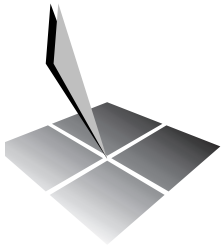


FieldPoint™

# FP-3000 Network Module User Manual

NATIONAL INSTRUMENTS™

July 2000 Edition
Part Number 322169B-01

**Worldwide Technical Support and Product Information**

www.ni.com

**National Instruments Corporate Headquarters**

11500 North Mopac Expressway Austin, Texas 78759-3504 USA Tel: 512 794 0100

**Worldwide Offices**

Australia 03 9879 5166, Austria 0662 45 79 90 0, Belgium 02 757 00 20, Brazil 011 284 5011,
Canada (Calgary) 403 274 9391, Canada (Ontario) 905 785 0085, Canada (Québec) 514 694 8521,
China 0755 3904939, Denmark 45 76 26 00, Finland 09 725 725 11, France 01 48 14 24 24,
Germany 089 741 31 30, Greece 30 1 42 96 427, Hong Kong 2645 3186, India 91805275406,
Israel 03 6120092, Italy 02 413091, Japan 03 5472 2970, Korea 02 596 7456, Mexico (D.F.) 5 280 7625,
Mexico (Monterrey) 8 357 7695, Netherlands 0348 433466, New Zealand 09 914 0488, Norway 32 27 73 00,
Poland 0 22 528 94 06, Portugal 351 1 726 9011, Singapore 2265886, Spain 91 640 0085,
Sweden 08 587 895 00, Switzerland 056 200 51 51, Taiwan 02 2528 7227, United Kingdom 01635 523545

For further support information, see the *Technical Support Resources* appendix. To comment on the documentation, send e-mail to techpubs@ni.com

# Important Information

## Warranty

The FieldPoint FP-3000 network module is warranted against defects in materials and workmanship for a period of one year from the date of shipment, as evidenced by receipts or other documentation. National Instruments will, at its option, repair or replace equipment that proves to be defective during the warranty period. This warranty includes parts and labor.

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



## Trademarks



## 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.

# Conventions

The following conventions are used in this manual:

| | |
|---|---|
| » | 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 tip, which alerts you to advisory information. |
|  | 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 on in the software, such as menu items and dialog box options. Bold text also denotes parameter names. |
| Fieldbus | The generic term *Fieldbus* refers to any bus that connects to field devices. This includes FOUNDATION Fieldbus, CAN, DNET, and Profibus. In this manual, the term *Fieldbus* refers specifically to the FOUNDATION Fieldbus. |
| *italic* | Italic text denotes variables, emphasis, a cross reference, or an introduction to a key concept. |
| `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, and code excerpts. |

# Contents

## Chapter 1
## FP-3000 Network Module Overview



## Chapter 2
## Installation and Configuration

## Chapter 3
## Example Applications

# Chapter 4
# Block Reference

# Appendix A
# Configuring the FP-3000



# Appendix B
# Fieldbus Parameters

# Appendix C
# FP-3000 Specific Parameters

## Appendix D
## Advanced Function Block Behavior



## Appendix E
## Specifications

## Appendix F
## Troubleshooting

# 1

# FP-3000 Network Module Overview

## Related Documentation

The following documents contain information that you might find helpful as you read this manual:

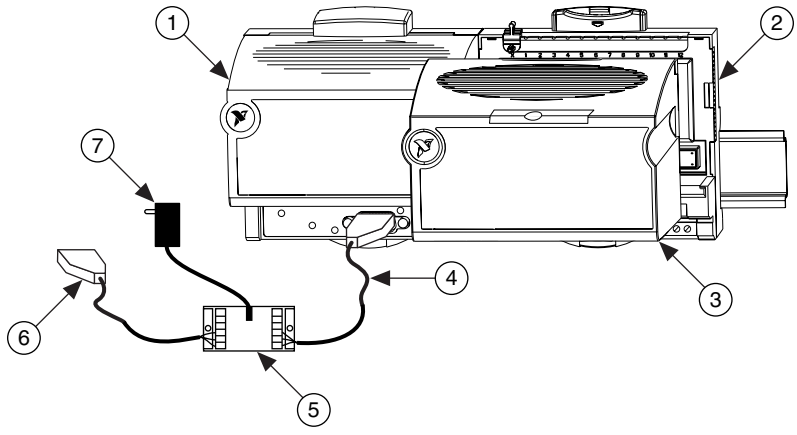
- Operating Instructions (for network module, terminal bases, and I/O modules)
- Fieldbus Foundation's *Wiring and Installation 31.25 kbit/s, Voltage Mode, Wire Medium Application Guide*
- *Fieldbus Standard for Use in Industrial Control Systems, Part 2*, ISA-550.01.1992

## FP-3000 Network Module Description

The FP-3000 is an intelligent network interface and controller module that manages a bank of up to nine FieldPoint I/O modules and terminal bases. The FP-3000 network module and the terminal bases snap together to form a high-speed data bus for communications between the FP-3000 network module and any I/O modules in the bank. The FP-3000 includes an H1 Fieldbus interface for direct connection to an H1 FOUNDATION Fieldbus link (also known as a segment). Refer to the *FOUNDATION Fieldbus Overview* document for more information on FOUNDATION Fieldbus. Under the best circumstances, you can connect at most 32 FP-3000 network modules to a single FOUNDATION Fieldbus link (without repeaters). The actual number of FP-3000 network modules that can be connected depends on your network setup and your applications.

The FP-3000 network module allows you to bring conventional analog and discrete I/O devices onto a FOUNDATION Fieldbus network. For example, the FP-3000 makes a 4–20 mA pressure transmitter connected to a FieldPoint 8-channel analog input module behave like a Fieldbus pressure transmitter. By using an FP-3000 network module, you can significantly reduce wiring and installation costs. Instead of running a pair of wires from

each 4–20 mA device to your controller, you can mount an FP-3000 network module in the field and run a single pair of wires (called the trunk) from your PC to the FP-3000. You connect the 4–20 mA devices to the FieldPoint I/O modules by short stretches of wire. The following figure shows an FP-3000 connected to a Fieldbus network.




| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 1 | FP-3000 Network Module | 2 | Terminal Base | 4 | Fieldbus Trunk | 6 | To Fieldbus |
| | | 3 | I/O Module | 5 | Relcom power hub | 7 | Power supply |

**Note** You can also create your own terminators and connectors as described in the *FOUNDATION Fieldbus Overview* document.

The FP-3000 is National Instruments' primary field device for FOUNDATION Fieldbus. It has an onboard processor that allows it to execute function blocks. The FP-3000 implements FOUNDATION Fieldbus-compliant Resource, AI, AO, DI, DO, and PID blocks. It also supports additional National Instruments defined blocks, LOG, STAT, and EXPR.

# Features of the FP-3000 Network Module

## Function Blocks

Conventional devices connected to I/O modules are made visible as Fieldbus function blocks. Function blocks are software modules which describe the fundamental elements of an I/O or control system. The FP-3000, like any FOUNDATION Fieldbus–compliant device, has one or more function blocks. The function blocks in different devices can be connected

to form a distributed control system. Function blocks can perform PID calculations, logic operations, read an input channel, write an output channel, or a number of other functions.

The FP-3000 implements FOUNDATION Fieldbus–compliant I/O function blocks, such as Analog Input (AI), Analog Output (AO), Discrete Input (DI), and Discrete Output (DO). These blocks provide functionality such as scaling, trending, and alarming. For example, when you connect a 4–20 mA pressure transmitter to a FieldPoint I/O, you can configure an FP-3000 Analog Input function block to convert from 4–20 mA to engineering units. You can set up alarm limits so that the FP-3000 sends an alarm when the pressure exceeds the limits. The FP-3000 network module can also collect trend samples and broadcast them to applications on a PC. For more information on function blocks, refer to Chapter 4, *Block Reference*.

## PID Control

The FP-3000 implements a FOUNDATION Fieldbus–compliant PID control function block. This PID can be used to control either an analog output element connected to FieldPoint I/O or a native Fieldbus device, such as a valve, connected to the Fieldbus network. The FP-3000 executes the PID and other function blocks deterministically in accordance with the schedule configured by the NI-FBUS Configurator and/or the user. For more information on PID control, refer to Chapter 4, *Block Reference*.

