text in this font denotes text that is a placeholder for a word or value that you must supply.

## Navigating Help (Windows Only)

To navigate this help file, use the **Contents**, **Index**, and **Search** tabs to the left of this window or use the following toolbar buttons located above the tabs:

- **Hide**—Hides the navigation pane from view.
- **Locate**—Locates the currently displayed topic in the **Contents** tab, allowing you to view related topics.
- **Back**—Displays the previously viewed topic.
- **Forward**—Displays the topic you viewed before clicking the **Back** button.
- **Options**—Displays a list of commands and viewing options for the help file.

## Searching Help (Windows Only)

Use the **Search** tab to the left of this window to locate content in this help file. If you want to search for words in a certain order, such as "related documentation," add quotation marks around the search words as shown in the example. Searching for terms on the **Search** tab allows you to quickly locate specific information and information in topics that are not included on the **Contents** tab.

## Wildcards

You also can search using asterisk (\*) or question mark (?) wildcards. Use the asterisk wildcard to return topics that contain a certain string. For example, a search for "prog\*" lists topics that contain the words "program," "programmatically," "progress," and so on.

Use the question mark wildcard as a substitute for a single character in a search term. For example, "?ext" lists topics that contain the words "next," "text," and so on.



**Note** Wildcard searching will not work on Simplified Chinese, Traditional Chinese, Japanese, and Korean systems.

## **Nested Expressions**

Use nested expressions to combine searches to further refine a search. You can use Boolean expressions and wildcards in a nested expression. For example, "example AND (program OR VI)" lists topics that contain "example program" or "example VI." You cannot nest expressions more than five levels.

## Boolean Expressions

Click the  button to add Boolean expressions to a search. The following Boolean operators are available:

- **AND** (default)—Returns topics that contain both search terms. You do not need to specify this operator unless you are using nested expressions.
- **OR**—Returns topics that contain either the first or second term.
- **NOT**—Returns topics that contain the first term without the second term.
- **NEAR**—Returns topics that contain both terms within eight words of each other.

# Search Options

Use the following checkboxes on the **Search** tab to customize a search:

- **Search previous results**—Narrows the results from a search that returned too many topics. You must remove the checkmark from this checkbox to search all topics.
- **Match similar words**—Broadens a search to return topics that contain words similar to the search terms. For example, a search for "program" lists topics that include the words "programs," "programming," and so on.
- **Search titles only**—Searches only in the titles of topics.

## Printing Help File Topics (Windows Only)

Complete the following steps to print an entire book from the **Contents** tab:

1. Right-click the book.
2. Select **Print** from the shortcut menu to display the **Print Topics** dialog box.
3. Select the **Print the selected heading and all subtopics** option.  
 **Note** Select **Print the selected topic** if you want to print the single topic you have selected in the **Contents** tab.
4. Click the **OK** button.

## **Printing PDF Documents**

This help file may contain links to PDF documents. To print PDF documents, click the print button located on the Adobe Acrobat Viewer toolbar.

# Configuring FieldPoint in LabVIEW

You can configure FieldPoint devices in LabVIEW using the [Project Explorer](#) window. If FieldPoint devices are connected to the host computer, you can let LabVIEW search for them. You also can use LabVIEW to configure I/O on banks and devices that are [offline](#), which means that they are not currently connected to the host computer. You may want to do that in order to design and build an application that will run on multiple similar FieldPoint banks.

If you already have a FieldPoint system or systems configured in MAX, you can import your .iak configuration file to a new project in LabVIEW.

If you are installing FieldPoint hardware for the first time, complete the following steps:

1. Install the hardware as described in the network module user manual or quick start guide.
2. Launch LabVIEW.
3. On the Getting Started dialog box, under Files»New, click **Empty Project**.
4. [Add the FieldPoint bank](#) you want to use to the project.
5. Configure the [I/O modules](#) and individual [channels](#) on the bank.
6. [Configure the bank](#) to respond in specific ways to system failures and other events.

# Importing FieldPoint Configuration Files into LabVIEW Projects

If you already have a FieldPoint system or systems configured in MAX, you can import your .iak configuration file to a new project in LabVIEW. Complete the following steps:

1. Right-click the project name in the configuration tree and select **Import»FieldPoint Configuration**.
2. Browse to the location of the .iak file, select it, and click **Open**.

When you import a configuration file, LabVIEW makes a copy of the file for the project and automatically [saves](#) the file when you change the project. The original configuration file remains unchanged.

## **Saving FieldPoint Configuration Files in LabVIEW Project**

When you save a LabVIEW project, LabVIEW saves the .iak configuration file in the same folder as the project file, with the same file name. Only the file extension differs, so that if your project file is called MyProject.lvproj, the corresponding .iak file is called MyProject.iak. If you want to move or copy the project to another computer, you must also move or copy the .iak file.

If you open a project that includes FieldPoint items and the .iak file is unavailable, all FieldPoint items will be invalid and will be represented by yellow icons in the configuration tree. If you can replace or recover the .iak file, close the project without saving, find the .iak file, put it in the same folder as your project file, and reopen the project. The valid FieldPoint items will be restored in the configuration tree.

## **Editing FieldPoint Configuration Files in MAX**

You must use MAX to perform some FieldPoint configuration tasks such as network configuration and access control. If you change or rename devices and items in MAX, the devices and items will be invalid when you reopen the project in LabVIEW.

If you open a project that includes FieldPoint items and the .iak file is unavailable, all FieldPoint items will be invalid and will be represented by yellow icons in the configuration tree. If you can replace or recover the .iak file, close the project without saving, find the .iak file, put it in the same folder as your project file, and reopen the project. The valid FieldPoint items will be restored in the configuration tree.

If you have renamed devices, items, or the .iak file, close the project without saving, open the .iak file in MAX, rename the devices, items, or .iak file, and resave the .iak file. Then reopen the project in LabVIEW.

# Adding a FieldPoint Bank to a LabVIEW Project

Use the Add Targets and Devices dialog box to add FieldPoint banks to your LabVIEW project.

[Adding a FieldPoint Ethernet bank on the network.](#)

[Adding a FieldPoint Ethernet bank on a different subnet.](#)

[Adding a FieldPoint serial bank.](#)

[Adding an offline FieldPoint bank to a project for configuration.](#)

## Searching for FieldPoint Devices on the Network

If a FieldPoint Ethernet network module is on the network and has an IP address, you can use the Add Targets and Devices dialog box to search for it and add it to your project. Complete the following steps:

1. Right-click the project name in the project tree and select **New»Targets and Devices**.
2. In the Add Targets and Devices dialog box that appears, **Existing target or device** should be selected by default. If it is not selected, select it now.
3. Use the folder tree in the dialog box to find the FieldPoint bank with the I/O you want to add to your project. If the I/O is on a bank with a FieldPoint RT controller such as the cFP-21xx, look in the Real-Time FieldPoint folder. If the I/O is on a bank with a non-RT Ethernet network module such as the FP-1601 or cFP-180x, look in the FieldPoint Network Modules folder.
4. When you find the bank with the I/O you want to add to your project, click **OK**.

## Adding FieldPoint Devices on Another Subnet

If a FieldPoint Ethernet network module is on a different subnet from the host PC and you know the IP address, you can use the Add Targets and Devices dialog box to add the bank to your project. Complete the following steps:

1. Right-click the project name in the project tree and select **New»Targets and Devices**.
2. In the Add Targets and Devices dialog box that appears, select **Existing device on remote subnet**.
3. Expand **FieldPoint Controller** and select the device from the list.
4. Enter the IP address and click **OK**.
5. An icon for the bank appears in the Project Explorer project tree. The default name of the bank is "FP Target" for RT controllers and "FP Res" for non-RT network modules. You can give the bank a descriptive name.
6. Right-click the bank icon and select **Properties** to display the General properties page. Verify the IP address and the name of the bank.

## Adding FieldPoint Serial Devices

You can use the Add Targets and Devices dialog box to add a FieldPoint serial bank to your project. Complete the following steps:

1. Install the serial network module and connect it to the host PC according to the documentation shipped with the module.
2. Right-click **My Computer** in the project tree and select **New»Targets and Devices**.
3. In the Add Targets and Devices dialog box that appears, select **New target or device**, select **FieldPoint Serial**, then click **OK**.
4. Configure the new serial comm resource in the [Add FieldPoint Serial](#) dialog box that appears.
5. Right-click the comm resource icon and select the type of serial network module for the bank you want to add. One serial comm resource can connect to several banks.
6. Right-click a bank icon and select **New»Targets and Devices**.
7. In the Add Targets and Devices dialog box that appears, select **New target or device**, then select an I/O module type, and click **OK**.
8. Repeat the previous step until the bank contains all the I/O modules that your application requires.
9. Configure [I/O modules](#) and [I/O items](#) on the new serial bank.
10. [Deploy](#) the configuration changes you have made.

## Device Overview

FieldPoint devices have channels for input and/or output. The channels are represented by [I/O items](#) under the device in the project tree. Some I/O items represent things other than input and output channels, such as a power supply or cold-junction compensation channel.

You can do any of the following tasks by right-clicking the module icon and selecting the task from the dropdown menu:

- **Add New Item**—When you add a device to the project tree, LabVIEW populates the device with the default I/O items for the device type. You can create custom data items by clicking **Add New Item**, then using the Add Logical Item dialog box to configure the item.
- **Deploy**—Downloads all the changes you have made on the [Channel](#) properties page to the selected device. Refer to the [Deploying Configuration Changes](#) page for more information about deploying.

For more information, refer to the [Device Properties](#) and [Channel Properties](#) pages.

## I/O Item Overview

When you add a device to the project tree, LabVIEW populates the device with the default I/O items for the device type. You can create custom data items by clicking **Add New Item**, then using the Add Logical Item dialog box to configure the item. Some I/O items represent things other than input and output channels, such as a power supply or cold-junction compensation channel.

You can use the [Item Properties](#) page to configure individual I/O items.

## Remote I/O

Remote I/O enables you to use a FieldPoint RT controller to access I/O modules connected to other network modules. You can use the Project Explorer to add items from I/O modules on any connected FieldPoint bank to your LabVIEW project.

Complete the following steps to access I/O on a FieldPoint serial bank:

1. [Add the FieldPoint banks](#) with the I/O items you want to your project.
2. Add I/O items from the new bank to a VI in your project under **My Computer**. Test the VI.
3. Configure the FieldPoint RT controller you want to use to access the I/O on the remote bank.
4. Disconnect the serial cable from the host computer COM port and connect it to a COM port on the FieldPoint RT controller.
5. Drag and drop the icon for the VI onto the icon for the FieldPoint RT controller in the project configuration tree.

Complete the following steps to access I/O on a FieldPoint Ethernet bank:

1. [Add the FieldPoint banks](#) with the I/O items you want to your project.
2. Add I/O items from the new bank to a VI in your project under **My Computer**. Test the VI.
3. Configure the FieldPoint RT controller you want to use to access the I/O on the remote bank.
4. Drag and drop the icon for the VI onto the icon for the FieldPoint RT controller in the project configuration tree.



**Note** Network I/O tasks run at normal priority. If a VI running embedded on an RT controller contains I/O items that must be accessed over Ethernet, set the VI to run at normal priority. Alternatively, you can use the [Real-Time Timing VIs](#) to ensure that normal-priority tasks in the VI can run.



**Note** When MAX or LabVIEW downloads remote I/O item names to a controller, it saves a file called fpreote.ini on the controller at `[IP address]/ni-rt/system`. If you are not using remote I/O, having this file on a controller adds unnecessary delay to startup time. To avoid downloading remote item names when you deploy configuration changes from a host PC, you can create a DWORD registry value called Disable Remote ItemName Download in `HKEY_CURRENT_USER\Software\National Instruments\FieldPoint Explorer\Settings` and assign a

nonzero data value to the new registry value. This change will take effect when you exit and relaunch MAX or LabVIEW. If MAX or LabVIEW has already downloaded remote item names to the controller, you can FTP to *[IP address]*/ni-rt/system and delete fremote.ini.

## Using I/O Items from Non-FieldPoint APIs

If you want to use a FieldPoint RT controller to access I/O items that are configured with non-FieldPoint software, you can add them to your project as [shared variables](#).

## Offline Configuration

Offline configuration enables you to configure channels, create items, and build VIs and applications without having FieldPoint hardware connected to the host computer.

To configure I/O for a FieldPoint Ethernet bank that is not connected to the host PC, complete the following steps:

1. Right-click the project name in the project tree and select **New»Targets and Devices**.
2. In the Add Targets and Devices dialog box that appears, select **New target or device**.
3. Expand **FieldPoint Controller** and select the network module you want to configure from the list.
4. Click **OK**.
5. An icon for the bank appears in the Project Explorer project tree. The default name of the bank is "FP Target" for RT controllers and "FP Res" for non-RT network modules. You can give the bank a descriptive name.
6. Right-click the bank icon and select **New»Targets and Devices**.
7. In the Add Targets and Devices dialog box that appears, select **New target or device**, then select an I/O module type, and click **OK**.
8. Repeat the previous step until the bank contains all the I/O modules that your application requires.
9. Configure [I/O modules](#) and [I/O items](#) on the new bank.
10. Configure the I/O channels and items.

To configure I/O for a FieldPoint serial bank that is not connected to the host PC, complete the following steps:

1. Right-click **My Computer** in the project tree and select **New»Targets and Devices**
2. In the Add Targets and Devices dialog box that appears, select **New target or device**, select **FieldPoint Serial**, then click **OK**.
3. Configure the new serial comm resource in the [Add FieldPoint Serial](#) dialog box that appears, then click **OK**.

4. An icon for the serial comm resource appears in the Project Explorer project tree. You can give the comm resource a descriptive name.
5. Right-click the comm resource icon and select the type of serial network module for the bank you want to add. One serial comm resource can connect to several banks.
6. Right-click a bank icon and select **New»Targets and Devices**.
7. In the Add Targets and Devices dialog box that appears, select **New target or device**, then select an I/O module type, and click **OK**.
8. Repeat the previous step until the bank contains all the I/O modules that your application requires.
9. Configure [I/O modules](#) and [I/O items](#) on the new serial bank.

## **Deploying Configuration Changes**

After you make configuration changes on properties pages, you can deploy the changes to download them to FieldPoint devices.

## Deploying Changes to the RT Controller and the Bank

Right-click the RT controller icon and select **Deploy** to download the following changes:

If LabVIEW is not connected to a FieldPoint bank and you right-click the bank and select **connect**, all changes you have made on properties pages for the bank are automatically deployed.

- [Watchdog configuration](#) for the bank
- Configuration changes on properties pages for the RT controller, *except* changes on the [General](#) properties page
- Local and [remote](#) item names
- [Custom scaling](#) configuration for items

Right-click the RT controller icon and select **Deploy All** to download all of the changes in the above list as well as the following changes:

- Changes on the [Bank and Network Module](#) properties pages
- Changes on the [Device](#) and [Channels](#) properties pages
- Local and [remote](#) item names
- [Custom scaling](#) configuration for items
- All VIs, build specifications, and libraries associated with the RT controller

## Deploying Changes to the Network Module

Right-click the network module icon and select **Deploy** to download the following changes:

- Changes on the [Bank and Network Module](#) properties pages
- [Watchdog](#) configuration for the bank and all I/O modules on the bank
- Local and [remote](#) item names



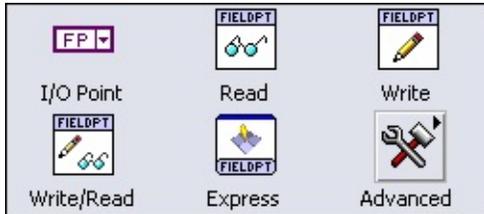
**Note** Deploying changes to the network module does not download VIs, build specifications, or changes on the RT-specific properties pages.

## Deploying Changes to I/O Modules

Right-click an I/O module icon and select **Deploy** to download changes on the [channel](#) properties page.

# FieldPoint VIs

Use the FieldPoint VIs located on the **Functions»Measurement I/O»FieldPoint** palette to interact with FieldPoint devices.



Palette Object	Description
<a href="#">I/O Point</a>	References a FieldPoint item configured in LabVIEW or Measurement & Automation Explorer (MAX). A FieldPoint item represents a channel or group of channels on a FieldPoint module.
<a href="#">Read (Polymorphic)</a>	Reads a single set of values from a FieldPoint item.
<a href="#">Write (Polymorphic)</a>	Asynchronously writes to a FieldPoint item when used with an FieldPoint Ethernet module. Synchronously writes to a FieldPoint item when used with a FieldPoint serial module or when running embedded on a FieldPoint RT controller.
<a href="#">Write/Read (Polymorphic)</a>	Asynchronously writes to and reads from FieldPoint items when used with a FieldPoint Ethernet module. Synchronously writes to and reads from FieldPoint items when used with a FieldPoint serial module and when running embedded on a FieldPoint RT controller. When running embedded on a FieldPoint RT controller, this VI provides better performance than separate FP Read and FP Write VIs.
<a href="#">Express</a>	Reads from or writes to channels on FieldPoint modules. The physical channels on the hardware are referred to as items in software.

Subpalette	Description
<a href="#">Advanced</a>	Use the Advanced VIs located on the <b>FieldPoint»Advanced</b> palette to interact with legacy VIs and for low-level Ethernet and serial communication.

## **Before Getting Started with the FieldPoint VIs**

You must configure FieldPoint network modules in Measurement & Automation Explorer (MAX) before using any of the FieldPoint virtual instruments (VIs) in LabVIEW. MAX stores all configuration information for the FieldPoint network and I/O modules in a configuration file (.iak) on your computer. For information about configuring FieldPoint network modules in MAX, refer to the *Measurement & Automation Explorer Help for FieldPoint* (**Start»Programs»National Instruments»FieldPoint»FieldPoint Help**).