## Block Instantiation and Deletion

You can instantiate (create copies of) or delete the PID function block on an as-needed basis. For example, if you are adding a new loop to an existing Fieldbus network, you can instantiate a PID function block in the FP-3000 to control the loop. You can also instantiate the I/O function blocks on an as-needed basis. If you have an 8-channel Analog Input module and you are using only two channels, you can save memory by instantiating only two AI function blocks. You can instantiate additional AI function blocks when you use additional channels. National Instruments recommends instantiating additional AI function blocks if you will be approaching the 150 block limit. For instruction, refer to the section *Create an FP-AI-110 Block* in Chapter 3, *Example Applications*.

## Interoperability

The FP-3000 network module can send or receive data from any FOUNDATION Fieldbus-compliant device. The PID block in the FP-3000 can get its input from any FOUNDATION Fieldbus-compliant device; it can also control any FOUNDATION Fieldbus-compliant output device.

The control and I/O functionality of the FP-3000 can be configured by any Fieldbus configurator that implements FOUNDATION Fieldbus instantiation and deletion and device description parsing. National Instruments NI-FBUS Configurator is such a configurator. This is possible because all of the features added by National Instruments are described using Device Descriptions. Any Fieldbus-compliant HMI package or OPC server that supports instantiation and deletion of function blocks and that can parse device description files can also access the FP-3000 function blocks.

## Scheduling Functionality

Fieldbus networks require a Link Active Scheduler (LAS) to control communications on the Fieldbus. The FP-3000 can act as a primary or back-up Link Active Scheduler. If the primary LAS (often an interface board in a PC) fails or is disconnected from the network, the FP-3000 takes over the bus and executes the schedule without causing a bump.

## HotPnP (Hot Plug and Play)

FP-3000 network modules can be added or removed from H1 Fieldbus networks without affecting other Fieldbus devices. The HotPnP feature simplifies system installation, configuration, and maintenance. With the HotPnP feature, you can remove or insert I/O modules into the FieldPoint terminal bases while power is on, even if the system is already engaged in network activity. You do not need to power down the FP-3000, Fieldbus network, or even a bank to insert, remove, or replace I/O modules. In addition, you do not need to change the operation of the host computer or software to use the HotPnP feature. You do not need to restart the host computer software to use the HotPnP feature. You can replace an I/O module only with another I/O module of the same type.

While one or more new or replacement I/O modules in a bank are being serviced by the HotPnP feature, the other I/O modules in the bank remain fully operational and accessible on the network without any interruptions. As soon as the FP-3000 configures the new I/O module through the HotPnP service, that I/O module becomes automatically accessible on the network.

**Caution** To avoid damaging the network module and the terminal bases, make sure that you do not add or remove *terminal bases* while power is applied to the bank. An I/O module can be hot-inserted only if an empty terminal base is already available in the bank.

# Using Third-Party Configuration Software with the FP-3000

If you choose to use third-party configuration software with the FP-3000, check that it supports the advanced features required by the FP-3000.

## Potential Problems with Third-Party Configuration Software

Because FieldPoint I/O modules are interchangable, it does not make sense to predefine the available function blocks. An FP-3000 initially has only a resource block. All other function blocks must be instantiated by the configuration software. Instantiation and deletion of function blocks is an advanced part of the FOUNDATION Fieldbus specification and not all configuration software packages offer it. If you want to use an FP-3000 with a third-party configuration software package, verify that the package supports the instantiation and deletion of function blocks.

## Parsing of Device Descriptions by Third-Party Configuration Software

Many of the features of the FP-3000 require that the configuration software be able to parse device description files. Verify that the third-party configuration software can read the standard `.ffo` and `.sym` device description files.

# 2

# Installation and Configuration

## Import Device Descriptions

The device description files contain information about the types of blocks and parameters supported by the FP-3000, along with online help describing the uses of given parameters. Before you can use the FP-3000 with NI-FBUS (or other host software), you must import the device description file on the host computer(s). To install the FP-3000 device description, complete the following steps:

**Note** This process is for use with National Instruments NI-FBUS. The process can vary with other host software packages.

1. Install and configure your NI-FBUS Fieldbus interface and software, if you have not done so already. For help, refer to the getting started manual that came with your interface.
2. Insert the device description disk or CD (shipped with the FP-3000) into the disk drive of the host computer.
3. Select **Start»Programs»National Instruments FBUS»Interface Config** to run the Interface Configuration utility.
4. Click on the **DD Info** button. The **DD Info** dialog box appears.
5. If the base directory field is blank, enter a base directory. The base directory you enter here will be where NI-FBUS looks for all device descriptions. Do not change the base directory after you have started importing device descriptions; otherwise, NI-FBUS will not be able to find the device descriptions you previously imported. Your device description files will automatically be placed in the appropriate manufacturer ID subdirectory under this base directory.

   Your base directory will include one folder for each different manufacturer for which you have imported device description. For example, if you import the device description for the National Instruments FP-3000 device, you will find a folder called `4e4943`. This is the National Instruments FOUNDATION Fieldbus device manufacturer ID number.

The next layer of folders is the device type. For example, the FP-3000 has a device type ID number of 4005.

Underneath this layer of directories you will find the individual device description files (`.ffo` and `.sym`)

6. If necessary, click on the **Browse** button to select the standard text dictionary, provided with NI-FBUS. The text dictionary has a `.dct` extension.
7. Click on the **Import DD** button. The **Import DD** dialog box appears.
8. Click on the **Browse** button, browse to the `.ffo` device description file path, and click on **Open**. The device description for your FP-3000 is supplied on the disk. For each device, there are two device description files, one that ends in `.ffo` and one that ends in `.sym`. Select the `.ffo` file, and the corresponding `.sym` file will be imported automatically. The file name will be in the form *`Digit Digit Digit Digit`*`.ffo` (for example, `0101.ffo`).

**Note** If you are importing device descriptions for multiple devices, you might see that they can have the same filenames. Each file contains information about the device and its manufacturer, and will be placed appropriately in the hierarchy under the base directory.

9. Click on **OK**. A window will appear that gives the full path to which the `.ffo` and `.sym` files were copied.
10. Click on **OK**.

You only need to install the device description file one time for a given version of the firmware. You do not have to repeat the device description installation for each FP-3000 connected to your computer. The computer uses this device description for all FP-3000 network modules on the bus.

For more information on device descriptions, refer to the section *Device Descriptions* in the *FOUNDATION Fieldbus Overview* document.

## Updating the Device Description

Any enhancement to the FP-3000 functionality, such as the addition of new function blocks or support of new types of I/O modules, results in a new Device Description file describing the features of the FP-3000. You must update the FP-3000 firmware and install the new Device Description files to take advantage of the new features. Refer to the section *Updating the FP-3000 Firmware* for instructions on downloading new firmware and installing the new Device Description file.

# Mount the FP-3000 and Terminal Bases

You can mount your FieldPoint system either to a standard 35 mm DIN rail or directly on a panel. Panel mounting is generally the more secure option, but DIN rail mounting might be more convenient for your application. The following sections give instructions for both mounting methods.

## Mounting the FP-3000 on a DIN Rail

The FP-3000 has a simple rail clip for reliable mounting onto a standard 35 mm DIN rail. Follow these steps to mount the FP-3000 on a DIN rail.

1. Use a flat-bladed screwdriver to open the DIN rail clip to the unlocked position, as shown in Figure 2-1.

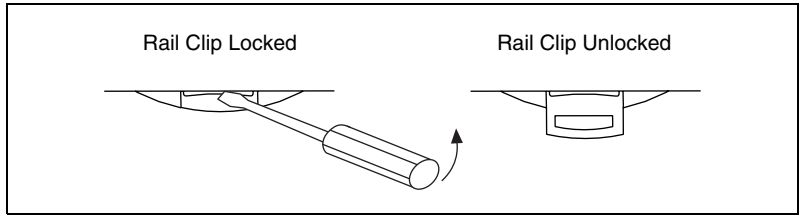



**Figure 2-1.** DIN Rail Clip

2. Hook the lip on the rear of the FP-3000 onto the top of a 35 mm DIN rail and press the FP-3000 down onto the DIN rail, as shown in Figure 2-2.

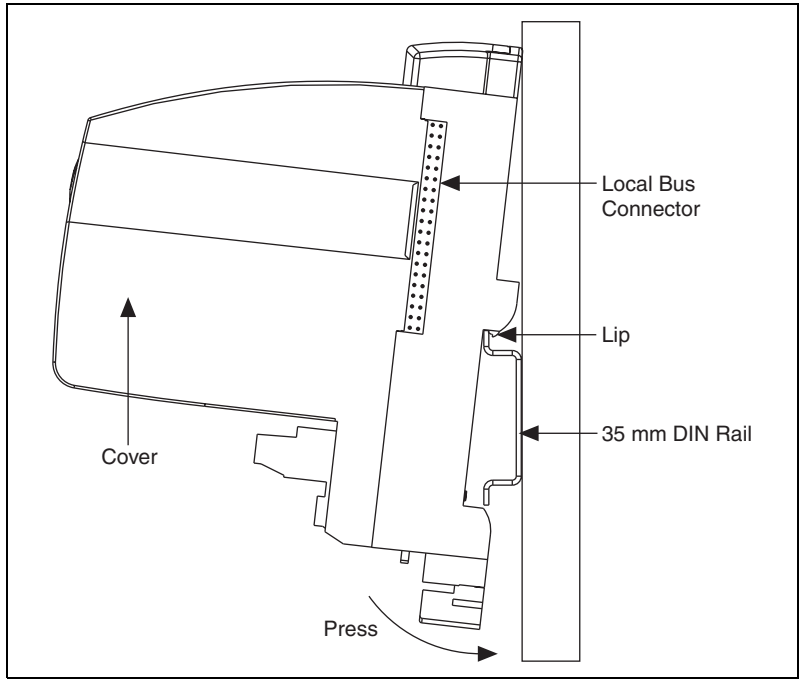



**Figure 2-2.** Mounting the FP-3000 onto a DIN Rail

3. Slide the FP-3000 to the desired position along the DIN rail. After the FP-3000 is in position, lock it to the DIN rail by pushing the rail clip to the locked position, as shown in Figure 2-1.

After the FP-3000 is mounted to the DIN rail, connect the terminal base to the FP-3000 as explained in the next section, *Connecting Terminal Bases with DIN Rail Mounting*.

## Connecting Terminal Bases with DIN Rail Mounting

Follow these steps to connect a terminal base to an FP-3000 network module using DIN rail mounting.

**Caution** To avoid damaging the FP-3000 and the terminal bases, make sure that power is not applied to the FP-3000 while you install or remove terminal bases.

1. Mount the terminal base onto the DIN rail in the same way you installed the FP-3000.
2. Attach the terminal base to the FP-3000 by firmly mating the local bus connectors.

3. To add more terminal bases, install them on the rail and connect their local bus connectors together. A single FP-3000 can support up to nine terminal bases.
4. Place the protective cover (from the bag of accessories that came with your FP-3000) onto the local bus connector of the last terminal base on the bank, as shown in Figure 2-3.

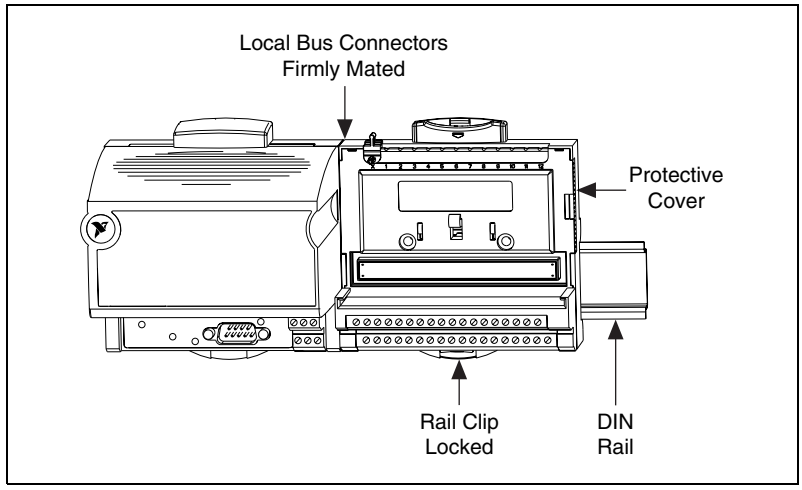



**Figure 2-3.** Connecting Terminal Bases

## Removing the FP-3000 from the DIN Rail

To remove an FP-3000 network module, unlock it from the DIN rail by placing a screwdriver in the slot on the rail clip and opening it to the unlocked position, as shown in Figure 2-1. Then, disconnect the FP-3000 from the local bus connector of the terminal base, and lift the FP-3000 off the rail. You should also remove the module from the terminal base to the right since it has a clip that snaps into the base to its left.

**Caution** To avoid damaging the FP-3000 and the terminal bases, make sure that power is not applied to the FP-3000 while you install or remove terminal bases.

# Mounting the FP-3000 to a Panel

Follow these steps to install the optional FieldPoint network panel mount accessory and mount the FP-3000 network module to a panel. You can order the panel mount accessory, part number 777609-01, from National Instruments.

1. Use a flat-bladed screwdriver to open the rail clip to the unlocked position, as shown in Figure 2-1.
2. Snap the panel mount accessory onto the module, as shown in Figure 2-4.

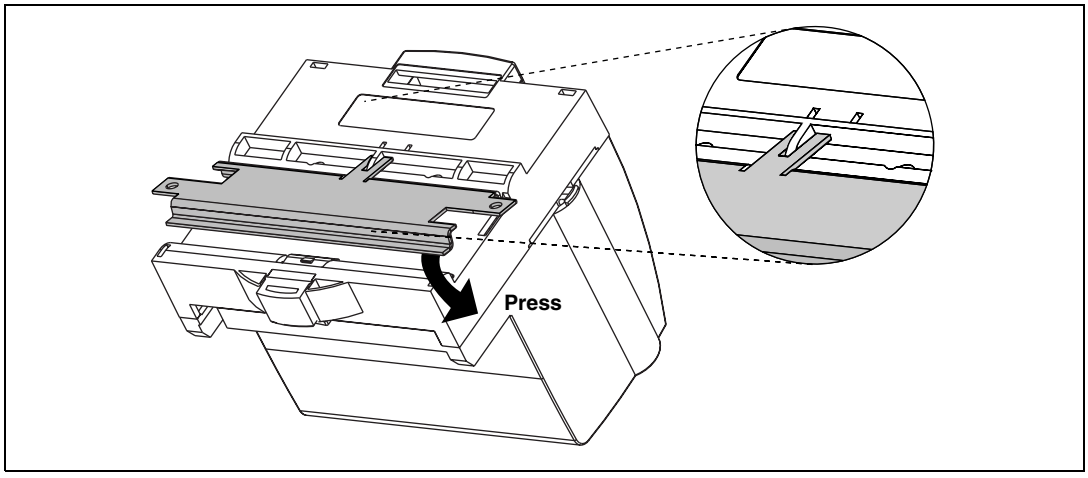



**Figure 2-4.** Installing the Network Panel Mount Accessory

3. Lock the panel mount accessory into place by pushing the rail clip to the locked position, as shown in Figure 2-1.
4. Mount the FP-3000 to your panel with the panel mount accessory. The installation guide that came with the panel mount accessory includes a guide that you can use to drill pilot holes for mounting the FP-3000.

## Connecting Terminal Bases with Panel Mounting

You can install terminal bases directly, without using the panel mount accessory needed to mount the FP-3000 network module. Follow these steps to connect terminal bases to a network module using panel mounting.

**Caution** To avoid damaging the FP-3000 and the terminal bases, make sure that power is not applied to the FP-3000 while you install or remove terminal bases.

1. Drill pilot holes in the panel to mount the terminal bases. A drilling guide is provided with the network module panel mount accessory.
2. Attach the terminal base to the FP-3000 by firmly mating the local bus connectors.