## FieldPoint I/O Point

FieldPoint I/O points refer to data items configured in LabVIEW and Measurement & Automation Explorer (MAX) and saved in FieldPoint configuration (.iak) files. A FieldPoint item represents a channel or group of channels on a FieldPoint module.



The FieldPoint IO Point function is supported only in LabVIEW 7.0 and later. If you want to run a VI that uses the FieldPoint IO Point function on a computer that does not have LabVIEW 7.0 or later, you must [modify the VI](#).

The .iak file that LabVIEW uses for FieldPoint I/O points in a VI depends on whether the VI is part of a project and, if so, which project the VI is part of.

## Using FieldPoint I/O in Project VIs

When you use a FieldPoint I/O point in a VI that is part of a project, the default .iak file is the one associated with the [project](#). You can add items to the VI from another .iak file by adding that file to the configuration list of the VI. To add an .iak file, click **Browse** on the FieldPoint I/O point control and add the .iak file on the **View Configurations** tab.

## I/O Control outside a Project

When you use a FieldPoint I/O point in a VI that is not part of a project, the default .iak file is the last one saved in MAX. You can add items to the VI from another .iak file by adding that file to the configuration list of the VI. Click **Browse** on the FieldPoint I/O point control and add the .iak file on the **View Configurations** tab.



**Note** LabVIEW identifies the last-saved .iak file when you create a new I/O point control or block-diagram constant, browse on an I/O point control, or run a VI that contains FieldPoint I/O. If you change the .iak file in MAX to a different file, you must exit and relaunch LabVIEW for the change to take effect.



**Note** If you save a VI outside a project and later open it inside a project, LabVIEW looks for FieldPoint I/O items in the project .iak file. If you save a VI inside a project and later open it outside the project, LabVIEW looks for FieldPoint I/O items in the .iak file last saved in MAX. If you load a different .iak file in MAX and you want a nonproject LabVIEW VI to use the new file, you must exit and relaunch LabVIEW. If you use MAX to make changes to the .iak file that a nonproject VI is using, and you want the changes to take effect in the VI, go to the **Browse** dialog and select **Refresh**.

## FieldPoint I/O Points in Embedded Applications

If you are using FieldPoint I/O in a real-time application, LabVIEW uses the project .iak file. When you run the VI embedded on an RT target, LabVIEW automatically downloads local and [remote](#) FieldPoint item names to the RT target.

## **Building LabVIEW Executable Applications Using FieldPoint I/O Points**

If you create an executable LabVIEW application to run on a host machine, you must use MAX to save the .iak file associated with the project. Otherwise, the application will return an error message and prompt the user to select an .iak file.

# FieldPoint Express

Reads from or writes to channels on FieldPoint modules. The physical channels on the hardware are referred to as items in software. You can create items in Measurement & Automation Explorer (MAX) that represent single or multiple channels on a FieldPoint module.

[Dialog Box Options](#)

[Block Diagram Inputs](#)

[Block Diagram Outputs](#)

## Options to Select in Configuration Dialog Box

Parameter	Description
<b>FieldPoint IO Point</b>	Refers to the FieldPoint item you want to read from or write to.
<b>IO Operation</b>	Determines whether the VI reads data from or writes data to the FieldPoint IO point.
<b>Name of VI</b>	Names the VI according to which of the following you select: <ul style="list-style-type: none"><li>- <b>Name of FieldPoint IO Point</b> assigns the name that appears in the <b>FieldPoint IO Point</b> field.</li><li>- <b>Name of IO operation</b> assigns the name of the selected <b>IO Operation</b>.</li><li>- <b>Custom name</b> assigns the name entered in the <b>custom name</b> field.</li></ul>
	Determines the values the VI writes. This parameter is enabled only if <b>write data</b> is selected for the <b>IO Operation</b> parameter. <ul style="list-style-type: none"><li>• <b>Input terminal</b> creates a <b>Value In</b> terminal. The VI writes the</li></ul>

<b>Write source</b>	<p>values wired to this terminal.</p> <ul style="list-style-type: none"> <li>- <b>Write every value input</b> writes to the FieldPoint IO point every time a value is input to <b>Value In</b>, regardless of whether <b>Value In</b> changes.</li> <li>- <b>Write updates only</b> writes to the FieldPoint IO point only when <b>Value In</b> changes.</li> <li>• <b>Constant value</b> writes the value entered in the <b>Value to Write</b> field.</li> </ul>
<b>Output timestamp</b>	Determines whether the VI associates a timestamp with the value read from or written to the FieldPoint IO point.

## Inputs to Wire on Block Diagram

Parameter	Description
<b>error in</b>	Describes error conditions that occurred before this VI or function runs.
<b>Value In</b>	The value to write to the FieldPoint IO point.

## Outputs to Wire on Block Diagram

Parameter	Description
<b>error out</b>	Contains error information. If <b>error in</b> indicates that an error occurred before this VI or function ran, <b>error out</b> contains the same error information. Otherwise, it describes the error status that this VI or function produces.
<b>Timestamp</b>	Supplies the time that the FieldPoint IO point performed the read operation.
<b>Value Out</b>	An array of values read from the FieldPoint IO point. Each value can be wired to a waveform chart and treated as a waveform. In this case, time 0 is the timestamp value from the first run of this

	instance of the VI.
<b>Data Changed?</b>	Returns TRUE if the value has changed since the last time this instance of the VI ran.

This Express VI uses the functionality found in the following VIs and functions:

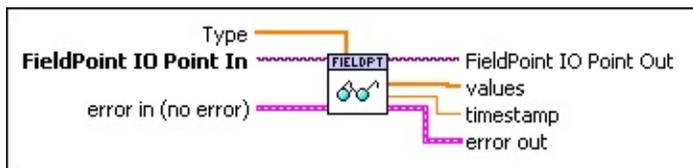
[FieldPoint IO Point](#)

[FP Read \(Polymorphic\) VI](#)

[FP Write \(Polymorphic\) VI](#)

# FP Read (Polymorphic)

Reads a single set of values from a FieldPoint item. FP Read operates synchronously when communicating over a serial network and when running embedded on a FieldPoint RT controller. It operates asynchronously when communicating with a FieldPoint system over Ethernet.



- POLY** **Type** supplies the data type of **values**. Valid [data types](#) are Float (DBL), Float Array, Boolean, and Boolean Array.
- POLY** **FieldPoint IO Point In** is a reference to the FieldPoint item(s) created in MAX. This polymorphic terminal accepts [FieldPoint I/O point](#) input or [FP Create Tag](#) input. National Instruments recommends that you use FieldPoint I/O point input. If you are using LabVIEW 6.x, the terminal is not polymorphic and only accepts input from FP Create Tag. The FieldPoint IO Point function is not supported in versions of LabVIEW before 7.0. If you want to run a VI that uses the FieldPoint IO Point function on a computer that does not have LabVIEW 7.0, you must [modify the VI](#).
- ESH** **error in (no error)** describes error conditions that occur before this VI or function runs. The default is no error. If an error occurred before this VI or function runs, the VI or function passes the **error in (no error)** value to **error out**. This VI or function runs normally only if no error occurred before this VI or function runs. If an error occurs while this VI or function runs, it runs normally and sets its own error status in **error out**. Use the Simple Error Handler or General Error Handler VIs to display the description of the error code. 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.
  - TF** **status** is TRUE (X) if an error occurred before this VI or function ran or FALSE (checkmark) to indicate a warning or that no error occurred before this VI or function ran. The default is FALSE.

 **code** is the error or warning code. The default is 0. If **status** is TRUE, **code** is a non-zero error code. If **status** is FALSE, **code** is 0 or a warning code.

 **source** describes the origin of the error or warning and is, in most cases, the name of the VI or function that produced the error or warning. The default is an empty string.

 **FieldPoint IO Point Out** is the throughput terminal that returns the value input from **FieldPoint IO Point In**.

 **values** is a data value or array of data values read from the FieldPoint IO point item. Valid [data types](#) are Float (DBL), Float Array, Boolean, and Boolean Array.

 **timestamp** returns the last time at which the VI received a value from the item. If this VI is running embedded on an RT controller or reading values over a serial network, **timestamp** returns the last time the VI read the item. If this VI is running on a host computer and reading values over Ethernet, **timestamp** returns the last time the value of the item changed.

 **error out** contains error information. If **error in** indicates that an error occurred before this VI or function ran, **error out** contains the same error information. Otherwise, it describes the error status that this VI or function produces. Right-click the **error out** indicator on the front panel and select **Explain Error** from the shortcut menu for more information about the error.

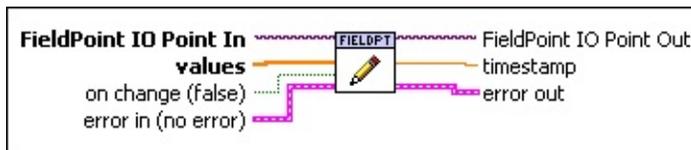
 **status** is TRUE (X) if an error occurred or FALSE (checkmark) to indicate a warning or that no error occurred.

 **code** is the error or warning code. If **status** is TRUE, **code** is a non-zero error code. If **status** is FALSE, **code** is 0 or a warning code.

 **source** describes the origin of the error or warning and is, in most cases, the name of the VI or function that produced the error or warning.

# FP Write (Polymorphic)

Asynchronously writes to a FieldPoint item when used with an FieldPoint Ethernet module. Synchronously writes to a FieldPoint item when used with a FieldPoint serial module or when running embedded on a FieldPoint RT controller.



**POLY** **FieldPoint IO Point In** is a reference to the FieldPoint item(s) created in MAX. This polymorphic terminal accepts [FieldPoint IO point](#) input or [FP Create Tag](#) input. National Instruments recommends that you use FieldPoint IO point input. If you are using LabVIEW 6.x, the terminal is not polymorphic and only accepts input from FP Create Tag. The FieldPoint IO Point function is not supported in versions of LabVIEW before 7.0. If you want to run a VI that uses the FieldPoint IO Point function on a computer that does not have LabVIEW 7.0, you must [modify the VI](#).

**POLY** **values** is a data value or array of data values written to the FieldPoint IO point item. Valid [data types](#) are Float (DBL), Float Array, Boolean, and Boolean Array. The number of values wired to the **values** terminal must match the number of values readable from the **FieldPoint IO Point In** terminal.

**TFI** **on change (false)** determines whether the VI writes only when the value to write has changed from the previously written value (TRUE). If FALSE, the VI writes every value. The first time FP Write (Polymorphic) is executed, the value is always written to the FieldPoint I/O Point item.

 Leave **on change** FALSE unless the value wired to **FieldPoint IO Point In** will remain constant. Multiplexed data items are not compatible with **on change**.

**ERR** **error in (no error)** describes error conditions that occur before this VI or function runs. The default is no error. If an error occurred before this VI or function runs, the VI or function passes the **error in (no error)** value to **error out**. This VI or function runs normally only if no error occurred

before this VI or function runs. If an error occurs while this VI or function runs, it runs normally and sets its own error status in **error out**. Use the Simple Error Handler or General Error Handler VIs to display the description of the error code. 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.

 **status** is TRUE (X) if an error occurred before this VI or function ran or FALSE (checkmark) to indicate a warning or that no error occurred before this VI or function ran. The default is FALSE.

 **code** is the error or warning code. The default is 0. If **status** is TRUE, **code** is a non-zero error code. If **status** is FALSE, **code** is 0 or a warning code.

 **source** describes the origin of the error or warning and is, in most cases, the name of the VI or function that produced the error or warning. The default is an empty string.

 **FieldPoint IO Point Out** is the throughput terminal that returns the value input from **FieldPoint IO Point In**.

 **timestamp** returns the time that the VI performed the write operation.

 **error out** contains error information. If **error in** indicates that an error occurred before this VI or function ran, **error out** contains the same error information. Otherwise, it describes the error status that this VI or function produces. Right-click the **error out** indicator on the front panel and select **Explain Error** from the shortcut menu for more information about the error.

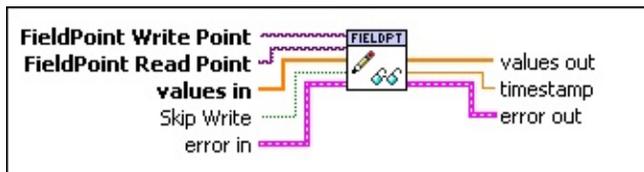
 **status** is TRUE (X) if an error occurred or FALSE (checkmark) to indicate a warning or that no error occurred.

 **code** is the error or warning code. If **status** is TRUE, **code** is a non-zero error code. If **status** is FALSE, **code** is 0 or a warning code.

 **source** describes the origin of the error or warning and is, in most cases, the name of the VI or function that produced the error or warning.

# FP Write/Read (Polymorphic)

Asynchronously writes to and reads from FieldPoint items when used with a FieldPoint Ethernet module. Synchronously writes to and reads from FieldPoint items when used with a FieldPoint serial module and when running embedded on a FieldPoint RT controller. When running embedded on a FieldPoint RT controller, this VI provides better performance than separate FP Read and FP Write VIs.



**FP** **FieldPoint Write Point** and **FieldPoint Read Point** are references to FieldPoint item(s) created in MAX. These terminals accept [FieldPoint IO Point](#) input.

**POLY** **values in** is a data value or an array of data values to write to the FieldPoint Write Point item. Valid [data types](#) are Float (DBL), Float Array, Boolean, and Boolean Array. The number of values wired to the **values in** terminal must match the number of values readable from the **FieldPoint Write Point** terminal.

**TF** **Skip Write (FALSE)** determines whether the VI writes before reading. If FALSE, the VI writes before reading. If TRUE, the VI reads but does not write.

**ERR** **error in (no error)** describes error conditions that occur before this VI or function runs. The default is no error. If an error occurred before this VI or function runs, the VI or function passes the **error in (no error)** value to **error out**. This VI or function runs normally only if no error occurred before this VI or function runs. If an error occurs while this VI or function runs, it runs normally and sets its own error status in **error out**. Use the Simple Error Handler or General Error Handler VIs to display the description of the error code. 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.

**TF** **status** is TRUE (X) if an error occurred before this VI or function ran or FALSE (checkmark) to indicate a warning or that no error occurred before this VI or function ran. The default is FALSE.

**I32** **code** is the error or warning code. The default is 0. If **status** is TRUE, **code** is a non-zero error code. If **status** is FALSE, **code** is 0 or a warning code.

**abc** **source** describes the origin of the error or warning and is, in most cases, the name of the VI or function that produced the error or warning. The default is an empty string.

**FOLY** **values out** returns the value of the FieldPoint Read Point item. Refer to [FieldPoint Data Types](#) for information about wiring this terminal.

**DBL** **timestamp** returns the last time at which the VI received a value from the FieldPoint Read Point item. If this VI is running embedded on an RT controller or reading values over a serial network, **timestamp** returns the last time the VI read the item. If this VI is running on a host computer and reading values over Ethernet, **timestamp** returns the last time the value of the item changed.

**FFH** **error out** contains error information. If **error in** indicates that an error occurred before this VI or function ran, **error out** contains the same error information. Otherwise, it describes the error status that this VI or function produces. Right-click the **error out** indicator on the front panel and select **Explain Error** from the shortcut menu for more information about the error.

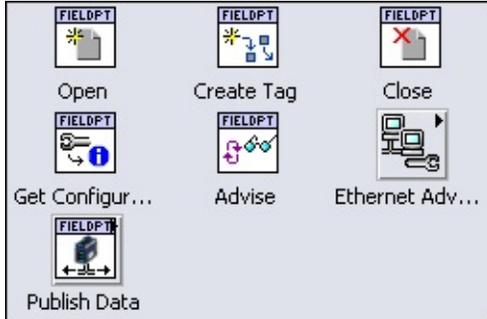
**TF** **status** is TRUE (X) if an error occurred or FALSE (checkmark) to indicate a warning or that no error occurred.

**I32** **code** is the error or warning code. If **status** is TRUE, **code** is a non-zero error code. If **status** is FALSE, **code** is 0 or a warning code.

**abc** **source** describes the origin of the error or warning and is, in most cases, the name of the VI or function that produced the error or warning.

# Advanced VIs

Use the Advanced VIs to interact with legacy VIs and for low-level Ethernet and serial communication.



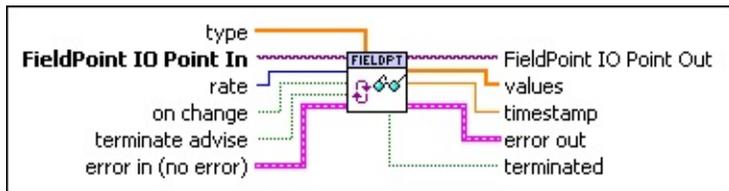
Palette Object	Description
<a href="#">Open</a>	Opens a communication session with the FieldPoint server and a specified .iak configuration file.
<a href="#">Create Tag</a>	Creates a tag reference for the specified data item.
<a href="#">Close</a>	Closes all communication resources in the associated .iak file and frees serial ports for other uses.
<a href="#">Get Configuration Info</a>	Reads the FieldPoint configuration information from the .iak configuration file. It can be used to get a list of configured comm resources, devices, or items.
<a href="#">Advise (Polymorphic)</a>	Asynchronously reads values from a FieldPoint item at a specified rate.

Subpalette	Description
<a href="#">Ethernet Advanced</a>	Use the VIs on this subpalette to communicate with and configure FieldPoint Ethernet banks over the network from within LabVIEW. If you configured the FieldPoint Ethernet module in Measurement & Automation Explorer (MAX), you do not need to use the Ethernet configuration VIs.
<a href="#">Publish Data</a>	Use the VIs on this subpalette to share LabVIEW data between two FieldPoint RT controllers or a FieldPoint RT controller and a local computer.

# FP Advise (Polymorphic)

Asynchronously reads values from a FieldPoint item at a specified rate. Follow the FP Advise (Polymorphic) programming tips. [Details](#)

**!** **Caution** When in doubt about whether to use FP Advise (Polymorphic) or [FP Read \(Polymorphic\)](#), use [FP Read \(Polymorphic\)](#).