3. Bolt, screw, or otherwise fasten the terminal base to the panel. Make sure that the local bus connectors remain firmly mated after the terminal base is mounted.
4. To add more terminal bases, repeat Steps 1 through 3, mating the local bus connectors of each new terminal base to the connector of the last installed base. If all the pilot holes were correctly drilled, the local bus connectors should remain firmly mated after all the bases are mounted to the panel.
5. Place the protective cover (from the bag of accessories that came with your FP-3000) onto the local bus connector of the last terminal base on the bank.

### Removing the FP-3000 and Terminal Bases from the Panel

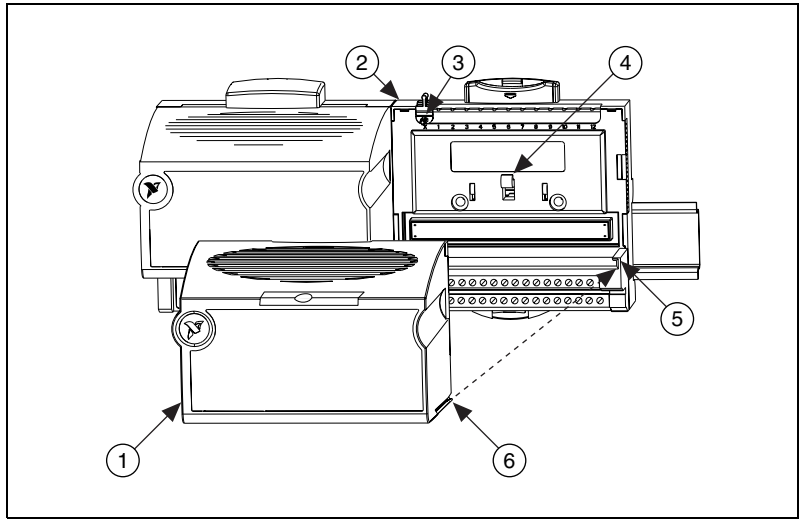
To remove an FP-3000 network module and terminal bases from the panel, reverse the process described in the previous sections, *Mounting the FP-3000 to a Panel* and *Connecting Terminal Bases with Panel Mounting*. First remove the terminal bases, starting with the last one, then remove the network module.

**Caution** To avoid damaging the FP-3000 and the terminal bases, make sure that power is not applied to the FP-3000 while you install or remove terminal bases.

## Mount I/O Modules onto Terminal Bases

Follow these steps to connect an I/O module to a terminal base:

1. Optional: Before mounting the module, you can set the key to prevent operators from inadvertently replacing a module with another module of a different type. It its default position, the key allows any type of module to be placed on that terminal base. Adjust the key to the appropriate slot for the type of module, if desired. For more information, refer to the operating instructions that came with your module.
2. Position the first module with its alignment slots aligned with the guide rails on the terminal base, as shown in Figure 2-5.
3. Firmly press the module onto the terminal base. The terminal base latch locks the I/O module into place.
4. Repeat this procedure to install additional I/O modules onto terminal bases.




| | | | | | |
|---|---|---|---|---|---|
| 1 | I/O Module | 3 | Key | 5 | Guide Rail |
| 2 | Terminal Base | 4 | Latch | 6 | Alignment Slot |

**Figure 2-5.** Mounting I/O Module to Terminal Base

## Removing I/O Modules

To remove an I/O module, push down on the ejector button at the top of the terminal base. If the button sticks, you can insert a 1/4" flat-bladed screwdriver behind the ejector button and twist. This motion unlatches the I/O module, which can then be lifted off of the terminal base.

## Inserting New I/O Modules During Operation

When a new I/O module is inserted, the FP-3000 automatically configures the I/O module to factory default settings. This configuration is accomplished without any intervention from the host computer or software.

## Replacing I/O Modules During Operation

The host computer can detect missing I/O modules through the block alarm on the associated function blocks.

When a new I/O module is connected in place of one that was removed, the FP-3000 first verifies that the replacement I/O module is compatible with the one that was removed. If the I/O module is either the same as or compatible with the one removed, the FP-3000 configures the replacement I/O module with its predecessor's configuration and output value settings.

Hot-swap a module only with a compatible module. If you hot-swap a module with an incompatible module, the associated function blocks must be entirely reconfigured.

# Connect the FP-3000 to the Fieldbus Network

The FP-3000 can be one of up to 32 devices connected to a Fieldbus network. The connection is made through the 9-pin male Dsub Fieldbus connector on the FP-3000, shown in Figure 2-6.

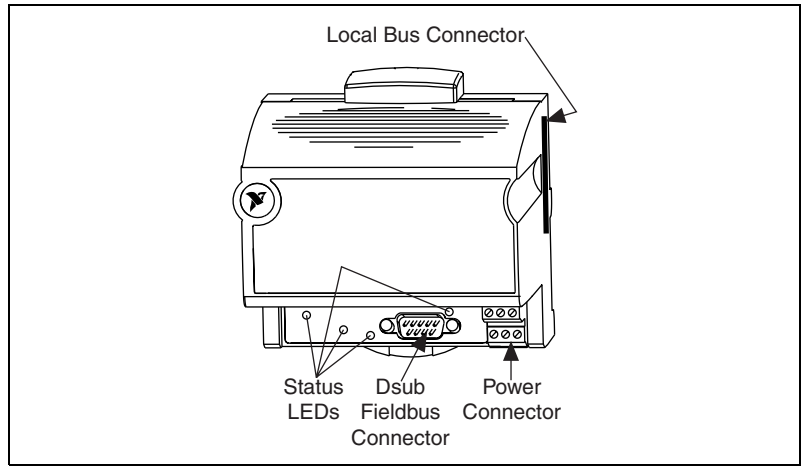



**Figure 2-6.** Fieldbus Connectors on the FP-3000

Use a Fieldbus cable with a 9-pin female Dsub connector to connect the FP-3000 to a properly terminated Fieldbus network. When you are only using an FP-3000, the power hub is not being used for power—it is only being used because it has the terminators inside. For other FOUNDATION Fieldbus devices that use bus powering, you would apply power to the hub, from which devices would get their power. Refer to the Fieldbus Foundation *Wiring and Installation 31.25 kbit/s, Voltage Mode, Wire Medium Application Guide* for specific information about wiring and installing a Fieldbus network. If you want to make your own Fieldbus cable, refer to the *Fieldbus Standard for Use in Industrial Control Systems, Part 2*, ISA-S50.02.1992. The FP-3000 Fieldbus connector pinout is shown in Figure 2-7.

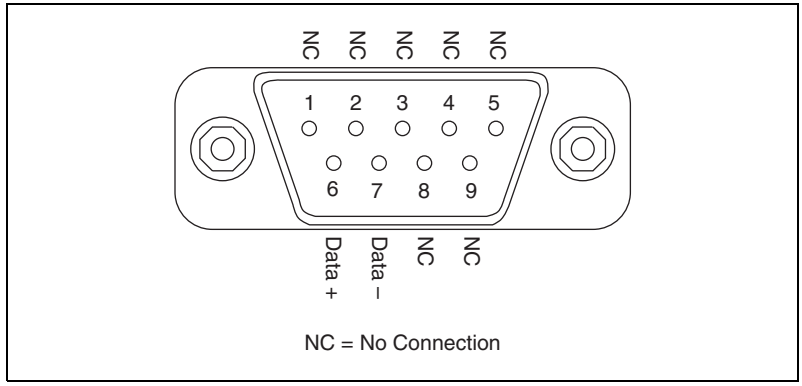



**Figure 2-7.** FP-3000 Connector Pinout

# Connect Power to the FP-3000

Insufficient powering is one of the main causes for problems observed with the FP-3000. Calculate how much power your bank will require and provide power accordingly.

## Bank Power Requirements

As with any FieldPoint bank, the power requirements depend on the network module, the I/O modules, and any devices being powered by the analog or digital outputs.

The power supply connected to the network module provides power to operate the entire bank of I/O modules via the backplane in the terminal bases. It does NOT provide power for the output signals (such as an analog outputs) unless you add external circuitry to make this the case.

## I/O Power Requirements

For an *output* module, the Vsupply and Common terminals of the terminal base must be wired and powered. Unless you add external circuitry, these terminals are not electrically connected to the Vsupply and Common of the backplane. They are used for power and referencing of the I/O signals only (not the I/O module itself). For example, they are used with analog output, digital output, pulse width modulator, or counter modules to provide power for the outputs.

The Vsupply and Common of *input* modules do not need to be wired. If the inputs you are measuring should have a common ground, you could wire the Common terminal of the module to the Common of the incoming signals.

## Supplying Power for Outputs

There are three ways to wire power for the outputs:

- Power the FP-3000 and each output module's output circuitry with a separate power supply. Do not cascade any of the Vsupply and Common terminals. This option provides the most isolation.
- Power all the output circuitry by connecting a second power supply, leaving the network module on its own power supply, and cascading power from one set of connections to the next. This establishes two independent grounds—one for the FP-3000. The backplane through the terminal bases and the I/O modules themselves allows for some isolation without using a dedicated supply for each output module.
- Power the outputs with the same supply as the FP-3000 and FieldPoint modules by cascading the Vsupply and Common from the network module to the first terminal base and then cascading power from one set of connectors to the next. This allows outputs to be powered by the same supply (assuming the supply can provide enough power to meet the demands of the modules AND any output currents). The major drawback of this scheme is that a single ground will be established for both the backplane and the input/output circuitry.

**Caution** This method defeats the isolation of the modules.

## Isolation

The Isolation rating is the maximum voltage differential that can occur between the common terminal on the terminal base (the input circuit's ground level) and the ground in the backplane of the module (the network module's ground level) without causing damage to the circuitry.

Safety Isolation (or working voltage) is the maximum voltage differential (per safety isolation specifications) that can be sustained between the common terminal on the terminal base (the input circuit's ground level) and the ground in the backplane of the module (the network module's ground level) while still allowing accurate measurements and safe working conditions for human operators.

## Calculating Power for a FieldPoint Bank

The power requirements for a FieldPoint bank that uses an FP-3000 network module are calculated as follows:

$$Power = 6 \text{ watts} + 1.15 * \sum(I/O \text{ Module Consumption})$$

This is the amount of power the network module consumes from the power supply to power itself and the I/O modules. It does not include any power consumed by devices that you wire to the terminal bases.

The power requirement for the FP-3000 network module alone (no connected I/O modules) is 6 Watts. The power requirement of each I/O module is listed in the catalog and in the Operating Instructions pamphlet for that module. Add the I/O modules' requirements and multiply by 1.15 (to account for the power requirements of the terminal bases). If you are using a separate power supply for outputs, the third term in the calculation for the primary power supply's requirements will be zero.

## Power Connections

**Note** You need to connect the FP-3000 to the network before applying power.

An 11–30 VDC power supply is required by each FP-3000 on your network. The FP-3000 filters and regulates this supplied power and provides power for all the I/O modules in the bank. Therefore, you do not need to provide power separately to each FieldPoint I/O module in the bank. As discussed in the section *Supplying Power for Outputs*, you will need to select a method for powering any output circuitry on output modules.

The power connector is a 6-pin screw terminal power connector whose pinout is shown in Figure 2-8. See Figure 2-6 for the location of the power connector.

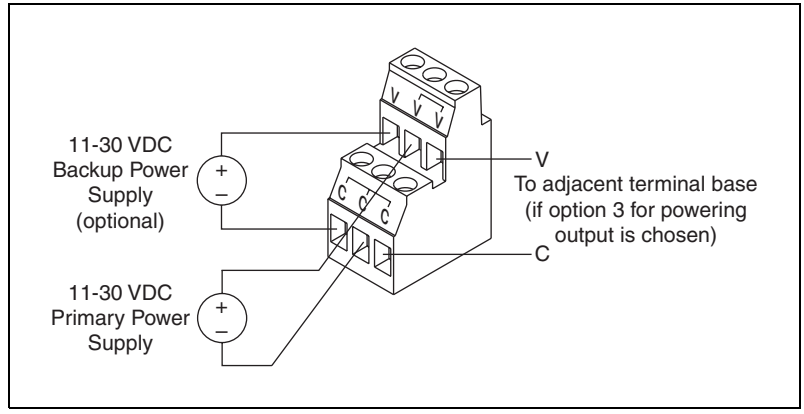



**Figure 2-8.** FP-3000 Power Connector Pinout

Connect the primary power supply to the center V and C pair with the positive and negative wires on your power cable in the V and C terminals, respectively. You can connect an optional backup power supply to the left V and C pair.

**Note** The FP-3000 will automatically use the power supply with the highest voltage. Do *not* use a battery backup with a higher voltage than the primary supply. In this case, the device will run off of the battery until the battery's voltage level drops below that of the primary power supply.

The right V and C pair provides a convenient means of connecting power to the V and C terminals of a terminal base. Figure 2-8 shows this optional connection.

If your field I/O devices need to be powered separately, you can use the terminals provided on each terminal base for such power supply connections. Refer to the documentation that came with your terminal base and I/O module for more information on powering your field I/O devices.

# LED Indicators

## Power-On Self Test (POST)

The power-on self test (POST) is a test suite that the FP-3000 performs at power up to verify its own operational status. The test takes several seconds. The test is non-invasive and therefore does not affect the operation of the network, nor does it affect any of your field wiring connected to the terminal bases in the bank.

If the self-test suite fails, the FP-3000 does not participate in the network communication traffic, eliminating potential conflicts with the other banks in your network.

The FP-3000 has four LED indicators: **POWER**, **NETWORK**, **PROCESS**, and **STATUS**. Figure 2-9 shows the LEDs on the FP-3000.

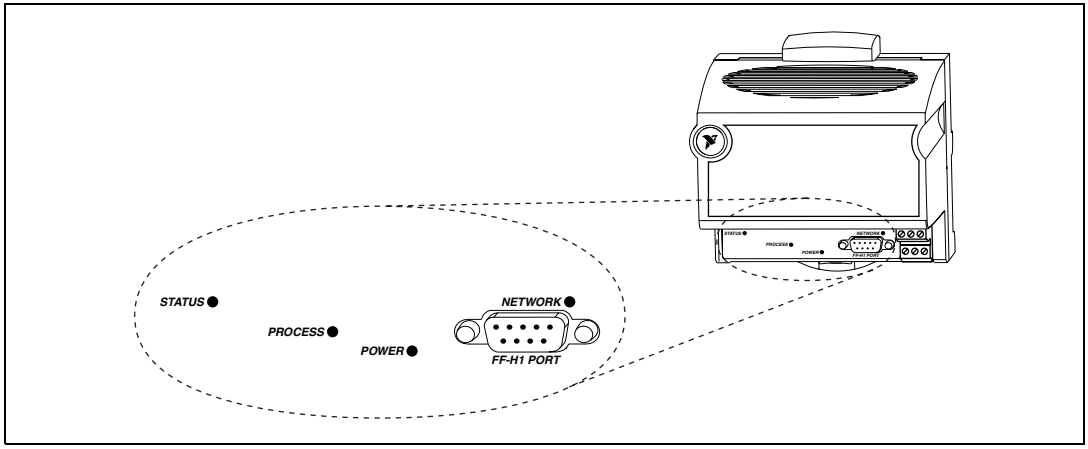



**Figure 2-9.** LEDs on the FP-3000

The FP-3000 indicates power-on self test failure through the **POWER** and **STATUS** LEDs.

When power is applied, the **POWER** LED blinks green for approximately seven seconds during the power on self test. If the self test passes, the **POWER** LED turns solid green and the **READY** LEDs on each I/O module are lit green. If the self test fails, the **POWER** LED is lit red and the module enters an inactive state.

The red **STATUS** LED is lit when non-volatile memory into memory buffer fails due to a checksum error. If **STATUS** is not lit, the FP-3000 has not detected a failure. The FP-3000 indicates specific error conditions by flashing **STATUS** a specific number of times. Table 2-1 describes the **STATUS** LED flashing sequences and the corresponding error conditions.