**POLY** **type** supplies the data type of **values**. Valid [data types](#) are Float (DBL), Float Array, Boolean, and Boolean Array.

**POLY** **FieldPoint IO Point In** is a reference to the FieldPoint item(s) created in MAX. This polymorphic terminal accepts [FieldPoint I/O point](#) input or [FP Create Tag](#) input. National Instruments recommends that you use FieldPoint I/O point input. If you are using LabVIEW 6.x, the terminal is not polymorphic and accepts input only from FP Create Tag. The FieldPoint IO Point function is not supported in versions of LabVIEW before 7.0. If you want to run a VI that uses the FieldPoint IO Point function on a computer that does not have LabVIEW 7.0, you must [modify the VI](#).

**U32** **rate** is the rate in milliseconds at which the VI polls the data item.

**on change** specifies whether the VI outputs data only when the data item has changed in value from its previous read value. If **on change** is TRUE, the VI returns data only when the data changes. If **on change** is FALSE, the VI returns data on every poll of the data item. This is useful for monitoring data that changes infrequently. If **on change** is TRUE and the FP Advise (Polymorphic) VI is waiting for data to change, the FP Advise (Polymorphic) VI continues to wait for the data to change before returning data even if you change the **on change** input to FALSE.

**TFI**

**!** **Caution** Do not place more than one FP Advise (Polymorphic) with **on change** set to true in the same loop.

**TFI** **terminate advise** specifies whether to terminate the advise operation.

This terminal should be set to FALSE until the final iteration of the loop containing the VI, then set to TRUE to shut down the background process. If FP Advise (Polymorphic) is in a loop and a value of TRUE is passed to **terminate advise**, FP Advise (Polymorphic) terminates the background process. However, if the loop continues to run, FP Advise (Polymorphic) runs again, restarts the background process, and reads values on the next iteration of the loop.

**Err** **error in (no error)** describes error conditions that occur before this VI or function runs. The default is no error. If an error occurred before this VI or function runs, the VI or function passes the **error in (no error)** value to **error out**. This VI or function runs normally only if no error occurred before this VI or function runs. If an error occurs while this VI or function runs, it runs normally and sets its own error status in **error out**. Use the Simple Error Handler or General Error Handler VIs to display the description of the error code. 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.

**TF** **status** is TRUE (X) if an error occurred before this VI or function ran or FALSE (checkmark) to indicate a warning or that no error occurred before this VI or function ran. The default is FALSE.

**I32** **code** is the error or warning code. The default is 0. If **status** is TRUE, **code** is a non-zero error code. If **status** is FALSE, **code** is 0 or a warning code.

**abc** **source** describes the origin of the error or warning and is, in most cases, the name of the VI or function that produced the error or warning. The default is an empty string.

**FOLY** **FieldPoint IO Point Out** is the throughput terminal that returns the value input from **FieldPoint IO Point In**.

**FOLY** **values** is an array of data values read from the FieldPoint IO point item. Valid [data types](#) are Float (DBL), Float Array, Boolean, and Boolean Array.

**DBL** **timestamp** returns the last time at which the value of the item changed.

**Err** **error out** contains error information. If **error in** indicates that an error occurred before this VI or function ran, **error out** contains the same error information. Otherwise, it describes the error status that this VI or function produces. Right-click the **error out** indicator on the front panel and select **Explain Error** from the shortcut menu for more information

about the error.

 **status** is TRUE (X) if an error occurred or FALSE (checkmark) to indicate a warning or that no error occurred.

 **code** is the error or warning code. If **status** is TRUE, **code** is a non-zero error code. If **status** is FALSE, **code** is 0 or a warning code.

 **source** describes the origin of the error or warning and is, in most cases, the name of the VI or function that produced the error or warning.

 **terminated** is set to TRUE when the advise operation has been terminated and FALSE while the server is executing the advise operation.

## FP Advise (Polymorphic) Details

Because FP Advise (Polymorphic) continually reads the IO point at a specified rate, the While Loop that contains FP Advise (Polymorphic) does not need a timer. The **on change** input causes FP Advise (Polymorphic) to return data only when there is a change in the data coming from the module (polling continues in the background at the specified rate). This allows you to make portions of the block diagram inactive.

 **Caution** Do not use the FP Advise (Polymorphic) VI in nonreentrant subVIs.

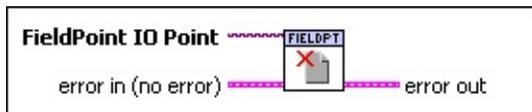
 **Caution** FP Advise (Polymorphic) does not work properly if you place it in a For Loop that auto-indexes an array of FieldPoint IO Point items. Use FP Read (Polymorphic) instead.

# FP Close (Polymorphic)

Closes all communication resources in the associated .iak file and frees serial ports for other uses. Follow the [program flow](#) guidelines when you use this VI.

 **Note** If you are using LabVIEW 7.0 or later and the [FieldPoint I/O Point function](#) (recommended), you do not need to use this VI because the FieldPoint I/O Point function closes communication resources when you exit LabVIEW. However, you can use this VI to close the resources while LabVIEW is still running.

## FP Close (IO)



**POLY** **FieldPoint IO Point In** is a reference to the FieldPoint item(s) created in MAX. This polymorphic terminal accepts [FieldPoint IO Point](#) input or refnum input. National Instruments recommends that you use FieldPoint IO Point input. This VI does not accept FieldPoint IO Point input in versions of LabVIEW before 7.1.

**ERR** **error in (no error)** describes error conditions that occur before this VI or function runs. The default is no error. If an error occurred before this VI or function runs, the VI or function passes the **error in (no error)** value to **error out**. If an error occurs while this VI or function runs, it runs normally and sets its own error status in **error out**. Use the Simple Error Handler or General Error Handler VIs to display the description of the error code. 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.

**TFI** **status** is TRUE (X) if an error occurred before this VI or function ran or FALSE (checkmark) to indicate a warning or that no error occurred before this VI or function ran. The default is FALSE.

 **code** is the error or warning code. The default is 0. If **status** is TRUE, **code** is a non-zero error code. If **status** is FALSE, **code** is 0 or a warning code.

 **source** describes the origin of the error or warning and is, in most cases, the name of the VI or function that produced the error or warning. The default is an empty string.

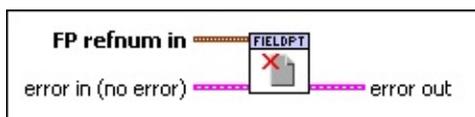
 **error out** contains error information. If **error in** indicates that an error occurred before this VI or function ran, **error out** contains the same error information. Otherwise, it describes the error status that this VI or function produces. Right-click the **error out** indicator on the front panel and select **Explain Error** from the shortcut menu for more information about the error.

 **status** is TRUE (X) if an error occurred or FALSE (checkmark) to indicate a warning or that no error occurred.

 **code** is the error or warning code. If **status** is TRUE, **code** is a non-zero error code. If **status** is FALSE, **code** is 0 or a warning code.

 **source** describes the origin of the error or warning and is, in most cases, the name of the VI or function that produced the error or warning.

## FP Close



 **FP refnum in** is the input handle of an FP operation.

 **server refnum** is a unique refnum generated by FP Open.

 **tag refnum** is a unique refnum generated by [FP Create Tag](#).

 **Note** Because tag refnums are associated with specific server refnums, multiple tag refnums will probably exist for each server refnum.

 **error in (no error)** describes error conditions that occur before this VI or function runs. The default is no error. If an error occurred before this VI

or function runs, the VI or function passes the **error in (no error)** value to **error out**. If an error occurs while this VI or function runs, it runs normally and sets its own error status in **error out**. Use the Simple Error Handler or General Error Handler VIs to display the description of the error code. 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.

 **status** is TRUE (X) if an error occurred before this VI or function ran or FALSE (checkmark) to indicate a warning or that no error occurred before this VI or function ran. The default is FALSE.

 **code** is the error or warning code. The default is 0. If **status** is TRUE, **code** is a non-zero error code. If **status** is FALSE, **code** is 0 or a warning code.

 **source** describes the origin of the error or warning and is, in most cases, the name of the VI or function that produced the error or warning. The default is an empty string.

 **error out** contains error information. If **error in** indicates that an error occurred before this VI or function ran, **error out** contains the same error information. Otherwise, it describes the error status that this VI or function produces. Right-click the **error out** indicator on the front panel and select **Explain Error** from the shortcut menu for more information about the error.

 **status** is TRUE (X) if an error occurred or FALSE (checkmark) to indicate a warning or that no error occurred.

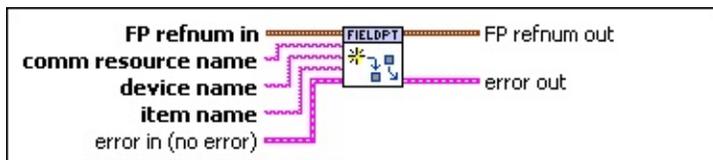
 **code** is the error or warning code. If **status** is TRUE, **code** is a non-zero error code. If **status** is FALSE, **code** is 0 or a warning code.

 **source** describes the origin of the error or warning and is, in most cases, the name of the VI or function that produced the error or warning.

# FP Create Tag

Creates a tag reference for the specified data item on the device and comm resource that is accessed by the communication session FP refnum. When you use this VI, the VI references configuration information either in the .iak file or embedded on a FieldPoint RT controller. Follow the [program flow](#) guidelines when you use this VI. After establishing the session, use [FP Advise \(Polymorphic\)](#), [FP Read \(Polymorphic\)](#), or [FP Write \(Polymorphic\)](#) to communicate with the device. In LabVIEW 6.1 you can create customized versions of the FP Create Tag VI that are tailored for use with an .iak file that you specify. [Details](#)

 **Note** If you are using LabVIEW 7.0 or later and the [FieldPoint I/O Point function](#) (recommended), you do not need to use this VI. The FieldPoint I/O Point function references the FieldPoint item with which you are communicating.



 **FP refnum in** is the input handle of an FP operation.

 **server refnum** is a unique refnum generated by [FP Open](#).

 **tag refnum** is a unique refnum generated by [FP Create Tag](#).

 **Note** Because tag refnums are associated with specific server refnums, multiple tag refnums will probably exist for each server refnum.

 **comm resource name** is the name of the comm resource as displayed in MAX.  **device name** is the name of the module as displayed in MAX.  **item name** is the name of the configured data item as displayed in MAX.  **error in (no error)** describes error conditions that occur before this VI or function runs. The default is

no error. If an error occurred before this VI or function runs, the VI or function passes the **error in (no error)** value to **error out**. This VI or function runs normally only if no error occurred before this VI or function runs. If an error occurs while this VI or function runs, it runs normally and sets its own error status in **error out**. Use the Simple Error Handler or General Error Handler VIs to display the description of the error code. 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.

 **status** is TRUE (X) if an error occurred before this VI or function ran or FALSE (checkmark) to indicate a warning or that no error occurred before this VI or function ran. The default is FALSE.

 **code** is the error or warning code. The default is 0. If **status** is TRUE, **code** is a non-zero error code. If **status** is FALSE, **code** is 0 or a warning code.

 **source** describes the origin of the error or warning and is, in most cases, the name of the VI or function that produced the error or warning. The default is an empty string.

 **FP refnum out** is the output handle of an FP operation.

 **server refnum** is a unique refnum generated by [FP Open](#).

 **tag refnum** is a unique refnum generated by [FP Create Tag](#).

 **error out** contains error information. If **error in** indicates that an error occurred before this VI or function ran, **error out** contains the same error information. Otherwise, it describes the error status that this VI or function produces. Right-click the **error out** indicator on

the front panel and select **Explain Error** from the shortcut menu for more information about the error.

 **status** is TRUE (X) if an error occurred or FALSE (checkmark) to indicate a warning or that no error occurred.

 **code** is the error or warning code. If **status** is TRUE, **code** is a non-zero error code. If **status** is FALSE, **code** is 0 or a warning code.

 **source** describes the origin of the error or warning and is, in most cases, the name of the VI or function that produced the error or warning.

## FP Create Tag Details

The values for **comm resource name**, **device name**, and **item name** match (case and space sensitive) the item names in the .iak configuration file. One FP Create Tag should be used for each item (whether a single channel or multiple channels on the same module) that you want to monitor or control in the program.

You can create Custom Create Tag VIs in LabVIEW 6.1. VIs that use Custom Create Tags are supported in LabVIEW 7.0. The Custom Create Tag VIs use ring controls instead of string controls for the comm resource, device name, and item name terminals. The ring controls on the Custom Create Tag VIs are populated by names from the specified .iak file so that the names can be entered without having to manually enter values for each field.

 **Caution** If you are embedding Custom Create Tag VIs on a FieldPoint RT controller, make sure all of the Custom Create Tag VIs are either identical or have a different name. Using two or more Custom Create Tag VIs that are different but have the same name can cause the FieldPoint controller to function incorrectly.

Complete the following steps to create a Custom Create Tag VI.

1. Launch MAX. Create and save your .iak file.
2. Launch LabVIEW. Select **Import FieldPoint Tags**

- from the **Tools** menu.
3. Click **Create Custom VI** and enter the **IAK File Path** and a **Configuration Name**. Click **OK**.
  4. Relaunch LabVIEW to display the Custom Create Tag VI on the FieldPoint VIs palette.

The Custom Create Tag VI is created and named FP Custom Create Tag (X).vi, where X is the configuration name you specify. The Custom Create Tag VI you created is located in your \\LabVIEW\vi.lib\FieldPoint\Custom Create Tag directory. The VI will not work if it is moved to a different location. The VI is available on the **Functions»FieldPoint»Create Tag»Custom Create Tag** palette.

Complete the following steps to use a Custom Create Tag VI.

1. Place the Custom Create Tag VI on the block diagram.
2. Right-click the **comm resource name, device name, or item name** terminal and select **Create Constant**.
3. Use the hand tool on the newly created constant to select a value from the list for the terminal.

All other uses for the Custom Create Tag VIs are the same as those for the standard FP Create Tag VI.



**Note** The comm resource name, device name, and item name terminals in the Custom Create Tag VI contain all of the comm resource names, device names, and item names in the specified .iak file. Not all device names or item names are valid for a given comm resource and not all item names are valid for a given device. If you select an invalid item name or device name, there will be an error when you execute the VI.

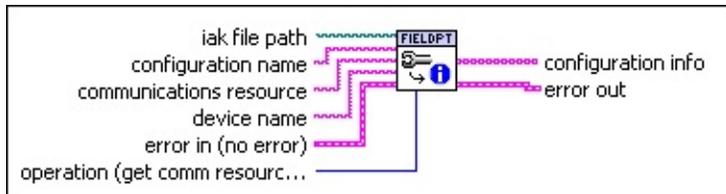


**Note** FP Create Tag does not automatically update from the .iak configuration file. If you modify the .iak file, you must update the FP

Custom Create Tag VI or create a new one. You must also update all the FP Custom Create Tag ring controls.

# FP Get Configuration Info

Reads the FieldPoint configuration information from the .iak configuration file. It can be used to get a list of configured comm resources, devices, or items.



 **iak file path** provides the .iak configuration file to use with the VI. If you leave this parameter empty, the VI uses the last .iak file used in MAX. To avoid errors and warnings that result from MAX selecting a different .iak file than the file you want the VI to reference, it is best to always wire in the path to the .iak file.

 **configuration name** is the name that the FP IO control associates with a .iak file on the host computer. You can wire the configuration name to this terminal to specify a .iak file path. Use the Browse dialog on the FP IO control to get the configuration name. If the **iak file path** terminal is wired, the VI ignores this terminal.

 **communications resource** is the name of the comm resource as displayed in MAX.

 **device name** is the name of the module under the comm resource as displayed in MAX.

 **error in (no error)** describes error conditions that occur before this VI or function runs. The default is no error. If an error occurred before this VI or function runs, the VI or function passes the **error in (no error)** value to **error out**. This VI or function runs normally only if no error occurred before this VI or function runs. If an error occurs while this VI or function runs, it runs normally and sets its own error status in **error out**. Use the Simple Error Handler or General Error Handler VIs to display the description of the error code. 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.

 **status** is TRUE (X) if an error occurred before this VI or function

ran or FALSE (checkmark) to indicate a warning or that no error occurred before this VI or function ran. The default is FALSE.

 **code** is the error or warning code. The default is 0. If **status** is TRUE, **code** is a non-zero error code. If **status** is FALSE, **code** is 0 or a warning code.

 **source** describes the origin of the error or warning and is, in most cases, the name of the VI or function that produced the error or warning. The default is an empty string.

 **operation (get comm resource list)** specifies the query operation to perform on the .iak configuration file:

- get configuration name list—Returns an array of strings that contains the FieldPoint configuration names registered on the local computer for [FieldPoint IO Point](#).
- get comm resource list—Returns an array of strings representing configured comm resources in the specified .iak file (or representing the comm resources associated with the controller, if the VI is running embedded).
- get device list—Returns an array of strings representing the devices configured on one comm resource.
- get item list—Returns an array of strings of representing configured data items on a particular device.
- get item list (long string)—Same as get item list except that it returns the item list in the slash-delimited form that the FP IO control uses: Name/Comm Resource/Device/Item.

 **configuration info** returns the result of the query operation. The **configuration info** output does not return items in any particular order.

 **error out** contains error information. If **error in** indicates that an error occurred before this VI or function ran, **error out** contains the same error information. Otherwise, it describes the error status that this VI or function produces. Right-click the **error out** indicator on the front panel and select **Explain Error** from the shortcut menu for more information about the error.

 **status** is TRUE (X) if an error occurred or FALSE (checkmark) to indicate a warning or that no error occurred.

 **code** is the error or warning code. If **status** is TRUE, **code** is a non-zero error code. If **status** is FALSE, **code** is 0 or a warning code.