**Table 2-1.** STATUS LED Flashes and Corresponding Error Conditions

| Number of Flashes | Error Condition |
|---|---|
| 0 (stays lit) | Configuration has changed and has not been stored in static (non-volatile) memory. |
| 1 green | Parameter storage of nonvolatile and static parameters has been lost. Re-enter all stored parameters into the module. You can do this by re-downloading a saved configuration over the Fieldbus when the Reset switch is enabled. |
| 2 | The FP-3000 detected an error in the terminal bases in the bank or identified a module in an illegal state. Verify that the protective cover is on the local bus connector of the last terminal base and that none of the pins of that connector are touching or bent. Verify that there are no more than nine terminal bases in the bank and that no terminal bases were added to the bank while power was applied. |

The multicolored **PROCESS** LED is used to indicate the current state of the processes being controlled by the FP-3000. When a PID function block on the FP-3000 module is in initialization, the light flashes green. When all the executing PID blocks on the FP-3000 are in `TARGET` mode, the light remains lit solid green. Any active alarm of priority greater than eight results in the light being lit red. For more information on PID blocks, refer to the section *PID (Proportional–Integral–Derivative)* in Chapter 4, *Block Reference*.

Table 2-2 describes the **NETWORK** LED states.

**Table 2-2.** Description of Fieldbus NETWORK LED States

| NETWORK LED State | Meaning |
|---|---|
| Off | Fieldbus port not receiving data. |
| Flashing green | Fieldbus port is currently the Link Active Scheduler on the Fieldbus link. The FP-3000 module is controlling communications on the Fieldbus. |

**Table 2-2.** Description of Fieldbus NETWORK LED States (Continued)

| NETWORK LED State | Meaning |
|---|---|
| Steady green | Fieldbus port functioning as a basic device. The FP-3000 cannot control communication on the Fieldbus. |
| Flashing red and green | Fieldbus port is seeing traffic but is at a default Fieldbus network address. You need to assign a permanent network address through a Fieldbus configurator. |
| Steady red | Fieldbus port encountered fatal network error. Check the Fieldbus network connections. |

# Autoconfigure the FP-3000

The autoconfiguration capabilities of the FP-3000 can save time and will initialize many parameters to reasonable values.

When you use the FP-3000 with the NI-FBUS Configurator version 2.3.5, autoconfiguration is done automatically the first time an FP-3000 is powered up and any time after the manual reset switch on the module has been used. The NI-FBUS Configurator will detect the connected I/O modules and instantiate function blocks for each channel.

The autoconfiguration process may take a few minutes, depending on how many I/O modules you have connected.

To force an autoconfiguration at any other time, follow these steps:

1. Import the Device Description for the FP-3000 in the NI-FBUS Interface Configuration utility, if you have not already done so.
2. Connect the FP-3000 to the bus and start the NI-FBUS Configurator.
3. Double-click on the resource block of the FP-3000. This opens the block configuration window, which floats on top of the NI-FBUS Configurator interface.
4. Click on the **OOS** (Out of Service) button if it is not already depressed to stop any running function blocks.
5. Select the **Options** tab.
6. Scroll down to the `FP_AUTOCONFIGURE` parameter.

7. Click on **Run** and change it to `Autoconfigure`.
8. Click on the **Write Changes** button. At this point, the FP-3000 is queried and function blocks are instantiated for all of the FieldPoint modules attached to the network module.

# Updating the FP-3000 Firmware

As the FP-3000 evolves, National Instruments will release updates to the module that contain new features. These new features will include support for new types of FieldPoint I/O modules as they are released, as well as new function blocks and other enhancements. To update the firmware on an FP-3000, the FP-3000 Update utility (provided with the new firmware) must be on a machine running the NI-FBUS Communications Manager. You need to use the National Instruments AT-FBUS or PCMCIA-FBUS card to download the new firmware. You do not need any special cables to update the firmware. The new firmware features will be described by a new version of the Device Description.

**Caution** Updating the firmware on the FP-3000 may cause all FP-3000 configuration settings to be lost, depending on the degree of change in the firmware. You should make sure that all settings for the FP-3000 have been saved in your PC configurator before you update the firmware so that you can restore the settings after you update the firmware.

It is possible for two FP-3000 modules with different versions of the firmware and different device descriptions to co-exist on the same Fieldbus link or a Fieldbus system. You do not need to update all the FP-3000 modules with the new firmware. Follow these steps to update the firmware:

1. Run **FBUpgrade** on the host computer. The **Fieldbus Firmware Update** dialog box appears.
2. Select the FP-3000 module that needs to be updated.
3. For firmware version 2.3.5, select `fp3k_235.bin`, and click on **Open**.
4. Click on **Download Firmware**. This process takes about 15 minutes.
5. Restart the FP-3000. You can do this by writing the value `restart processor` to the `RESTART` parameter of the FP-3000 resource block.
6. Verify that the FP-3000 is running the new firmware by looking at the `VERSION_INFORMATION` parameter on the FP-3000 resources block. The firmware revision should match the version of the firmware you installed.
7. At the end of the process, the FP-3000 is updated to include the new features and the configuration information in the FP-3000 is cleared.

# 3

# Example Applications

This chapter provides examples that show you how to configure the FP-3000 to perform common tasks, including reading from a 4–20 mA current loop device, taking temperature readings from a thermocouple module, and controlling the output current through an analog output module. This chapter also provides information about hardware and software configuration.

These examples assume you have the NI-FBUS Configurator; however, you can use any Fieldbus configuration utility. Refer to *Using Third-Party Configuration Software with the FP-3000* in Chapter 1, *FP-3000 Network Module Overview*, for limitations of using third-party configuration software. If you are not using the NI-FBUS Configurator, refer to the documentation that came with your configurator for more details on how to perform software configuration-related tasks.

Before you run these examples, install the FP-3000 and the I/O modules. Connect the FP-3000 to the Fieldbus network and power it on. Start the NI-FBUS Configurator on your PC. Your configurator should show the FP-3000. For more information on installing the FP-3000, refer to Chapter 2, *Installation and Configuration*.

## Initial Power On: Assigning FP-3000 Network Address and Device Tag

If you are powering on the FP-3000 for the first time, you need to perform some extra steps before you try these examples. You must assign each FP-3000 a unique device tag and Fieldbus network address before it can become operational. If you are using the NI-FBUS Configurator, the configurator automatically assigns a network address to the FP-3000 when it powers up. It also assigns a tag, which you can change if desired. You can change the tag to anything you want, but it must be unique. The process of automatic assignment of network addresses and tags can take a few minutes. After the FP-3000 has a network address and tag, you can perform any of these examples. If you are not using the NI-FBUS Configurator, refer to the documentation that came with your configurator for information about setting the network address and device tag.

If you need more help setting up your network, refer to the *NI-FBUS Configurator User Manual*.

# Example 1: Converting a 4–20 mA Pressure Sensor to Fieldbus Using FP-3000

One common application of the FP-3000 is interfacing to a conventional device, such as a 4-20 mA pressure sensor or a 4-20 mA temperature transmitter. This example helps you configure the FP-3000 to interface to a 4-20 mA pressure sensor. In this example, you instantiate an AI function block, assign a tag to the block, set up scaling parameters and range for the I/O channel, schedule the function block, and download the configuration to the FP-3000.

## Getting Started

Example 1 requires the following materials:

- 4–20 mA sensor, such as a pressure sensor. If you do not have a sensor you will not be able to make measurements, but can still use this exercise to practice setting up a Fieldbus configuration.
- FP-3000 network module.
- Terminal base connected to FP-3000
- FP-AI-100, FP-AI-110, or FP-AI-111 module installed in terminal base (this example assumes you are using the FP-AI-110 module).
- Host configuration system capable of instantiating function blocks on devices (such as National Instruments NI-FBUS Configurator).

Wire the 4–20 mA current loop into a current source input for the FP-AI-110 terminals. To determine which terminals to use for channel zero, refer to the wiring diagram in the Operating Instructions that came with your FP-AI-110 module. Make sure your current loop is powered and the sensor is operating normally.

**Tip** If you autoconfigured your FP-3000, you can skip to the section *Set the Input Range*.

## Create an FP-AI-110 Block

You must create a block for the FP-AI-110 since the pressure sensor is connected to a channel on the FP-AI-110 input module. To create a block for the FP-AI-110, follow these steps:

1. Right-click on the FP-3000 in the configuration tree at the left side of the screen.
2. Select **Instantiate New Block**. This causes a dialog box to appear that lists all the blocks supported by the FP-3000 and allows them to be instantiated.
3. Select **AI 110 Block** (or the block appropriate for your AI module) from the list, then click on the **OK** button. This creates the correct analog input block on the FP-3000.

## Assign a Tag to the New Block

By default, new blocks are created without a tag. To assign a tag, follow these steps:

1. Right-click on the new block, then select **Set Tag**.
2. Enter the tag you choose in the dialog box. The tag can be up to 32 characters in length and should not contain the dot (".") character.
3. Make sure **Set to OOS Mode** is checked.
4. Click on **Set**.

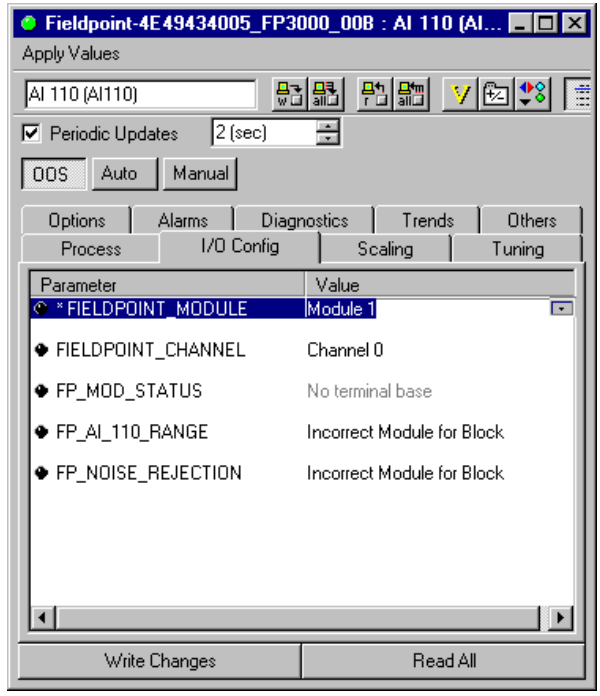
## Select the Module and Channel

1. Double-click on the block. This opens the block configuration window, which floats on top of the NI-FBUS Configurator interface. Select the **I/O Config** tab.
2. Determine the FP-AI-110 module number by counting each module in the order it is attached to the FP-3000, beginning with one. In this example, your FP-AI-110 is probably the only module connected to the FP-3000. Therefore, set the value of `FIELDPOINT_MODULE` to `Module 1`.

**Table 3-1.** Assigning Module Numbers

| FP-3000 | FP-AI-110 | FP-AO-200 | FP-DO-401 |
|---|---|---|---|
| N/A | `Module 1` | `Module 2` | `Module 3` |

3. Since the transmitter is wired to the terminals associated with channel zero on the module, set the `FIELDPOINT_CHANNEL` parameter to `Channel 0`.




## Set the Input Range

1. Click on the `FP_AI_110_RANGE` parameter on the `I/O Config` tab.
2. Set the parameter to `3.5-21 mA`, since this range most closely matches the 4–20 mA that you expect from your transmitter.

## Scale the Reading

1. Set the `XD_SCALE` parameter on the **Scaling** tab, which tells the block the range of values to expect from the transducer. Enter the following:

```
XD_SCALE
    EU_100       0.020
    EU_0         0.004
    Units_Index  A
    Decimal      3
```

This tells the AI block to expect readings in the range of 0.004 to 0.020 Amps (4 to 20 mA) from the pressure sensor. The `Decimal` field is unused by the FP-3000, but may be used in some HMIs to determine the number of digits to display to the right of the decimal point when displaying this value.

2. Determine the pressure in your desired engineering units at 4 mA and at 20 mA. For example, suppose the sensor reads 10 in$H_20$ (inches of water) at 4 mA and 250 in$H_2O$ at 20 mA. Go to the `OUT_SCALE` parameter on the **Scaling** tab and enter the following:

```
OUT_SCALE
    EU_100       250
    EU_0         10
    Units_Index  inH2O
    Decimal      3
```

3. Set the type of scaling. The block is flexible enough to either ignore scaling (`Direct`), use linear scaling (`Indirect`), or use square root scaling (`Indirect square root`). Since you want the block to use linear scaling, set the `L_TYPE` parameter on the **Scaling** tab to `Indirect`.
4. Close the block configuration window.

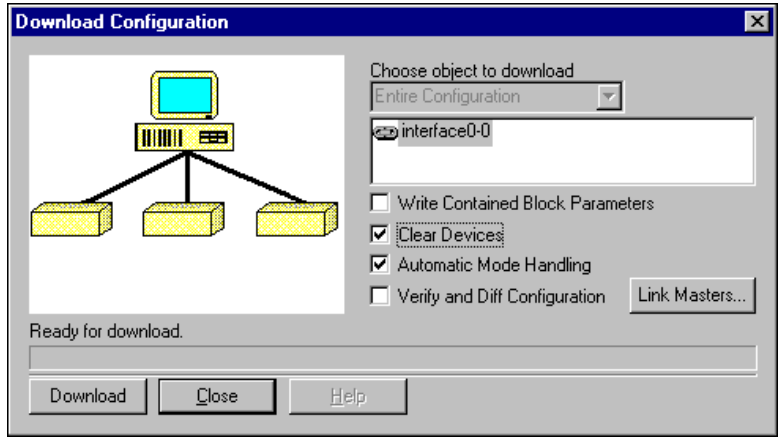
The block converts the raw 4-20 mA value and reports it in engineering units through the `PV` and `OUT` parameters of the function block.

## Set Up Scheduling

Before the block will operate, you need to schedule the block to execute. All Fieldbus function blocks (including function blocks on the FP-3000) execute according to a schedule. You can specify the order of function blocks in the schedule and the rate at which the schedule is repeated. To make the NI-FBUS Configurator create a schedule so that your block can execute, follow these steps:

1. Double-click on **Function Block Application** in the configuration tree of the NI-FBUS Configurator. This opens the Function Block Application Editor window in the middle window of the NI-FBUS Configurator.
2. Drag the block from the configuration tree to the Function Block Application Editor window. The NI-FBUS Configurator automatically generates a schedule for the block that causes it to run every second (refer to the documentation that came with your configurator for information about changing the execution period).

3. To download this schedule to the device, select **Configure»Download Configuration**. The following dialog box appears. This dialog box enables the configuration to be downloaded. Select the **Clear Devices** and **Automatic Mode Handling** checkboxes, then click on the **Download** button. Go through the download process, as described in the documentation that came with your configurator.




## Bring the Block Online

1. Go to the MODE_BLK parameter on the **Process** tab of the FP-3000 resource block and set the TARGET to Auto, or click on the **Auto** button in the block configuration window.
2. Go to the MODE_BLK parameter on the **Process** of the block you created and set the TARGET to Auto, or click on the **Auto** button in the block configuration window.
3. If the **Periodic Updates** checkbox is checked, the MODE_BLK.ACTUAL parameter should change to Auto after a few seconds. Otherwise, re-read the parameter by clicking on the **Read Selected** or **Read All** button with MODE_BLK selected. If it does not go to Auto, refer to Appendix F, *Troubleshooting*, for more information.

Once the block goes to Auto, it is fully operational. You can look at the value of OUT on the **Process** tab and see the pressure reading from the sensor in inches of water. The pressure reading can be displayed on an HMI, trended, or used for control (refer to the next example for more information about using the reading for control).

Close any open block configuration windows before you proceed to the next example.

# Example 2: Temperature Control with the FP-3000

One common application of the FP-3000 is controlling temperature. A temperature control application might include a heating element and a temperature sensor and require temperature to be maintained at a constant level, plus or minus some tolerance. Such an application would be well suited for PID control. In this example, the thermocouple measures the temperature in an enclosure, a PID block performs control, and the current output from an FP-AO-200 heats the heating element.

If you want to know how to get a thermocouple reading but are not interested in closed-loop PID control, perform only the steps in the section *Taking Temperature Readings*. After you complete those steps, the FP-3000 takes temperature readings. If you want to know how to output current to a simple device (like a resistive heating element) but are not interested in closed-loop PID control, proceed to the section *Controlling a Heating Element*, and perform the steps there. After you complete those steps, the FP-3000 controls output current.