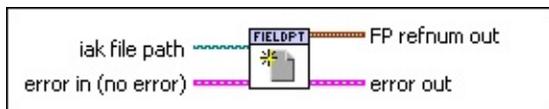


**source** describes the origin of the error or warning and is, in most cases, the name of the VI or function that produced the error or warning.

# FP Open

Opens a communication session with the FieldPoint server and a specified .iak configuration file. Use only one FP Open for each configuration file. Do not place FP Open in a While Loop. Follow the [program flow](#) guidelines when you use this VI.

 **Note** If you are using LabVIEW 7.0 or later and the [FieldPoint I/O Point function](#) (recommended), you do not need to use this VI.



 **iak file path** provides the .iak server configuration file to use with the specified server. If you leave this parameter empty, the server uses the last file used in MAX. To avoid errors and warnings that result from MAX selecting a different .iak file than the file you want the VI to reference, it is best to always wire in the path to the .iak file.

 **error in (no error)** describes error conditions that occur before this VI or function runs. The default is no error. If an error occurred before this VI or function runs, the VI or function passes the **error in (no error)** value to **error out**. This VI or function runs normally only if no error occurred before this VI or function runs. If an error occurs while this VI or function runs, it runs normally and sets its own error status in **error out**. Use the Simple Error Handler or General Error Handler VIs to display the description of the error code. 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.

 **status** is TRUE (X) if an error occurred before this VI or function ran or FALSE (checkmark) to indicate a warning or that no error occurred before this VI or function ran. The default is FALSE.

 **code** is the error or warning code. The default is 0. If **status** is TRUE, **code** is a non-zero error code. If **status** is FALSE, **code** is 0 or a warning code.

 **source** describes the origin of the error or warning and is, in most cases, the name of the VI or function that produced the error or

warning. The default is an empty string.



**FP refnum out** is the output handle of an FP operation.



**server refnum** is a unique refnum generated by [FP Open](#).



**tag refnum** is a unique refnum generated by [FP Create Tag](#).



**error out** contains error information. If **error in** indicates that an error occurred before this VI or function ran, **error out** contains the same error information. Otherwise, it describes the error status that this VI or function produces. Right-click the **error out** indicator on the front panel and select **Explain Error** from the shortcut menu for more information about the error.



**status** is TRUE (X) if an error occurred or FALSE (checkmark) to indicate a warning or that no error occurred.



**code** is the error or warning code. If **status** is TRUE, **code** is a non-zero error code. If **status** is FALSE, **code** is 0 or a warning code.



**source** describes the origin of the error or warning and is, in most cases, the name of the VI or function that produced the error or warning.

# Program Flow

If you are using [FP Open](#), [FP Create Tag](#), or [FP Close](#), make sure that your VI follows the program flow guidelines.

[Input Item](#)

[Output Item](#)

[Multiple Item](#)

# Ethernet Advanced VIs

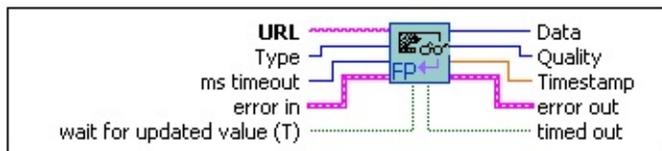
Use the Ethernet Advanced VIs to communicate with and configure FieldPoint Ethernet banks over the network from within LabVIEW. If you configured the FieldPoint Ethernet module in Measurement & Automation Explorer (MAX), you do not need to use the Ethernet configuration VIs.



Palette Object	Description
<a href="#">Network Read</a>	Reads data from FieldPoint I/O modules connected to a FieldPoint Ethernet network module. Also reads data from LabVIEW publish data items created with the <a href="#">Publish Data VIs</a> and published from a [c]FP-2xxx controller.
<a href="#">Configure Range</a>	Enables programmatic configuration of FieldPoint channel ranges. If you configured the ranges in LabVIEW or MAX, you do not need to use this VI.
<a href="#">Configure Attribute</a>	Enables programmatic configuration of FieldPoint channel attributes. If you configured the channel attributes in LabVIEW or MAX, you do not need to use this VI.
<a href="#">Configure Deadband</a>	Enables programmatic configuration of FieldPoint deadband values. If you configured the deadband values in LabVIEW or MAX, you do not need to use this VI.

# FieldPoint Network Read

Reads data from FieldPoint I/O modules connected to a FieldPoint Ethernet network module. Also reads data from LabVIEW publish data items created with the [Publish Data VIs](#) and published from a [c]FP-2xxx controller. To write values to I/O modules or Publish Data items, use the DataSocket Write VI or the [Publish Data VIs](#), respectively. [Details](#)



**URL** identifies the data source to read. When you are working online, use the DataSocket Select URL VI to obtain the correct URL for the data item. Enter the IP address of the FieldPoint network module in the **Browse host:** field. Browse to the desired data item. Copy the URL of the item and paste it into a string control or a constant in your LabVIEW VI. Wire the control or the constant to the FieldPoint Network Read URL terminal.

When entering URLs, make sure the URL contains a valid IP address, in the ###.###.###.### format, followed by a slash, /. Leading zeros must be excluded as shown in the following example: 150.166.42.111/.

For I/O channel items, the URLs must contain the following elements in the order shown:

- The IP address of the module.
- FP/
- The numeric address of the module on the FieldPoint bank concatenated with the abbreviated name of the module. The address of an I/O module is its position in the FieldPoint bank relative to the network module. The network module is at address 0; the first I/O module to the right of the network module is at address 1; the second I/O module to the right of the network module is at address 2, and so on. The abbreviated name of a FieldPoint module consists of the letters to the right of the FP- on the identification label. For example, URLs communicating with a FieldPoint bank consisting of a [c]FP-

2000, [c]FP-DO-400, and [c]FP-DI-330 contain /02000/, /1DO/, or /2DI/.



- The two-digit numeric values of the I/O channel being read.

For LabVIEW items, the URLs must contain the following elements in the order shown:

- The IP address of the network module.
- LV/
- The block name under which the item is published. (RTPublished is the default.)
- The name of the published data item.

The following are examples of acceptable URLs:

- 150.166.42.111/FP/2DO/01
- fieldpoint://150.166.42.111/FP/2DO/01
- 120.114.103.1/FP/3TC/01
- 120.114.103.1/FP/02000/01
- 120.114.103.1/LV/RTPublished/Data1
- 120.114.103.1/LV/TemperatureAverages/Cooler1



**Note** The network protocol part of the URL does not need to be included. If it is included, the VI ignores it. The FieldPoint Network Read VI always uses the National Instruments Industrial Automation Network protocol ("fieldpoint://" or "lookout://"). To read data items from another network protocol, use the DataSocket Read function.

**Type** specifies the type of data to be read and defines the type of the data output terminal. The terminal accepts I32, Double, Boolean, and String data types. An error results if the data type wired in does not match the data type of the item to which it is subscribed.



**ms timeout** specifies the amount of time to wait for a value update. This time is ignored if **wait for updated value** is false and an initial value has arrived. The default is 10 seconds.



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

to **error out**. This VI or function runs normally only if no error occurred before this VI or function runs. If an error occurs while this VI or function runs, it runs normally and sets its own error status in **error out**. Use the Simple Error Handler or General Error Handler VIs to display the description of the error code. 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.

 **status** is TRUE (X) if an error occurred before this VI or function ran or FALSE (checkmark) to indicate a warning or that no error occurred before this VI or function ran. The default is FALSE.

 **code** is the error or warning code. The default is 0. If **status** is TRUE, **code** is a non-zero error code. If **status** is FALSE, **code** is 0 or a warning code.

 **source** describes the origin of the error or warning and is, in most cases, the name of the VI or function that produced the error or warning. The default is an empty string.

**wait for updated value** instructs the VI to wait for an updated value when TRUE. If the value has been updated since the last read, the VI returns the new value immediately. Otherwise, the VI waits the number of milliseconds defined in **ms timeout** for an update. If an update does not occur in the timeout period, the current value is returned and the **timed out** value is TRUE. If **wait for updated value** is FALSE, the VI returns the current value of the data.



**Data** is the result of the read. If the function times out, the value returned is the same as the value that was last read. A zero, an empty, or an equivalent value is returned by the VI if the function times out and nothing is read or if the type of data is incompatible. The type (I32, Double, Boolean, or String) of the data returned is specified by the type wired into the **type** terminal and the data type of the published item.



**Quality** returns the value of the quality attribute of the **data** item. A zero value indicates good quality.



**Timestamp** is the value of the Timestamp attribute of the **data** item the last time that it was successfully read.



**error out** contains error information. If **error in** indicates that an error occurred before this VI or function ran, **error out** contains the same error information. Otherwise, it describes the error status that this VI or function produces. Right-click the **error out** indicator on the front panel

and select **Explain Error** from the shortcut menu for more information about the error.

 **status** is TRUE (X) if an error occurred or FALSE (checkmark) to indicate a warning or that no error occurred.

 **code** is the error or warning code. If **status** is TRUE, **code** is a non-zero error code. If **status** is FALSE, **code** is 0 or a warning code.

 **source** describes the origin of the error or warning and is, in most cases, the name of the VI or function that produced the error or warning.

 **timed out** returns TRUE if the function times out while it is waiting for an update or an initial value.

## FieldPoint Network Read Details

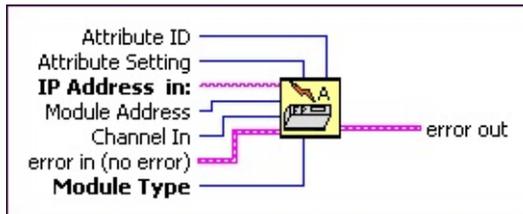
In addition to reading data from FieldPoint modules connected to FieldPoint Ethernet modules and from LabVIEW publish data items, the FieldPoint Network Read VI reads the quality and timestamp attributes of data items. The FieldPoint Network Read VI is available with LabVIEW versions 6.1 and higher. If an I/O module is connected to a FieldPoint Ethernet controller that is visible over the network, the channel data for the I/O module is also visible over the network.

# Configure Attribute

Enables programmatic configuration of FieldPoint channel attributes. If you configured the channel attributes in Measurement & Automation Explorer (MAX), you do not need to use this VI.



**Caution** Be attentive when using the FP Configure Attribute VI. If you wire an incorrect **module type** and/or **module address**, you can change the characteristics of the FieldPoint system, resulting in damage to the process or device the FieldPoint system is controlling. Hot swapping of modules during programmatic configuration is not supported.



**Attribute ID** is the ID number of the channel attribute you want to configure. The attribute ID corresponds to the index of the attribute on the ring control. Note that Input Filter and Noise Rejection are equivalent at index 1 and that there is no attribute with an ID of zero. If the terminal is unwired, the default value is 1. If the terminal is wired to an invalid value, an error is generated. This terminal does not need to be wired if a polymorphic subVI specific to the desired attribute is chosen. To select an attribute, right-click the **attribute ID** terminal and select the attribute from the **Select Type** menu. Different enumerated values are provided for the **Attribute Setting** terminal depending on which attribute subVI is chosen.



**Attribute Setting** is the ID number or value of the desired channel attribute (ID number for enumerated attributes, value for integer attributes). If the terminal is unwired, the default value is one. If the terminal is wired to an invalid value, an error is generated. To select an attribute, right-click the **attribute setting** terminal and select the attribute from the **Select Type** menu. Different enumerated values are provided for the **Attribute Setting** terminal depending on which attribute subVI is chosen. After selecting the attribute, right-click the **Attribute Setting**

terminal and select **Create»Constant**.

 **IP Address in:** is the address of the FieldPoint network module, in XXX.XXX.XXX.XXX format (leading zeros excluded). Provide only the IP address, not a complete URL. If the terminal is unwired or invalid, an error occurs.

 **Module Address** is the address of the module containing the channel for attribute configuration. The first module after the network module is address 1; the second is address 2, and so on. If the terminal is unwired, the default value is zero. If the terminal is wired to an invalid value, an error occurs.

 **Channel In** is the channel for attribute configuration. If the terminal is unwired, the default value is zero. If the terminal is wired to an invalid value, an error occurs.

 **error in (no error)** describes error conditions that occur before this VI or function runs. The default is no error. If an error occurred before this VI or function runs, the VI or function passes the **error in (no error)** value to **error out**. This VI or function runs normally only if no error occurred before this VI or function runs. If an error occurs while this VI or function runs, it runs normally and sets its own error status in **error out**. Use the Simple Error Handler or General Error Handler VIs to display the description of the error code. 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.

 **status** is TRUE (X) if an error occurred before this VI or function ran or FALSE (checkmark) to indicate a warning or that no error occurred before this VI or function ran. The default is FALSE.

 **code** is the error or warning code. The default is 0. If **status** is TRUE, **code** is a non-zero error code. If **status** is FALSE, **code** is 0 or a warning code.

 **source** describes the origin of the error or warning and is, in most cases, the name of the VI or function that produced the error or warning. The default is an empty string.

 **Module Type** is the type name of the module from an enumerated control.

 **error out** contains error information. If **error in** indicates that an error occurred before this VI or function ran, **error out** contains the same error information. Otherwise, it describes the error status that this VI or

function produces. Right-click the **error out** indicator on the front panel and select **Explain Error** from the shortcut menu for more information about the error.

 **status** is TRUE (X) if an error occurred or FALSE (checkmark) to indicate a warning or that no error occurred.

 **code** is the error or warning code. If **status** is TRUE, **code** is a non-zero error code. If **status** is FALSE, **code** is 0 or a warning code.

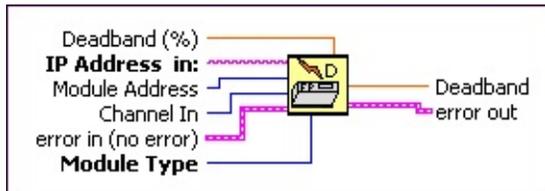
 **source** describes the origin of the error or warning and is, in most cases, the name of the VI or function that produced the error or warning.

# Configure Deadband

Enables programmatic configuration of FieldPoint deadband values. If you configured the deadband values in Measurement & Automation Explorer (MAX), you do not need to use this VI.



**Caution** Be attentive when using the FP Configure Deadband VI. If you wire an incorrect **module type** and/or **module address**, you can change the characteristics of your FieldPoint system resulting in damage to the process or device your FieldPoint system is controlling. Hot swapping of modules during programmatic configuration is not supported.



**DBL** **Deadband (%)** is the desired deadband as a percent of the input channel's range. If the terminal is unwired, the default value is zero. If the terminal is wired to an invalid value, an error occurs.

**abc** **IP Address in:** is the address of the FieldPoint network module, in XXX.XXX.XXX.XXX format (leading zeros excluded). Provide only the IP address, not a complete URL. If the terminal is unwired or invalid, an error occurs.

**US** **Module Address** is the address of the module for deadband configuration. The first module after the network module is address 1; the second is address 2, and so on. If the terminal is unwired or invalid, an error occurs.

**US** **Channel In** is the channel for deadband configuration. If the terminal is unwired, the default value is zero. If the terminal is wired to an invalid value, an error occurs.

**Err** **error in (no error)** describes error conditions that occur before this VI or function runs. The default is no error. If an error occurred before this VI or function runs, the VI or function passes the **error in (no error)** value to **error out**. This VI or function runs normally only if no error occurred before this VI or function runs. If an error occurs while this VI or function

runs, it runs normally and sets its own error status in **error out**. Use the Simple Error Handler or General Error Handler VIs to display the description of the error code. 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.

 **status** is TRUE (X) if an error occurred before this VI or function ran or FALSE (checkmark) to indicate a warning or that no error occurred before this VI or function ran. The default is FALSE.

 **code** is the error or warning code. The default is 0. If **status** is TRUE, **code** is a non-zero error code. If **status** is FALSE, **code** is 0 or a warning code.

 **source** describes the origin of the error or warning and is, in most cases, the name of the VI or function that produced the error or warning. The default is an empty string.

 **Module Type** is the type name of the module from an enumerated control.

 **Deadband** outputs the deadband percentage wired in at **Deadband (%)**.

 **error out** contains error information. If **error in** indicates that an error occurred before this VI or function ran, **error out** contains the same error information. Otherwise, it describes the error status that this VI or function produces. Right-click the **error out** indicator on the front panel and select **Explain Error** from the shortcut menu for more information about the error.

 **status** is TRUE (X) if an error occurred or FALSE (checkmark) to indicate a warning or that no error occurred.

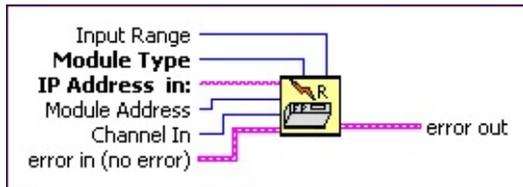
 **code** is the error or warning code. If **status** is TRUE, **code** is a non-zero error code. If **status** is FALSE, **code** is 0 or a warning code.

 **source** describes the origin of the error or warning and is, in most cases, the name of the VI or function that produced the error or warning.

# Configure Range

Enables programmatic configuration of FieldPoint channel ranges. If you configured the ranges in Measurement & Automation Explorer (MAX), you do not need to use this VI.

**Caution** Be attentive when using the FP Configure Range VI. If you wire an incorrect **module type** and/or **module address**, you can change the characteristics of your FieldPoint system resulting in damage to the process or device your FieldPoint system is controlling. Hot swapping of modules during programmatic configuration is not supported.



**U8** **Input Range** is the ID number of the desired channel range. Equivalent to the index of the channel range in the ring control on the front panel. The recommended method is right-clicking the terminal and selecting **Create»Constant**. Select the range you want from the drop-down menu. If wired to an invalid value, an error is generated (the value must be valid for the type of module specified and at the specified address). The default value is 1 (4–20 mA).

**U8** **Module Type** is the type name of the module from an enumerated control.

**abc** **IP Address in:** is the address of the FieldPoint network module, in XXX.XXX.XXX.XXX format (leading zeros excluded). Provide only the IP address, not a complete URL. If the terminal is unwired or invalid, an error occurs.

**U8** **Module Address** is the address of the module for which you are configuring the range. The first module after the network module is address 1; the second is address 2, and so on. If the terminal is unwired, an error occurs.

**U8** **Channel In** is the selected module channel for attribute configuration. If the terminal is unwired, the default value is zero. If the terminal is wired to an invalid value, an error occurs.

 **error in (no error)** describes error conditions that occur before this VI or function runs. The default is no error. If an error occurred before this VI or function runs, the VI or function passes the **error in (no error)** value to **error out**. This VI or function runs normally only if no error occurred before this VI or function runs. If an error occurs while this VI or function runs, it runs normally and sets its own error status in **error out**. Use the Simple Error Handler or General Error Handler VIs to display the description of the error code. 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.

 **status** is TRUE (X) if an error occurred before this VI or function ran or FALSE (checkmark) to indicate a warning or that no error occurred before this VI or function ran. The default is FALSE.

 **code** is the error or warning code. The default is 0. If **status** is TRUE, **code** is a non-zero error code. If **status** is FALSE, **code** is 0 or a warning code.

 **source** describes the origin of the error or warning and is, in most cases, the name of the VI or function that produced the error or warning. The default is an empty string.

 **error out** contains error information. If **error in** indicates that an error occurred before this VI or function ran, **error out** contains the same error information. Otherwise, it describes the error status that this VI or function produces. Right-click the **error out** indicator on the front panel and select **Explain Error** from the shortcut menu for more information about the error.

 **status** is TRUE (X) if an error occurred or FALSE (checkmark) to indicate a warning or that no error occurred.

 **code** is the error or warning code. If **status** is TRUE, **code** is a non-zero error code. If **status** is FALSE, **code** is 0 or a warning code.

 **source** describes the origin of the error or warning and is, in most cases, the name of the VI or function that produced the error or warning.

## FieldPoint Publish Data VIs

Use these VIs to share LabVIEW data between two FieldPoint RT controllers or a FieldPoint RT controller and a local computer. The data is published under a block (folder) name and item name.

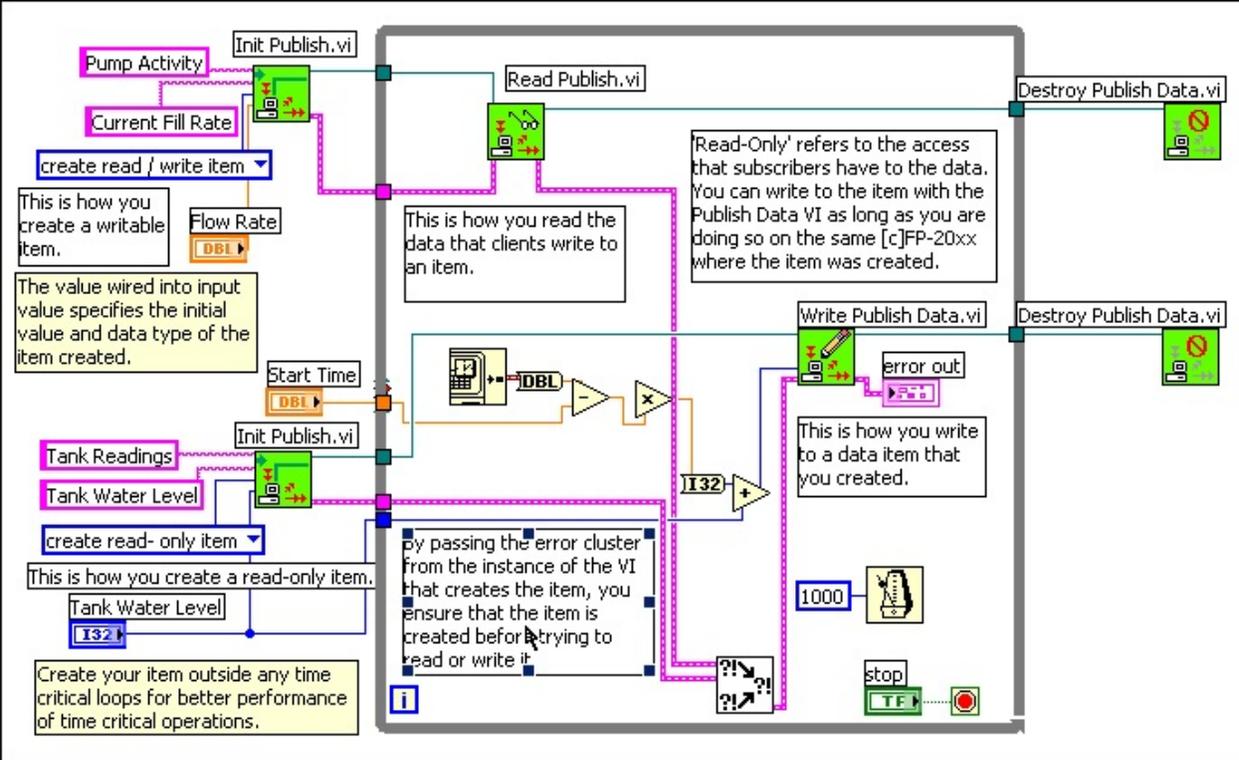


**Note** This palette appears only if LabVIEW Real-Time is installed.



The Publish Data palette contains subVIs for individual publish operations: Init Publish Data, Read Publish Data, Write Publish Data, and Destroy Publish Data. Publish Data.vi merges all four functions into one subVI. For faster performance, use the individual, green subVIs.

The following figure shows a typical control application using Init Publish Data, Read Publish Data, Write Publish Data, and Destroy Publish Data. This application controls the level of water in a tank by varying the fill rate of a pump. Clients on the network can view the level of the tank and the fill rate of the pump.

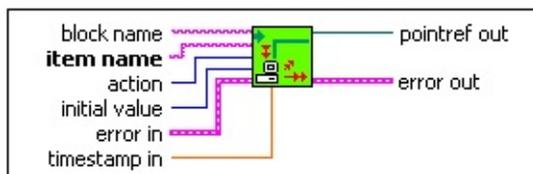


# Init Publish Data

Creates a LabVIEW item that can be subscribed to over a network or on the local computer. Use this VI along with the Read Publish Data and Write Publish Data VIs to share LabVIEW data between two FieldPoint RT controllers or a FieldPoint RT controller and a local computer. The data is published under a block (folder) name and item name. [Details](#)



**Note** Do not run this VI on the host computer. This VI must run embedded on a FieldPoint RT controller.



**abc** **block name** is the name of the block (folder) under which the data is published. If you leave this terminal unwired, the data is published under the default block, RTPublished. If the block does not exist when the VI is called, it is created. Block names are not case sensitive.

**abc** **item name** is the name under which data is published. Item names are not case sensitive.

**US** **action** indicates what action the VI should take. Right-click the terminal, select **Create»Constant**, and select the desired action. You can wire integer values to this terminal as follows: 0 to create a read/write item, 1 to create a read-only item. Other values result in an error. [Details](#)

**POLY** **initial value** provides the value that the data item is created with and determines the data type of the item. If the item already exists and the value wired to this terminal is not the correct data type, an error results. Permissible data types are I32, Double, Boolean, and String.

**FFT** **error in (no error)** describes error conditions that occur before this VI or function runs. The default is no error. If an error occurred before this VI or function runs, the VI or function passes the **error in (no error)** value to **error out**. This VI or function runs normally only if no error occurred before this VI or function runs. If an error occurs while this VI or function runs, it runs normally and sets its own error status in **error out**. Use the Simple Error Handler or General Error Handler VIs to display the

description of the error code. 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.

 **status** is TRUE (X) if an error occurred before this VI or function ran or FALSE (checkmark) to indicate a warning or that no error occurred before this VI or function ran. The default is FALSE.

 **code** is the error or warning code. The default is 0. If **status** is TRUE, **code** is a non-zero error code. If **status** is FALSE, **code** is 0 or a warning code.

 **source** describes the origin of the error or warning and is, in most cases, the name of the VI or function that produced the error or warning. The default is an empty string.

 **timestamp in** enables you to assign a custom timestamp to the new item. If this terminal is not wired, Init Publish Data uses the system time for the initial timestamp value.

 **refnum out** is a unique number that Init Publish Data assigns to the data item. Pass this value to Read Publish Data or Write Publish Data if you want those VIs to read from or write to the data item. Pass this value to Destroy Publish Data Item if you want that VI to destroy the data item.

 **error out** contains error information. If **error in** indicates that an error occurred before this VI or function ran, **error out** contains the same error information. Otherwise, it describes the error status that this VI or function produces. Right-click the **error out** indicator on the front panel and select **Explain Error** from the shortcut menu for more information about the error.

 **status** is TRUE (X) if an error occurred or FALSE (checkmark) to indicate a warning or that no error occurred.

 **code** is the error or warning code. If **status** is TRUE, **code** is a non-zero error code. If **status** is FALSE, **code** is 0 or a warning code.

 **source** describes the origin of the error or warning and is, in most cases, the name of the VI or function that produced the error or warning.

## Init Publish Data Details

You must run any VI that contains the Init Publish Data VI at least once before adding the data items from the VI to Measurement & Automation Explorer (MAX).

To add a data item from the Init Publish Data VI to MAX, complete the following steps.

1. Make sure LabVIEW Real-Time (RT) is targeted to a FieldPoint RT controller. Refer to the *LabVIEW Real-Time User Manual* for more information about targeting LabVIEW RT to a controller.
2. Place an Init Publish Data VI on the block diagram.
3. Right-click the Init Publish Data VI. Select **Select Type** and select the data type for the new item you want to create.
4. Double-click the Init Publish Data VI.
5. On the Init Publish Data VI front panel, enter values in the **block name** and **item name** fields.
6. Select an action from the **action** menu. Select **Create read-only item** for an item that clients can view over the network but cannot change. Select **Create read/write item** for an item that clients can view and change over the network.
7. Run the Init Publish Data VI.
8. In MAX, right-click the FieldPoint RT controller under **My System»Data Neighborhood** and select **Create New Item** from the shortcut menu.
9. In the **Create New** dialog box, select **LabVIEW Item**. Click the **Continue** button

0. Select the item from the **Address of this item** field.
1. In the **Create New LabVIEW Item** dialog box enter a name for the LabVIEW item in the **Name** field. Click the **OK** button.
2. Save the .iak configuration file.

The data published by this VI can be read or written to over a network by subscribing to the block and item names, using the DataSocket Read or DataSocket Write VIs. The value of an item may also be read or changed on the local machine where it was created by using this VI and passing in the same block and item name.

## Action Details

If **action** is **create read/write item** and **item** does not exist, the item is created with read/write privileges for subscribers and is given an initial value of the value wired in at **input value**.

If **action** is **create read/write item** and **item** exists, a warning is issued on **error out**, and the value of the item is updated to the value wired in at **input value**.

If **action** is **create read-only item** and **item** does not exist, the item is created with read-only privileges for subscribers, and given an initial value of the value wired in at **input value**. Note that read-only items are read-only to clients, but [Write Publish Data](#) can change them.

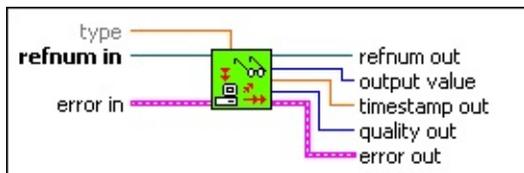
If **action** is **create read-only item** and **item** exists, a warning is issued on **error out**, and the value of the item is updated to the value wired in at **input value**.

# Read Publish Data

Reads LabVIEW data published over a network. [Details](#)



**Note** Do not run this VI on the host computer. This VI must run embedded on a FieldPoint RT controller.



-  **type** determines the data type of **output value**. Valid data types are Float (DBL), Float Array, Boolean, and Boolean Array. The data type must match that of the **initial value** terminal of [Init Publish Data](#).
-  **refnum in** is a unique number that Init Publish Data assigns to the data item. Wire **refnum out** from Init Publish Data to this terminal.
-  **error in (no error)** describes error conditions that occur before this VI or function runs. The default is no error. If an error occurred before this VI or function runs, the VI or function passes the **error in (no error)** value to **error out**. This VI or function runs normally only if no error occurred before this VI or function runs. If an error occurs while this VI or function runs, it runs normally and sets its own error status in **error out**. Use the Simple Error Handler or General Error Handler VIs to display the description of the error code. 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.
  -  **status** is TRUE (X) if an error occurred before this VI or function ran or FALSE (checkmark) to indicate a warning or that no error occurred before this VI or function ran. The default is FALSE.
  -  **code** is the error or warning code. The default is 0. If **status** is TRUE, **code** is a non-zero error code. If **status** is FALSE, **code** is 0 or a warning code.
  -  **source** describes the origin of the error or warning and is, in most cases, the name of the VI or function that produced the error or warning. The default is an empty string.
-  **refnum out** is a passthrough terminal that returns the value of **refnum in**.

-  **output value** returns the new value of the data item.
-  **timestamp out** provides the time read from the data item the last time that it was successfully written.
-  **quality out** is a quality stamp read from the data item. A value of zero is used for good quality.
-  **error out** contains error information. If **error in** indicates that an error occurred before this VI or function ran, **error out** contains the same error information. Otherwise, it describes the error status that this VI or function produces. Right-click the **error out** indicator on the front panel and select **Explain Error** from the shortcut menu for more information about the error.
  -  **status** is TRUE (X) if an error occurred or FALSE (checkmark) to indicate a warning or that no error occurred.
  -  **code** is the error or warning code. If **status** is TRUE, **code** is a non-zero error code. If **status** is FALSE, **code** is 0 or a warning code.
  -  **source** describes the origin of the error or warning and is, in most cases, the name of the VI or function that produced the error or warning.

## Read Publish Data Details

You must run the [Init Publish Data VI](#) at least once before adding the data items from the VI to Measurement & Automation Explorer (MAX).

To add a data item from the Init Publish Data VI to MAX, complete the following steps.

1. Make sure LabVIEW Real-Time (RT) is targeted to a FieldPoint RT controller. Refer to the *LabVIEW Real-Time User Manual* for more information about targeting LabVIEW RT to a controller.
2. Place an Init Publish Data VI on the block diagram.
3. Right-click the Init Publish Data VI. Select **Select**

- Type** and select the data type for the new item you want to create.
4. Double-click the Init Publish Data VI.
  5. On the Init Publish Data VI front panel, enter values in the **block name** and **item name** fields.
  6. Select an action from the **action** menu.
  7. Run the Init Publish Data VI.
  8. In MAX, right-click the FieldPoint RT controller under **My System»Data Neighborhood** and select **Create New Item** from the shortcut menu.
  9. In the **Create New** dialog box, select **LabVIEW Item**. Click the **Continue** button
  10. Select the item from the **Address of this item** field.
  11. In the **Create New LabVIEW Item** dialog box enter a name for the LabVIEW item in the **Name** field. Click the **OK** button.
  12. Save the .iak configuration file.

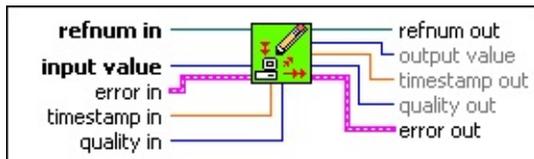
The data published by this VI can be read or written to over a network by subscribing to the block and item names, using the DataSocket Read or DataSocket Write VIs. The value of an item may also be read or changed on the local machine where it was created by using the Read Publish Data and Write Publish Data VIs and passing in the same block and item name.

# Write Publish Data

Writes a new value to a LabVIEW data item created by [Init Publish Data](#). This VI can write even to data items that are read-only to clients. [Details](#)



**Note** Do not run this VI on the host computer. This VI must run embedded on a FieldPoint RT controller.



**refnum in** is a unique number that Init Publish Data assigns to the data item. Wire **refnum out** from [Init Publish Data](#) to this terminal.



**input value** is the new value to write to the data item. If the data type of **input value** does not match that of the data item, Write Publish Data returns an error.



**error in (no error)** describes error conditions that occur before this VI or function runs. The default is no error. If an error occurred before this VI or function runs, the VI or function passes the **error in (no error)** value to **error out**. This VI or function runs normally only if no error occurred before this VI or function runs. If an error occurs while this VI or function runs, it runs normally and sets its own error status in **error out**. Use the Simple Error Handler or General Error Handler VIs to display the description of the error code. 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.



**status** is TRUE (X) if an error occurred before this VI or function ran or FALSE (checkmark) to indicate a warning or that no error occurred before this VI or function ran. The default is FALSE.



**code** is the error or warning code. The default is 0. If **status** is TRUE, **code** is a non-zero error code. If **status** is FALSE, **code** is 0 or a warning code.



**source** describes the origin of the error or warning and is, in most cases, the name of the VI or function that produced the error or warning. The default is an empty string.

-  **timestamp in** enables you to assign a custom timestamp to the new item. If this terminal is not wired, Write Publish Data uses the system time for the initial timestamp value.
-  **quality in** indicates the quality of the published data. Good quality is a zero value (the default).
-  **refnum out** is a passthrough terminal that returns the value of **refnum in**.
-  **output value** returns the value of the data item after the write operation.
-  **timestamp out** returns the last time that a value was successfully written to the data item.
-  **quality out** is a quality stamp read from the data item. A value of zero is used for good quality.
-  **error out** contains error information. If **error in** indicates that an error occurred before this VI or function ran, **error out** contains the same error information. Otherwise, it describes the error status that this VI or function produces. Right-click the **error out** indicator on the front panel and select **Explain Error** from the shortcut menu for more information about the error.
  -  **status** is TRUE (X) if an error occurred or FALSE (checkmark) to indicate a warning or that no error occurred.
  -  **code** is the error or warning code. If **status** is TRUE, **code** is a non-zero error code. If **status** is FALSE, **code** is 0 or a warning code.
  -  **source** describes the origin of the error or warning and is, in most cases, the name of the VI or function that produced the error or warning.

## Write Publish Data Details

You must run the Init Publish Data VI at least once before adding the data items from the VI to Measurement & Automation Explorer (MAX).

To add a data item from the Read Publish Data VI to MAX, complete the following steps.

1. Make sure LabVIEW Real-Time (RT) is targeted to a

FieldPoint RT controller. Refer to the *LabVIEW Real-Time User Manual* for more information about targeting LabVIEW RT to a controller.

2. Place an Init Publish Data VI on the block diagram.
3. Right-click the Init Publish Data VI. Select **Select Type** and select the data type for the new item you want to create.
4. Double-click the Init Publish Data VI.
5. On the Init Publish Data VI front panel, enter values in the **block name** and **item name** fields.
6. Select an action from the **action** menu.
7. Run the Init Publish Data VI.
8. In MAX, right-click the FieldPoint RT controller under **My System»Data Neighborhood** and select **Create New Item** from the shortcut menu.
9. In the **Create New** dialog box, select **LabVIEW Item**. Click the **Continue** button
10. Select the item from the **Address of this item** field.
11. In the **Create New LabVIEW Item** dialog box enter a name for the LabVIEW item in the **Name** field. Click the **OK** button.
12. Save the .iak configuration file.

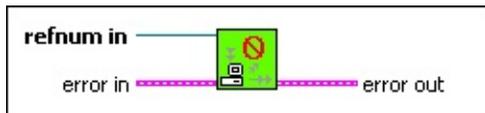
The data published by this VI can be read or written to over a network by subscribing to the block and item names, using the DataSocket Read or DataSocket Write VIs. The value of an item may also be read or changed on the local machine where it was created by running the Read Publish Data and Write Publish Data VIs and passing in the same block and item name.

# Destroy Publish Data

Destroys a LabVIEW data item created by [Init Publish Data](#) so that the item is no longer available to clients.



**Note** Do not run this VI on the host computer. This VI must run embedded on a FieldPoint RT controller.



**refnum in** is a unique number that Init Publish Data assigns to the data item. Wire **refnum out** from Init Publish Data to this terminal.



**error in (no error)** describes error conditions that occur before this VI or function runs. The default is no error. If an error occurred before this VI or function runs, the VI or function passes the **error in (no error)** value to **error out**. This VI or function runs normally only if no error occurred before this VI or function runs. If an error occurs while this VI or function runs, it runs normally and sets its own error status in **error out**. Use the Simple Error Handler or General Error Handler VIs to display the description of the error code. 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.



**status** is TRUE (X) if an error occurred before this VI or function ran or FALSE (checkmark) to indicate a warning or that no error occurred before this VI or function ran. The default is FALSE.



**code** is the error or warning code. The default is 0. If **status** is TRUE, **code** is a non-zero error code. If **status** is FALSE, **code** is 0 or a warning code.



**source** describes the origin of the error or warning and is, in most cases, the name of the VI or function that produced the error or warning. The default is an empty string.



**error out** contains error information. If **error in** indicates that an error occurred before this VI or function ran, **error out** contains the same error information. Otherwise, it describes the error status that this VI or function produces. Right-click the **error out** indicator on the front panel

and select **Explain Error** from the shortcut menu for more information about the error.



**status** is TRUE (X) if an error occurred or FALSE (checkmark) to indicate a warning or that no error occurred.



**code** is the error or warning code. If **status** is TRUE, **code** is a non-zero error code. If **status** is FALSE, **code** is 0 or a warning code.



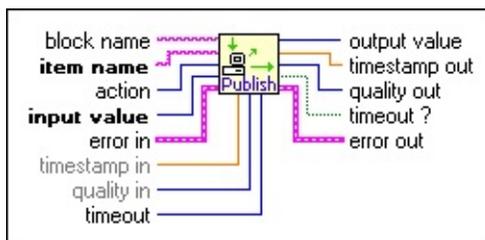
**source** describes the origin of the error or warning and is, in most cases, the name of the VI or function that produced the error or warning.

# Publish Data

Publishes LabVIEW data that can be subscribed to over a network or on the local computer. You can use this VI to share LabVIEW data between two FieldPoint RT controllers or a FieldPoint RT controller and a local computer. The data is published under a block (folder) name and item name. This VI merges all the operations of Init Publish Data, Read Publish Data, Write Publish Data, and Destroy Publish Data. [Details](#)



**Note** Do not run this VI on the host computer. This VI must run embedded on a FieldPoint RT controller.



- abc** **block name** is the name of the block (folder) under which the data is published. If you leave this terminal unwired, the data is published under the default block, RTPublished. If the block does not exist when the VI is called, it is created. Block names are not case sensitive.
- abc** **item name** is the name under which data is published. Item names are not case sensitive.
- US** **action** indicates what action the VI should take. Right-click the terminal, select **Create»Constant**, and select the desired action. Integer values may also be wired in as follows: 0 to create read/write, 1 to create read-only, 2 to read, 3 to write, 4 to delete item. Other values result in an error. [Details](#)
- POLY** **input value** provides the value to be written to the data item. The [action](#) input determines the behavior of the **input value** terminal. If the item does not exist, an item is created with the data type and initial value wired to this terminal. If the item does exist and the value wired to this terminal is not the correct data type, an error results (unless the action is **delete item**). Permissible data types are I32, Double, Boolean, and String.
- ERR** **error in (no error)** describes error conditions that occur before this VI or function runs. The default is no error. If an error occurred before this VI

or function runs, the VI or function passes the **error in (no error)** value to **error out**. This VI or function runs normally only if no error occurred before this VI or function runs. If an error occurs while this VI or function runs, it runs normally and sets its own error status in **error out**. Use the Simple Error Handler or General Error Handler VIs to display the description of the error code. 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.

 **status** is TRUE (X) if an error occurred before this VI or function ran or FALSE (checkmark) to indicate a warning or that no error occurred before this VI or function ran. The default is FALSE.

 **code** is the error or warning code. The default is 0. If **status** is TRUE, **code** is a non-zero error code. If **status** is FALSE, **code** is 0 or a warning code.

 **source** describes the origin of the error or warning and is, in most cases, the name of the VI or function that produced the error or warning. The default is an empty string.

 **timestamp in** provides the value of the timestamp associated with the data item. You can use it to assign a custom timestamp to the item. If this terminal is not wired, Publish Data writes the system time to the timestamp of the item.

 **quality in** indicates the quality of the published data. Good quality is a zero value (the default).

 **timeout** provides the maximum specified period of time, in milliseconds, for the VI to wait for a change in the value of the data item before returning. The default timeout value is zero. If a zero value is written, the VI does not wait. If a value less than zero is written, the read function never times out.

 **Note** If you wire in a negative timeout value, the VI does not return until the value of the data item changes. Using negative timeouts with read-only items is strongly discouraged.

 **output value** returns the new value of the data item, if it differs from the value at **input value** prior to the end of the timeout period. If a timeout occurs, the VI returns the value at **input value**.

 **timestamp out** provides the time read from the data item the last time that it was successfully written.

 **quality out** is a quality stamp read from the data item. A value of zero is

used for good quality.

 **timeout ?** returns TRUE if a timeout occurred when reading the item. A read timeout does not generate an error condition.

 **error out** contains error information. If **error in** indicates that an error occurred before this VI or function ran, **error out** contains the same error information. Otherwise, it describes the error status that this VI or function produces. Right-click the **error out** indicator on the front panel and select **Explain Error** from the shortcut menu for more information about the error.

 **status** is TRUE (X) if an error occurred or FALSE (checkmark) to indicate a warning or that no error occurred.

 **code** is the error or warning code. If **status** is TRUE, **code** is a non-zero error code. If **status** is FALSE, **code** is 0 or a warning code.

 **source** describes the origin of the error or warning and is, in most cases, the name of the VI or function that produced the error or warning.

## Publish Data Details

You must run any VI that contains the Publish Data VI at least once before adding the data items from the VI to Measurement & Automation Explorer (MAX).

To add a data item from the Publish Data VI to MAX, complete the following steps.

1. Make sure LabVIEW Real-Time (RT) is targeted to a FieldPoint RT controller. Refer to the *LabVIEW Real-Time User Manual* for more information about targeting LabVIEW RT to a controller.
2. Place a Publish Data VI on the block diagram.
3. Right-click the Publish Data VI. Select **Select Type**

and select the data type for the new item you want to create.

4. Double-click the Publish Data VI.
5. On the Publish Data VI front panel, enter values in the **block name** and **item name** fields.
6. Select an action from the **action** menu.
7. Run the Publish Data VI.
8. In MAX, right-click the FieldPoint RT controller under **My System»Data Neighborhood** and select **Create New Item** from the shortcut menu.
9. In the **Create New** dialog box, select **LabVIEW Item**. Click the **Continue** button
10. Select the item from the **Address of this item** field.
11. In the **Create New LabVIEW Item** dialog box enter a name for the LabVIEW item in the **Name** field. Click the **OK** button.
12. Save the .iak configuration file.

The data published by this VI can be read or written to over a network by subscribing to the block and item names, using the DataSocket Read or DataSocket Write VIs. The value of an item may also be read or changed on the local machine where it was created by using this VI and passing in the same block and item name.

## Action Details

If **action** is **create read/write item** and **item** does not exist, the item is created with read/write privileges for subscribers and is given an initial

value of the value wired in at **input value**. The VI then waits for the specified timeout period for the value to be changed, and returns the result in **output value**.

If **action** is **create read/write item** and **item** exists, a warning is issued on **error out**, and the value of the item is updated to the value wired in at **input value**. The VI waits for the specified timeout period for the value to be changed from the value wired in at input value, and returns the result in **output value**.

If **action** is **create read-only item** and **item** does not exist, the item is created with read-only privileges for subscribers, and given an initial value of the value wired in at **input value**. Note that read-only items are read-only to subscribers, but this VI may change them by wiring in the block and item name and writing to them. The VI then waits for the specified timeout period for the value to be changed, and returns the result in **output value**.

If **action** is **create read-only item** and **item** exists, a warning is issued on **error out**, and the value of the item is updated to the value wired in at **input value**. The VI waits for the specified timeout period for the value to be changed from the value wired in at input value, and returns the result in **output value**.

If **action** is **read item** (default action) and **item** does not exist, the item is created with read/write privileges for subscribers, and given an initial value of the value wired in at **input value**. The VI then waits for the specified timeout period for the value to be changed, and returns the result in **output value**.

If **action** is **read item** (default action) and **item** exists, the VI waits for the specified timeout period for the value to be changed from the value wired in at **input value**, and returns the result in **output value**.

If **action** is **write item** and **item** does not exist, the item is created with read/write privileges for subscribers, and given an initial value of the value wired in at **input value**. The VI then waits for the specified timeout

period for the value to be changed, and returns the result in **output value**.

If **action** is **write item** and **item** exists, the value of the item is updated to the value wired in at **input value**. The VI waits for the specified timeout period for the value to be changed from the value just written, and returns the result in **output value**.

If **action** is **delete item** and **item** does not exist, a warning is issued on error out. The VI does not wait for the specified timeout period or attempt to read the item, but returns the value wired in at **input value** as the **output value**, and default value for all other outputs.

If **action** is **delete item** and **item** exists, the item is removed from publication and all client subscriptions show bad quality for the item data. The data is no longer updated to clients, and clients are not able to browse the item. If there are no more items remaining in the block that contained the item data, it is deleted as well. The VI does not wait for the specified timeout period or attempt to read the item, but returns the value wired in at input value as the output value, and default values for all other outputs.

## **[c]FP-AI-100**

All ID values are hexadecimal.

## **Module ID**

010A

## Channel Status

Message	ID
Out of range	01

## Ranges

Name	ID
024 mA	00
3.524 mA	01
±24 mA	02
±6 V	05
06 V	06
±1.2 V	07
01.2 V	08
018 V	0E
±36 V	0F
036 V	11
±18 V	12

**No Attributes**

**No Commands**

## **[c]FP-AI-102**

All ID values are hexadecimal.

**Module ID**

0115

## Channel Status

Message	ID
Out of range	01

## Ranges

Name	ID
020 V	13
±20 V	14
060 V	15
±60 V	16
0120 V	17
±120 V	18

**No Attributes**

**No Commands**

## **[c]FP-AI-110**

All ID values are hexadecimal.

## **Module ID**

0101

## Channel Status

Message	ID
Out of range	01

## Ranges

Name	ID
021 mA	00
3.521 mA	01
±21 mA	02
±10.4 V	03
010.4 V	04
±5.2 V	05
05.2 V	06
±1.04 V	07
01.04 V	08
±325 mV	09
±65 mV	0A

## Attributes

Name	ID	Value	ID
Noise Rejection	01	60 Hz	00
		50 Hz	01
		500 Hz	02

**No Commands**

## **[c]FP-AI-111**

All ID values are hexadecimal.

## **Module ID**

010C

## Channel Status

Message	ID
Out of range	01

## Ranges

Name	ID
021 mA	00
3.521 mA	01
±21 mA	02

## Attributes

Name	ID	Value	ID
Noise Rejection	01	60 Hz	00
		50 Hz	01
		500 Hz	02

**No Commands**

## **cFP-AI-118**

All ID values are hexadecimal.

## **Module ID**

0127

## Channel Status

Message	ID
Out of range	01

## Ranges

Name	ID
$\pm 10.4$ V	03
010.4 V	04
$\pm 5.2$ V	05
05.2 V	06
$\pm 1.04$ V	07
01.04 V	08
018 V	0E
$\pm 18$ V	12

## Attributes

Name	ID	Value	ID
Noise Rejection 01		10 Hz	0B
		None	FE

**No Commands**

## **FP-AI-C020**

All ID values are hexadecimal.

## Channel Status

Message	ID
Empty	01
Out of range	02

## Ranges

Name	ID
020 mA	00

## Attributes

Name	ID	Value	ID
Input Filter 01		20 Hz	07
		100 Hz	06

**No Commands**

## **FP-AI-C420**

All ID values are hexadecimal.

## Channel Status

Message	ID
Empty	01
Out of range	02

## Ranges

Name	ID
420 mA	01

## Attributes

Name	ID	Value	ID
Input Filter 01		20 Hz	07
		100 Hz	06

**No Commands**

## **[c]FP-AIO-600**

All ID values are hexadecimal.

## **Module ID**

0123

## Channel Status

Message	ID
Overcurrent protection (inputs)	01
Open current loop (outputs)	02

## Ranges

Type	Name	ID
Input	024 mA	00
	3.524 mA	01
	±24 mA	02
	±12 V	03
	012 V	04
	±6 V	05
	06 V	06
	018 V	0E
	±36 V	0F
	036 V	11
	±18 V	12
	Output	021 mA
3.521 mA		01

**No Attributes**

**No Commands**

## **[c]FP-AIO-610**

All ID values are hexadecimal.

**Module ID**

011B

## Channel Status

Message	ID
Overcurrent protection (inputs)	01
Overcurrent protection (outputs)	07

## Ranges

Type	Name	ID
Input	024 mA	00
	3.524 mA	01
	±24 mA	02
	±12 V	03
	012 V	04
	±6 V	05
	06 V	06
	018 V	0E
	±36 V	0F
	036 V	11
	±18 V	12
	Output	±12 V
012 V		04

**No Attributes**

**No Commands**

## **FP-AI-V1**

All ID values are hexadecimal.

## Channel Status

Message	ID
Empty	01
Out of range	02

## Ranges

Name	ID
01 V	08

## Attributes

Name	ID	Value	ID
Input Filter	01	20 Hz	07
		2 kHz	05

**No Commands**

## **FP-AI-V5**

All ID values are hexadecimal.

## Channel Status

Message	ID
Empty	01
Out of range	02

## Ranges

Name	ID
05 V	06

## Attributes

Name	ID	Value	ID
Input Filter	01	20 Hz	07
		2 kHz	05

**No Commands**

## **FP-AI-V5B**

All ID values are hexadecimal.

## Channel Status

Message	ID
Empty	01
Out of range	02

## Ranges

Name	ID
$\pm 5\text{ V}$	05

## Attributes

Name	ID	Value	ID
Input Filter	01	20 Hz	07
		2 kHz	05

**No Commands**

## **FP-AI-V10**

All ID values are hexadecimal.

## Channel Status

Message	ID
Empty	01
Out of range	02

## Ranges

Name	ID
010 V	04

## Attributes

Name	ID	Value	ID
Input Filter	01	20 Hz	07
		2 kHz	05

**No Commands**

## **FP-AI-V10B**

All ID values are hexadecimal.

## Channel Status

Message	ID
Empty	01
Out of range	02

## Ranges

Name	ID
$\pm 10$ V	03

## Attributes

Name	ID	Value	ID
Input Filter	01	20 Hz	07
		2 kHz	05

**No Commands**

## **FP-AI-V50m**

All ID values are hexadecimal.

## Channel Status

Message	ID
Empty	01
Out of range	02

## Ranges

Name	ID
050 mV	19

## Attributes

Name	ID	Value	ID
Input Filter	01	20 Hz	07
		500 Hz	02

**No Commands**

## **FP-AI-V100m**

All ID values are hexadecimal.

## Channel Status

Message	ID
Empty	01
Out of range	02

## Ranges

Name	ID
0100 mV	1A

## Attributes

Name	ID	Value	ID
Input Filter	01	1 kHz	08
		20 Hz	07

**No Commands**

## **[c]FP-AO-200**

All ID values are hexadecimal.

## **Module ID**

0102

## Channel Status

Message	ID
Open current loop	01

## Ranges

Name	ID
021 mA	00
3.521 mA	01

**No Attributes**

**No Commands**

## **[c]FP-AO-210**

All ID values are hexadecimal.

## **Module ID**

010F

## Channel Status

Message	ID
Overcurrent protection	01

## Ranges

Name	ID
010.2 V	04

**No Attributes**

**No Commands**

## **FP-AO-C020**

All ID values are hexadecimal.

## Channel Status

Message	ID
Empty	01

## Ranges

Name	ID
020 mA	00

**No Attributes**

**No Commands**

## **FP-AO-C024**

All ID values are hexadecimal.

## Channel Status

Message	ID
Empty	01

## Ranges

Name	ID
0-24 mA	00

**No Attributes**

**No Commands**

## **FP-AO-C420**

All ID values are hexadecimal.

## Channel Status

Message	ID
Empty	01

## Ranges

Name	ID
420 mA	01

**No Attributes**

**No Commands**

## **FP-AO-V5**

All ID values are hexadecimal.

## Channel Status

Message	ID
Empty	01

## Ranges

Name	ID
05 V	06

**No Attributes**

**No Commands**

## **FP-AO-V5B**

All ID values are hexadecimal.

## Channel Status

Message	ID
Empty	01

## Ranges

Name	ID
$\pm 5\text{ V}$	05

**No Attributes**

**No Commands**

## **FP-AO-V10**

All ID values are hexadecimal.

## Channel Status

Message	ID
Empty	01

## Ranges

Name	ID
010 V	04

**No Attributes**

**No Commands**

## **FP-AO-V10B**

All ID values are hexadecimal.

## Channel Status

Message	ID
Empty	01

## Ranges

Name	ID
$\pm 10$ V	03

**No Attributes**

**No Commands**

## **[c]FP-CTR-500**

All ID values are hexadecimal.

## **Module ID**

010D

## Channel Status

Message	ID
Overflow since last read	01

## Ranges

Name	ID
Boolean	10
065535 Counts	40

## Attributes

Name	ID	Value	ID
Terminal Count	05	065535	
Terminal Count MSB [Most Significant Byte]	06		
Count Source	07	External Count Input	00
		Previous Channel	01
		1 kHz Reference	02
		32 kHz Reference	03
Gate Source	08	Gate Input 0	00
		Gate Input 1	01
		Gate Input 2	02
		Gate Input 3	03
		Always Disabled	04
		Always Enabled	05
Read Reset Mode	09	Dont Reset On Read	00
		Reset On Read	01
Noise Rejection	01	200 Hz	03
		50 kHz	04
Output Source	0A	Counter Channel 0	00
		Counter Channel 1	01
		Counter Channel 2	02
		Counter Channel 3	03
		Counter Channel 4	04
		Counter Channel 5	05
		Counter Channel 6	06
		Counter Channel 7	07
		Discrete Data	08
Output Mode	0B	Toggle, Reset Off	00
		Toggle, Reset On	01
		On Pulse	02
		Off Pulse	03

## Commands

Name	ID	Action	ID
Control	01	Increment	02
		Reset	01

## **[c]FP-CTR-502**

All ID values are hexadecimal.

## **Module ID**

0114

## Channel Status

Message	ID
Overflow since last read	01

## Ranges

Name	ID
Boolean	10
065535 Counts	40

## Attributes

Name	ID	Value	ID
Terminal Count	05	065535	
Terminal Count MSB [Most Significant Byte]	06		
Count Source	07	External Count Input	00
		Previous Channel	01
		1 kHz Reference	02
		32 kHz Reference	03
Gate Source	08	Gate Input 0	00
		Gate Input 1	01
		Gate Input 2	02
		Gate Input 3	03
		Always Disabled	04
		Always Enabled	05
Read Reset Mode	09	Dont Reset On Read	00
		Reset On Read	01
Noise Rejection	01	200 Hz	03
		50 kHz	04
Output Source	0A	Counter Channel 0	00
		Counter Channel 1	01
		Counter Channel 2	02
		Counter Channel 3	03
		Counter Channel 4	04
		Counter Channel 5	05
		Counter Channel 6	06
		Counter Channel 7	07
		Discrete Data	08
Output Mode	0B	Toggle, Reset Off	00
		Toggle, Reset On	01
		On Pulse	02
		Off Pulse	03

## Commands

Name	ID	Action	ID
Control	01	Increment	02
		Reset	01

## **[c]FP-DI-300**

All ID values are hexadecimal.

## **Module ID**

0109

## Ranges

Name	ID
Boolean	10

**No Attributes**

**No Commands**

## **[c]FP-DI-301**

All ID values are hexadecimal.

## **Module ID**

0105

## Ranges

Name	ID
Boolean	10

**No Attributes**

**No Commands**

## **cFP-DI-304**

All ID values are hexadecimal.

## **Module ID**

0129

## Ranges

Name	ID
Boolean	10

**No Attributes**

**No Commands**

## **[c]FP-DI-330**

All ID values are hexadecimal.

## **Module ID**

0103

## Ranges

Name	ID
Boolean	10

**No Attributes**

**No Commands**

## **FP-DI-AC120**

All ID values are hexadecimal.

## Channel Status

Message	ID
Empty	01

## Ranges

Name	ID
Boolean	10

**No Attributes**

**No Commands**

## **FP-DI-AC240**

All ID values are hexadecimal.

## Channel Status

Message	ID
Empty	01

## Ranges

Name	ID
Boolean	10

**No Attributes**

**No Commands**

## **FP-DI-DC**

All ID values are hexadecimal.

## Channel Status

Message	ID
Empty	01

## Ranges

Name	ID
Boolean	10

**No Attributes**

**No Commands**

## **[c]FP-DIO-550**

All ID values are hexadecimal.

## **Module ID**

0126

## Ranges

Name	ID
Boolean	10

**No Attributes**

**No Commands**

## **[c]FP-DO-400**

All ID values are hexadecimal.

## **Module ID**

0104

## Ranges

Name	ID
Boolean	10

**No Attributes**

**No Commands**

## **[c]FP-DO-401**

All ID values are hexadecimal.

**Module ID**

0106

## Ranges

Name	ID
Boolean	10

**No Attributes**

**No Commands**

## **[c]FP-DO-403**

All ID values are hexadecimal.

**Module ID**

0111

## Ranges

Name	ID
Boolean	10

**No Attributes**

**No Commands**

## **[c]FP-DO-410**

All ID values are hexadecimal.

## **Module ID**

0110

## Channel Status

Message	ID
Current limited	07

## Ranges

Name	ID
Boolean	10

**No Attributes**

**No Commands**

## **FP-DO-AC120**

All ID values are hexadecimal.

## Channel Status

Message	ID
Empty	01

## Ranges

Name	ID
Boolean	10

**No Attributes**

**No Commands**

## **FP-DO-AC240**

All ID values are hexadecimal.

## Channel Status

Message	ID
Empty	01

## Ranges

Name	ID
Boolean	10

**No Attributes**

**No Commands**

## **FP-DO-DC60**

All ID values are hexadecimal.

## Channel Status

Message	ID
Empty	01

## Ranges

Name	ID
Boolean	10

**No Attributes**

**No Commands**

## **FP-DO-DC200**

All ID values are hexadecimal.

## Channel Status

Message	ID
Empty	01

## Ranges

Name	ID
Boolean	10

**No Attributes**

**No Commands**

## **[c]FP-PG-522**

All ID values are hexadecimal.

**Module ID**

0113

## Ranges

Name	ID
Boolean	10
065535 Counts	40

## Attributes

Name	ID	Value	ID
Pulse Mode	0E	Finite	00
		Continuous	01
On Time	0F	165535	
On Time MSB [Most Significant Byte]	10		
Off Time	11	065535	
Off Time MSB	12		
Resolution	13	100 $\mu$ s	00
		10 ms	01
		1 s	02

## Commands

Name	ID	Action	ID
Control	01	Stop After Current Pulse	04
		Stop Immediately	03
Generate Pulses	02	165535	
Generate Pulses MSB	03		

## **[c]FP-PWM-520**

All ID values are hexadecimal.

## **Module ID**

010E

## Ranges

Name	ID
0100%	38

## Attributes

Name	ID	Value	ID
Period (ms)	0C	165535	
Period (ms) MSB [Most Significant Byte]	0D		

**No Commands**

## **[c]FP-QUAD-510**

All ID values are hexadecimal.

**Module ID**

0116

## Ranges

Name	ID
Boolean	10
065535	40
$\pm 160$ count/ $\mu$ s	50
$\pm 80$ count/ $\mu$ s	51
$\pm 40$ count/ $\mu$ s	52
$\pm 20$ count/ $\mu$ s	53
$\pm 10$ count/ $\mu$ s	54
$\pm 5$ count/ $\mu$ s	55
$\pm 2.5$ count/ $\mu$ s	56
$\pm 1.25$ count/ $\mu$ s	57

## Attributes

Name	ID	Value	ID
Reset Mode	14	Dont Reset on Index	00
		Reset on Index	01

## Commands

Name	ID	Action	ID
Control	01	Reset	01

## **FP-RLY-420**

All ID values are hexadecimal.

## **Module ID**

0108

## Ranges

Name	ID
Boolean	10

**No Attributes**

**No Commands**

## **cFP-RLY-421**

All ID values are hexadecimal.

## **Module ID**

0121

## Ranges

Name	ID
Boolean	10

**No Attributes**

**No Commands**

## **FP-RLY-422**

All ID values are hexadecimal.

**Module ID**

0112

## Ranges

Name	ID
Boolean	10

**No Attributes**

**No Commands**

## **cFP-RLY-423**

All ID values are hexadecimal.

## **Module ID**

0122

## Ranges

Name	ID
Boolean	10

**No Attributes**

**No Commands**

## **cFP-RLY-425**

All ID values are hexadecimal.

## **Module ID**

0125

## Ranges

Name	ID
Boolean	10

**No Attributes**

**No Commands**

## **[c]FP-RTD-122**

All ID values are hexadecimal.

## **Module ID**

010B

## Channel Status

Message	ID
Out of range	01
Open RTD	02

## Ranges

Name	ID
731123 K	26
200 to 850 °C	27
328 to 1562 °F	28
0400 Ω	30
04000 Ω	31

## Attributes

Name	ID	Value	ID
RTD Type (R0 and TCR) 04		Pt100, TCR=0.00375	00
		Pt100, TCR=0.00385	01
		Pt100, TCR=0.003911	02
		Pt100, TCR=0.003916	03
		Pt100, TCR=0.003920	04
		Pt100, TCR=0.003928	05
		Pt1000, TCR=0.00375	06
		Pt1000, TCR=0.00385	07
		Pt1000, TCR=0.003911	08
		Pt1000, TCR=0.003916	09
		Pt1000, TCR=0.003920	0A
		Pt1000, TCR=0.003928	0B

**No Commands**

## **[c]FP-RTD-124**

All ID values are hexadecimal.

**Module ID**

0118

## Channel Status

Message	ID
Out of range	01
Open RTD	02

## Ranges

Name	ID
731123 K	26
200 to 850 °C	27
328 to 1562 °F	28
0400 Ω	30

## Attributes

Name	ID	Value	ID
RTD Type (R0 and TCR) 04		Pt100, TCR=0.00375	00
		Pt100, TCR=0.00385	01
		Pt100, TCR=0.003911	02
		Pt100, TCR=0.003916	03
		Pt100, TCR=0.003920	04
		Pt100, TCR=0.003928	05

**No Commands**

## **FP-RTD-PT100-3**

All ID values are hexadecimal.

## Channel Status

Message	ID
Empty	01
Open RTD	02

## Ranges

Name	ID
50 to 350 °C	2F
58 to 622 °F	32

**No Attributes**

**No Commands**

## **FP-RTD-PT100-4**

All ID values are hexadecimal.

## Channel Status

Message	ID
Empty	01
Open RTD	02

## Ranges

Name	ID
50 to 350 °C	2F
58 to 622 °F	32

**No Attributes**

**No Commands**

## **FP-RTD-PT100**

All ID values are hexadecimal.

## Channel Status

Message	ID
Empty	01
Open RTD	02

## Ranges

Name	ID
50 to 350 °C	2F
58 to 622 °F	32

**No Attributes**

**No Commands**

## **[c]FP-SG-140**

All ID values are hexadecimal.

**Module ID**

0119

## Channel Status

Message	ID
Out of range	01
Overcurrent protection	02

## Ranges

Name	ID
$\pm 3.90625$ mV/V	64
$\pm 7.8125$ mV/V	65
$\pm 31.25$ mV/V	66
$\pm 62.5$ mV/V	67

## Attributes

Name	ID	Value	ID
Noise Rejection	01	15 Hz	09
		60 Hz	00
		240 Hz	0A
Excitation Voltage	15	10 V	00
		5 V	01
		2.5 V	02
Half-Bridge Completion	16	Half-Bridge Completion OFF	00
		Half-Bridge Completion ON	01

**No Commands**

## **FP-TB-10**

All ID values are hexadecimal.

## **Module ID**

0217

## **Ranges, Attributes, and Commands**

Refer to the module reference pages for the dual-channel modules you are using.

## **[c]FP-TC-120**

All ID values are hexadecimal.

## **Module ID**

0107

## Channel Status

Message	ID
Out of range	01
Open thermocouple	02

## Ranges

Name	ID
±50 mV	0A
±25 mV	0B
20 to 80 mV	0C
±100 mV	0D
02048 K	20
270 to 1770 °C	21
454 to 3218 °F	22

## Attributes

Attribute Name	Attribute ID	Attribute Value	Attribute ID
Thermocouple Type	02	J	00
		K	01
		T	02
		E	03
		R	04
		S	05
		N	06
		B	07
CJC Source	03	Internal	00
		0 °C	01
		25 °C	02

**No Commands**

## **FP-TC-J**

All ID values are hexadecimal.

## Channel Status

Message	ID
Empty	01
Open thermocouple	02

## Ranges

Name	ID
210 to 1200 °C	29
346 to 2191 °F	2A

**No Attributes**

**No Commands**

## **FP-TC-K**

All ID values are hexadecimal.

## Channel Status

Message	ID
Empty	01
Open thermocouple	02

## Ranges

Name	ID
100 to 1372 °C	2B
148 to 2501 °F	2C

**No Attributes**

**No Commands**

# Important Information

[Warranty](#)

[Copyright](#)

[Trademarks](#)

[Patents](#)

[Warning Regarding Use of NI Products](#)

## Warranty

The media on which you receive National Instruments software are warranted not to fail to execute programming instructions, due to defects in materials and workmanship, for a period of 90 days from date of shipment, as evidenced by receipts or other documentation. National Instruments will, at its option, repair or replace software media that do not execute programming instructions if National Instruments receives notice of such defects during the warranty period. National Instruments does not warrant that the operation of the software shall be uninterrupted or error free.

A Return Material Authorization (RMA) number must be obtained from the factory and clearly marked on the outside of the package before any equipment will be accepted for warranty work. National Instruments will pay the shipping costs of returning to the owner parts which are covered by warranty.

National Instruments believes that the information in this document is accurate. The document has been carefully reviewed for technical accuracy. In the event that technical or typographical errors exist, National Instruments reserves the right to make changes to subsequent editions of this document without prior notice to holders of this edition. The reader should consult National Instruments if errors are suspected. In no event shall National Instruments be liable for any damages arising out of or related to this document or the information contained in it.

EXCEPT AS SPECIFIED HEREIN, NATIONAL INSTRUMENTS MAKES NO WARRANTIES, EXPRESS OR IMPLIED, AND SPECIFICALLY DISCLAIMS ANY WARRANTY OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. CUSTOMER'S RIGHT TO RECOVER DAMAGES CAUSED BY FAULT OR NEGLIGENCE ON THE PART OF NATIONAL INSTRUMENTS SHALL BE LIMITED TO THE AMOUNT THERETOFORE PAID BY THE CUSTOMER. NATIONAL INSTRUMENTS WILL NOT BE LIABLE FOR DAMAGES RESULTING FROM LOSS OF DATA, PROFITS, USE OF PRODUCTS, OR INCIDENTAL OR CONSEQUENTIAL DAMAGES, EVEN IF ADVISED OF THE POSSIBILITY THEREOF. This limitation of the liability of National Instruments will apply regardless of the form of action, whether in

contract or tort, including negligence. Any action against National Instruments must be brought within one year after the cause of action accrues. National Instruments shall not be liable for any delay in performance due to causes beyond its reasonable control. The warranty provided herein does not cover damages, defects, malfunctions, or service failures caused by owner's failure to follow the National Instruments installation, operation, or maintenance instructions; owner's modification of the product; owner's abuse, misuse, or negligent acts; and power failure or surges, fire, flood, accident, actions of third parties, or other events outside reasonable control.

## **Copyright**

Under the copyright laws, this publication may not be reproduced or transmitted in any form, electronic or mechanical, including photocopying, recording, storing in an information retrieval system, or translating, in whole or in part, without the prior written consent of National Instruments Corporation.

## Trademarks

National Instruments, NI, ni.com, and LabVIEW are trademarks of National Instruments Corporation. Refer to the *Terms of Use* section on ni.com/legal for more information about [National Instruments trademarks](#).

FireWire® is the registered trademark of Apple Computer, Inc.

Handle Graphics®, MATLAB®, Real-Time Workshop®, Simulink®, and Stateflow® are registered trademarks, and TargetBox<sup>2</sup>™, xPC TargetBox<sup>2</sup>™, and Target Language Compiler™ are trademarks of The MathWorks, Inc.

Tektronix® and Tek are registered trademarks of Tektronix, Inc.

Other product and company names mentioned herein are trademarks or trade names of their respective companies.

Members of the National Instruments Alliance Partner Program are business entities independent from National Instruments and have no agency, partnership, or joint-venture relationship with National Instruments.

## Patents

For patents covering National Instruments products, refer to the appropriate location: **Help»Patents** in your software, the patents.txt file on your CD, or [ni.com/patents](http://ni.com/patents).

# **WARNING REGARDING USE OF NATIONAL INSTRUMENTS PRODUCTS**

(1) NATIONAL INSTRUMENTS PRODUCTS ARE NOT DESIGNED WITH COMPONENTS AND TESTING FOR A LEVEL OF RELIABILITY SUITABLE FOR USE IN OR IN CONNECTION WITH SURGICAL IMPLANTS OR AS CRITICAL COMPONENTS IN ANY LIFE SUPPORT SYSTEMS WHOSE FAILURE TO PERFORM CAN REASONABLY BE EXPECTED TO CAUSE SIGNIFICANT INJURY TO A HUMAN.

(2) IN ANY APPLICATION, INCLUDING THE ABOVE, RELIABILITY OF OPERATION OF THE SOFTWARE PRODUCTS CAN BE IMPAIRED BY ADVERSE FACTORS, INCLUDING BUT NOT LIMITED TO FLUCTUATIONS IN ELECTRICAL POWER SUPPLY, COMPUTER HARDWARE MALFUNCTIONS, COMPUTER OPERATING SYSTEM SOFTWARE FITNESS, FITNESS OF COMPILERS AND DEVELOPMENT SOFTWARE USED TO DEVELOP AN APPLICATION, INSTALLATION ERRORS, SOFTWARE AND HARDWARE COMPATIBILITY PROBLEMS, MALFUNCTIONS OR FAILURES OF ELECTRONIC MONITORING OR CONTROL DEVICES, TRANSIENT FAILURES OF ELECTRONIC SYSTEMS (HARDWARE AND/OR SOFTWARE), UNANTICIPATED USES OR MISUSES, OR ERRORS ON THE PART OF THE USER OR APPLICATIONS DESIGNER (ADVERSE FACTORS SUCH AS THESE ARE HEREAFTER COLLECTIVELY TERMED "SYSTEM FAILURES"). ANY APPLICATION WHERE A SYSTEM FAILURE WOULD CREATE A RISK OF HARM TO PROPERTY OR PERSONS (INCLUDING THE RISK OF BODILY INJURY AND DEATH) SHOULD NOT BE RELIANT SOLELY UPON ONE FORM OF ELECTRONIC SYSTEM DUE TO THE RISK OF SYSTEM FAILURE. TO AVOID DAMAGE, INJURY, OR DEATH, THE USER OR APPLICATION DESIGNER MUST TAKE REASONABLY PRUDENT STEPS TO PROTECT AGAINST SYSTEM FAILURES, INCLUDING BUT NOT LIMITED TO BACK-UP OR SHUT DOWN MECHANISMS. BECAUSE EACH END-USER SYSTEM IS CUSTOMIZED AND DIFFERS FROM NATIONAL INSTRUMENTS' TESTING PLATFORMS AND BECAUSE A USER OR APPLICATION DESIGNER MAY USE NATIONAL INSTRUMENTS PRODUCTS IN COMBINATION WITH

OTHER PRODUCTS IN A MANNER NOT EVALUATED OR CONTEMPLATED BY NATIONAL INSTRUMENTS, THE USER OR APPLICATION DESIGNER IS ULTIMATELY RESPONSIBLE FOR VERIFYING AND VALIDATING THE SUITABILITY OF NATIONAL INSTRUMENTS PRODUCTS WHENEVER NATIONAL INSTRUMENTS PRODUCTS ARE INCORPORATED IN A SYSTEM OR APPLICATION, INCLUDING, WITHOUT LIMITATION, THE APPROPRIATE DESIGN, PROCESS AND SAFETY LEVEL OF SUCH SYSTEM OR APPLICATION.

# Technical Support and Professional Services

Visit the following sections of the National Instruments Web site at [ni.com](http://ni.com) for technical support and professional services:

- [Support](#)—Online technical support resources at [ni.com/support](http://ni.com/support) include the following:
  - **Self-Help Resources**—For answers and solutions, visit the award-winning National Instruments Web site for software drivers and updates, a searchable [KnowledgeBase](#), [product manuals](#), step-by-step troubleshooting wizards, thousands of example programs, tutorials, application notes, instrument drivers, and so on.
  - **Free Technical Support**—All registered users receive free Basic Service, which includes access to hundreds of Applications Engineers worldwide in the [NI Developer Exchange](#) at [ni.com/exchange](http://ni.com/exchange). National Instruments Applications Engineers make sure every question receives an answer.

For information about other [technical support options](#) in your area, visit [ni.com/services](http://ni.com/services) or [contact](#) your local office at [ni.com/contact](http://ni.com/contact).

- [Training and Certification](#)—Visit [ni.com/training](http://ni.com/training) for self-paced training, eLearning virtual classrooms, interactive CDs, and Certification program information. You also can register for instructor-led, hands-on courses at locations around the world.
- [System Integration](#)—If you have time constraints, limited in-house technical resources, or other project challenges, National Instruments Alliance Partner members can help. To learn more, call your local NI office or visit [ni.com/alliance](http://ni.com/alliance).

If you searched [ni.com](http://ni.com) and could not find the answers you need, contact your [local office](#) or NI corporate headquarters. You also can visit the [Worldwide Offices](#) section of [ni.com/niglobal](http://ni.com/niglobal) to access the branch office Web sites, which provide up-to-date contact information, support phone numbers, email addresses, and current events.

## Device Properties

On the Device tab, you can rename the device and see the address and type of the device.

This tab includes the following components:

- **Device Name**—You can change the name of the device here.
- **Address**—This is the location of the device on the FieldPoint bank. The controller or network module has address 0.
- **Device Type**—Displays the type of the device.

After you make changes on properties pages, you must [deploy](#) the changes to download them to FieldPoint devices. If you do not deploy the changes, they are saved only in the project file and not on the devices.

# Channel Properties

On the Channel properties page, you can configure ranges and attributes of channels.

This page includes the following components:

- **Channels**—The ring control displays the type of the channels listed in the box below. If the device has channels of more than one type, the list box displays only the channels of the type selected in the ring control.
- **Range**—Use this ring control to configure the range for the channel or channels selected in the list box at left.
- **Power-Up Output Value**—You can use this control to configure one or more output channels to output a value at system startup.
- **Channel Attributes**—Select a channel attribute from the **Attribute** ring control, then select a value for the attribute from the **Value** ring control.
- **Compare**—Click this button to compare the current settings for the device in your project to the settings saved on the device itself.
- **OK**—Click this button to apply the changes you have made to the project settings on the host computer. The changes do not affect the settings on devices and banks until you click [Deploy](#) in the project tree view.

## Network Watchdog

The network watchdog enables you to guard your system against failures in the network connection, cables, or host computer. If the network watchdog is enabled and the network module loses communication with all hosts or clients over the network, the network module sets output channels to predefined watchdog values. Use this page to configure the network watchdog for the bank.

Select a device or output channel from the tree and use the dialog box that appears to configure watchdog settings. In order for the watchdog to work, you must enable it for the channel, the device, and the bank, and enter an output value for the channel.

- **Enable/Disable**—Enables or disables the network watchdog for the bank. If enabled for the bank, for a device, and for individual output channels on that device, the network watchdog sets those channels to watchdog output values if the network module loses communication with all hosts or clients over the network.
- **Timeout (ms)**—The amount of time that the bank waits for a response after polling the network, before it determines that there is a communication failure and applies the watchdog output values.
- **Watchdog Value**—The value that a channel outputs when the watchdog is activated.

After you make changes on properties pages, you must [deploy](#) the changes to download them to FieldPoint devices. If you do not deploy the changes, they are saved only in the project file and not on the devices.

## Add FieldPoint Serial Comm Resource

Use this dialog box to configure and add a new FieldPoint serial comm resource. A serial comm resource consists of a COM port on the host computer and a FieldPoint serial network module.

- **Name**—You can use this control to give the serial comm resource a descriptive name.
- **Port**—The COM port on the host computer to which the serial network module is or will be connected.
- **Timeout (ms)**—The amount of time the host computer waits for a response from the network module before reporting an error.
- **Baud Rate**—The rate at which the COM port communicates with serial network modules connected to it. All serial network modules connected to the same comm resource must be configured for the same baud rate.
- **OK**—Add the new serial comm resource with the properties selected above.

## Item Properties

On the Item Properties tab, you can use one or more channels to create custom data items.

This tab includes the following components:

- **Name**—You can use this control to give unique and descriptive names to items.
- **Device Name**—The name of the device that contains the item.
- **Type**—The data type of the selected channel.
- **Advise Rate (ms)**—The interval between updates to the network module.
- **Channel Bit Mask (hex)**—The channel bit mask identifies all channels used in the data item. You can copy the mask and use it to create an identical item on an identical device.

## General Properties

In the [Project Explorer](#) window, right-click the node for the RT target and select **Properties** from the shortcut menu to display this properties page.

This page includes the following components:

- **Name**—The name of the RT target.
- **IP Address/DNS Name**—The logical location or identifying name of the RT target on the network.

## Custom Scaling

You can use a custom linear scale to scale values using the equation  $y = mx + b$ , where  $x$  is a pre-scaled value and  $y$  is a scaled value.

When you select a range for a channel, the FieldPoint software automatically scales I/O data to the engineering units you select. Custom scaling enables you to apply a second level of scaling that conforms to a simple, linear slope-intercept equation. Although range selection is a form of scaling, in this discussion of custom scaling we refer to the input and output values as they appear in software as pre-scaled values.

For example, suppose you want to relate pre-scaled values to scaled values according to the following linear scale:

$$y = 2x + 0$$

where  $x$  is a pre-scaled value, and  $y$  is a scaled value.

For input operations, if the device acquires a pre-scaled value of 2, the scaled value that is read is 4. For output operations, if you write a scaled value of 2, the device generates a value of 1.

- **Enable Scaling**—Check this box to apply the custom linear scale to data on the selected channel.
- **Slope**—The slope  $m$  in the equation  $y = mx + b$ .
- **Y-Intercept**—The  $y$ -intercept  $b$  in the equation  $y = mx + b$ .
- **Equation**—Displays the linear equation for the scale.
- **Pre-Scaled Range**—The range in engineering units that the FieldPoint software is configured to apply to channel data automatically, without custom scaling.
- **Scaled Units**—The engineering units that the FieldPoint software displays for the channel in MAX.

## Bank and Network Module Properties

In the [Project Explorer](#) window, right-click the node for the FieldPoint bank and select **Properties** from the shortcut menu to display the [FieldPoint Controller Properties](#) dialog box. Select **Network Module Properties** from the **Category** list to display this page.

Use this page to configure settings for the FieldPoint RT target in the project from which you displayed this page. Use the [General](#) properties page to see the name and IP address or DNS name of the bank.

This page includes the following components:

- **Network Module Type**—You can change the type if you replace the network module on the bank with one of another type, or if you are editing configuration settings to apply them to a similar bank with a different module type.
- **Pause (ms)**—The amount of time the controller sleeps after polling the I/O modules. A higher value lowers the bank update rate but allows more time for RT applications.
- **Enable as Time Server**—A time server ensures that all FieldPoint banks on the network are synchronized. Any computer running FieldPoint software and connected to the network can be a time server. A time client synchronizes with a time server periodically. Any FieldPoint Ethernet network module can be a time client. A [c]FP-2xxx running LabVIEW RT 7.1 or later can be a time client, a time server, or both.
- **factory configuration**—If this box is checked, the bank restores the factory defaults for all device settings on the next power cycle. On FieldPoint serial banks, power-up output values for module channels remain the same even if this box is checked.

After you make changes on properties pages, you must [deploy](#) the changes to download them to FieldPoint devices. If you do not deploy the changes, they are saved only in the project file and not on the devices.

If you need to copy a configuration to another bank, open the FieldPoint configuration file in MAX, copy it using the Copy Bank Wizard, then reimport it into a project in LabVIEW.

## JavaScript Disabled

The HTML file you are trying to access uses JavaScript.

**Windows** If you are viewing the file from your computer or from a CD and you have Internet Explorer 4.0 or later installed, JavaScript is enabled by default. If you are viewing the file from a network, such as on an intranet or on the Web, or if you do not have Internet Explorer 4.0 or later installed, you must enable JavaScript to view the file. To enable JavaScript, click the **Options** button in the toolbar and select **Internet Options** from the shortcut menu.

**Macintosh** and **UNIX** Your browser does not support JavaScript, or you have disabled JavaScript. If your browser does not support JavaScript, install a recent browser version that does support JavaScript. To enable JavaScript in Internet Explorer or Netscape, select **Edit»Preferences**.

# FieldPoint Data Types

The **type** terminal on FP Read and FP Advise can be wired with any of four values: Float (DBL), Float Array, Boolean, and Boolean Array. The **Value In** and **Value Out** terminals on the FieldPoint Express block and the **values** terminal on FP Write and FP Write/Read take the same data types.



**Float (DBL)** works with single-channel analog and single-channel counter I/O. Float (DBL) also works with multichannel digital I/O, and it is especially useful if your application is designed to respond to specific combinations of digital channel data. If you are a new LabVIEW user, or if each channel affects a different part of your application, the Boolean Array data type is more suitable for your multichannel digital I/O.



**Float Array** works with multichannel analog and counter I/O. Float Array is the default data type for all FieldPoint VIs.



**Boolean** works with single-channel digital I/O.



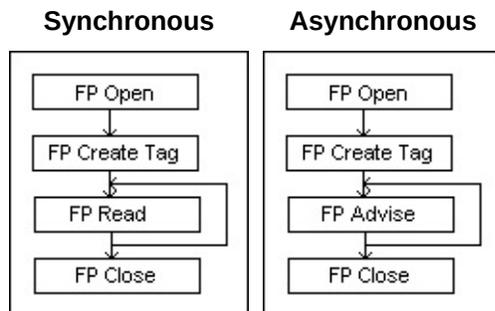
**Boolean Array** works with multichannel digital I/O. Multichannel digital data is returned as a 16-element array regardless of the number of channels included in the FieldPoint item. Array elements corresponding to channels that do not exist or that are not included in the FieldPoint item are always FALSE.

Drop a FieldPoint VI onto the block diagram or front panel, then do one of the following:

- Wire a LabVIEW control of the correct data type to the **type** or **values** terminal.
- Right-click the FieldPoint VI, click **Select Type**, and select the type from the drop-down list.

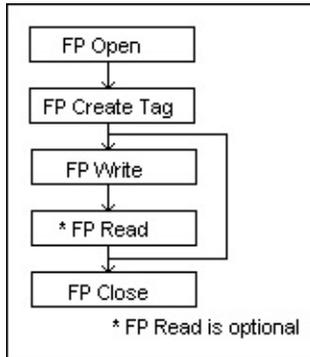
# Input Item

The following figures illustrate the program flow for input items.



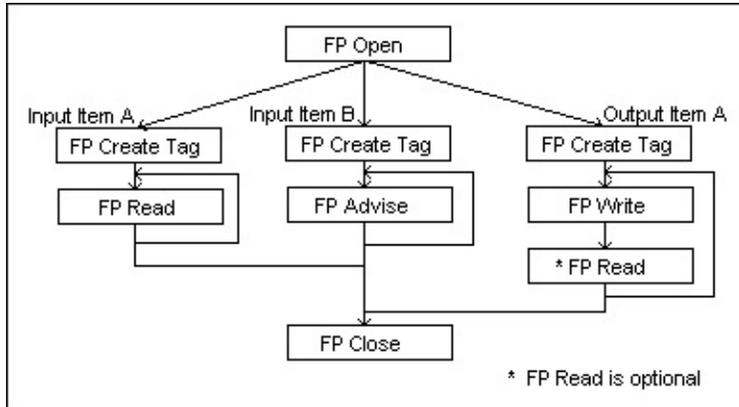
## Output Item

The following figure illustrates the program flow for an output item. FP Write (Polymorphic) returns an error code if a channel cannot be changed, but does not return an error for certain channel specific errors (such as Open Current Loop on a [c]FP-AO-200).



# Multiple Items

The following figure illustrates one possible program flow for multiple items.



## Branch Offices

Office	Telephone Number
Australia	1800 300 800
Austria	43 0 662 45 79 90 0
Belgium	32 0 2 757 00 20
Brazil	55 11 3262 3599
Canada	800 433 3488
China	86 21 6555 7838
Czech Republic	420 224 235 774
Denmark	45 45 76 26 00
Finland	385 0 9 725 725 11
France	33 0 1 48 14 24 24
Germany	49 0 89 741 31 30
India	91 80 51190000
Israel	972 0 3 6393737
Italy	39 02 413091
Japan	81 3 5472 2970
Korea	82 02 3451 3400
Lebanon	961 0 1 33 28 28
Malaysia	1800 887710
Mexico	01 800 010 0793
Netherlands	31 0 348 433 466
New Zealand	0800 553 322
Norway	47 0 66 90 76 60
Poland	48 22 3390150
Portugal	351 210 311 210
Russia	7 095 783 68 51
Singapore	1800 226 5886
Slovenia	386 3 425 4200
South Africa	27 0 11 805 8197
Spain	34 91 640 0085
Sweden	46 0 8 587 895 00
Switzerland	41 56 200 51 51
Taiwan	886 02 2377 2222
Thailand	662 278 6777
United Kingdom	44 0 1635 523545
United States (Corporate)	512 683 0100

## FieldPoint Controller Properties

In the [Project Explorer](#) window, right-click the FieldPoint RT controller icon and select **Properties** from the shortcut menu to display this dialog box.

Use the **Category** list at the left side of the dialog box to set the following options:

- [General](#)—Sets the name and IP address for the FieldPoint bank.
- [Bank and Network Module Properties](#)—Configures network module behavior.
- [VI Server: Configuration](#)—Configures the VI Server for the RT controller.
- [VI Server: Machine Access](#)—Controls machine access to VIs through the VI Server for the RT controller.
- [VI Server: User Access](#)—Controls user access to VIs through the VI Server for the RT controller.
- [VI Server: Exported VIs](#)—Identifies which VIs other applications can access through the VI Server for the RT controller.
- [Conditional Disable Symbols](#)—Defines symbols to use with the Conditional Disable structure in any VI in the RT controller.