## Getting Started

Example 2 requires the following materials:

- Thermocouple or RTD input module (FP-TC-120 or FP-RTD-122) installed in a terminal base
- FP-3000 network module
- Analog output (AO) module installed in terminal base, such as the FP-AO-200 or FP-PWM-520 (this example uses the FP-AO-200)

Wire the thermocouple to channel zero of the FP-TC-120 module, paying attention to the polarity of the thermocouple wires. Next, wire the heating element (a small light bulb or even a resistor will work) to channel zero of the FP-AO-200 module. To determine which terminals to use for channel zero, refer to the wiring diagram in the Operating Instructions that came with your modules.

You also need to connect a power supply to the V and C terminals of the FP-AO-200 module to source power on the output channel. Refer to the FP-AO-200 operating instructions and *Supplying Power for Outputs* in Chapter 2, *Installation and Configuration*, for more information.

# Taking Temperature Readings

**Tip** If you autoconfigured your FP-3000, you can skip to the section *Set the Input Range and Thermocouple Type*.

## Create an FP-TC-120 Block

Once the hardware for control loop has been installed, you need to instantiate, or create, an I/O block for the thermocouple input channel. To instantiate an I/O block in the NI-FBUS Configurator, follow these steps:

1. Right-click on the FP-3000 in the configuration tree.
2. Select **Instantiate New Block**.
3. Select **TC 120 Block** from the list, then click on the **OK** button.

## Assign a Tag to the New Block

By default, new blocks are created without a tag. To assign a tag, follow these steps:

1. Right-click on the new block, then select **Set Tag**.
2. Enter the tag you choose in the dialog box. The tag can be up to 32 characters in length and should not contain the dot (“.”) character.
3. Make sure **Set to OOS Mode** is checked.
4. Click on **Set**.

## Select the Module and Channel

1. Double-click on the new block. In the block configuration window that appears, select the **I/O Config** tab.
2. Determine the FP-TC-120 module number by counting each module in the order it is attached to the FP-3000, beginning with one. In this example, your FP-TC-120 is probably the only module connected to the FP-3000. Therefore, set the value of `FIELDPOINT_MODULE` to `Module 1`.
3. Since the transmitter is wired to the terminals associated with channel zero on the module, set the `FIELDPOINT_CHANNEL` parameter to `Channel 0`.

## Set the Input Range and Thermocouple Type

1. Click on the FP_TC_120_RANGE parameter on the **I/O Config** tab in the block configuration window. This parameter tells the AI block the range of values it should expect from the FP-TC-120 module.
2. Set the parameter to 0–2048K (degrees Kelvin).
3. Set the FP_THERMOCOUPLE_TYPE to the type of thermocouple you have connected (such as J or K type thermocouple). This allows the FP-TC-120 module to perform the necessary calculations on-board to convert the electrical input from your thermocouple to the 0-2048 degrees Kelvin that it will send to the AI block.

**Tip** The Kelvin scale covers more than all possible temperature you could measure. It corresponds to –273.15 °C to 1775 °C.

## Scale the Reading

1. You can avoid setting the XD_SCALE value manually if you set the CFG_OPTS option on the **Options** tab to Automatically Adjust XD_SCALE. This allows the FP-3000 to copy the value from FP_TC_120_RANGE straight into the XD_SCALE parameter. Otherwise, tell the block the range of values to expect from the transducer. Go to the XD_SCALE parameter on the **Scaling** tab and enter the following:

   ```
   XD_SCALE
      EU_100       2048
      EU_0         0
      Units_Index  K
      Decimal      2
   ```

   This tells the AI block to expect readings in the range of 0 to 2048 K from the thermocouple module.
2. Determine the output scale. If you want to output the temperature in degrees Kelvin, you can set OUT_SCALE to the same values as XD_SCALE above. If you want to change to Celsius, you can do so by setting OUT_SCALE as follows:

   ```
   OUT_SCALE
      EU_100       1775.00
      EU_0         -273.15
      Units_Index  °C
      Decimal      2
   ```

   These temperatures correspond to the Kelvin range you specified for the XD_SCALE.

3. Set the block to use linear scaling by setting the L_TYPE parameter on the **Scaling** tab to Indirect.

At this point, you could schedule the block to execute and bring it online. However, for this exercise, you will wait until you have finished configuring each of your blocks to do this. Refer to the section *Set Up Scheduling* if you would like to do this now.

## Bring the Block Online

1. Go to the MODE_BLK parameter on the **Process** tab of the FP-3000 resource block and set the TARGET to Auto, or click on the **Auto** button on the block configuration window.
2. Go to the MODE_BLK parameter on the **Process** tab of the new block you created and set the TARGET to Auto, or click on the **Auto** button on the block configuration window.
3. If the **Periodic Updates** checkbox is checked, the MODE_BLK.ACTUAL parameter should change to Auto after a few seconds. Otherwise, re-read the parameter by clicking on the **Read Selected** or **Read All** button with MODE_BLK selected. If it does not go to Auto, refer to Appendix F, *Troubleshooting*, for more information.

Once the block goes to Auto, it is fully operational. You can look at the value of OUT on the **Process** tab and see the temperature reading from the thermocouple in degrees Celsius. The temperature reading is ready to be used for control.

If you are only interested in taking thermocouple readings and not interested in closed loop control, you are finished with this example.

Close any open block configuration windows before you proceed to the next section.

# Controlling a Heating Element

**Tip** If you autoconfigured your FP-3000, you can skip to the section *Set the Output Range*.

## Create an FP-AO-200 Block

Create an I/O block for the FP-AO-200 channel to control the heating element. To create an I/O block in the NI-FBUS Configurator, follow these steps:

1. Right-click on the FP-3000 in the configuration tree.
2. Select **Instantiate New Block**.
3. Select **AO 200 Block** from the list, then click on the **OK** button.

## Assign a Tag to the New Block

By default, new blocks are created without a tag. To assign a tag, follow these steps:

1. Right-click on the new block, then select **Set Tag**.
2. Enter the tag you choose in the dialog box. The tag can be up to 32 characters in length and should not contain the dot (“.”) character.
3. Make sure **Set to OOS Mode** is checked.
4. Click on **Set**.

## Select the Module and Channel

1. Double-click on the new block. In the block configuration window that appears, select the **I/O Config** tab.
2. Determine the FP-AO-200 module number by counting each module in the order it is attached to the FP-3000, beginning with one. In this example, your FP-TC-120 is probably the only module connected to the FP-3000, and the FP-AO-200 is probably the second module. Therefore, set the value of `FIELDPOINT_MODULE` to `Module 2`.
3. Since the transmitter is wired to the terminals associated with channel zero on the module, set the `FIELDPOINT_CHANNEL` parameter to `Channel 0`.

## Set the Output Range

1. Click on the `FP_AO_200_RANGE` parameter on the **I/O Config** tab in the block configuration window.
2. Set the parameter to a current range of `0-21 mA`.

## Scale the Output

1. You can avoid setting the XD_SCALE value manually if you set the CFG_OPTS option on the **Options** tab to Automatically adjust XD_SCALE. This allows the FP-3000 to copy the value from FP_AO_200_RANGE straight into the XD_SCALE parameter. Otherwise, tell the block the range of values to output to the transducer module. Go to the XD_SCALE parameter on the **Scaling** tab and enter the following:

   ```
   XD_SCALE
       EU_100       0.021
       EU_0         0
       Units_Index  A
       Decimal      2
   ```

   This tells the AO block to output readings in the range of 0 to 0.021 A (0-21 mA) to the FP-AO-200 module.

2. Determine the Process Variable (PV) scale. This scaling parameter is used to convert from the units of Set Point (SP) to percent of scale. For output function blocks like analog output, SP is the value you want the block to output. For this example, set PV_SCALE to 0 to 100 percent. With these settings, a controller or operator changing the setpoint of this AO block writes the percentage of scale, with 100% being maximum output current.

   ```
   PV_SCALE
       EU at 100%   100
       EU at 0%     0
       Units Index  %
       Decimal      2
   ```

In the case of Analog Output blocks, you do not need to explicitly set the type of scaling. The block will always use both XD_SCALE and PV_SCALE parameters linearly.

## Bring the Block Online

1. Go to the MODE_BLK parameter on the **Process** tab and set the TARGET to Auto, or click on the **Auto** button on the block configuration window.
2. If the **Periodic Updates** checkbox is checked, the MODE_BLK.ACTUAL parameter should change to Auto after a few seconds. Otherwise, re-read the parameter by clicking on the **Read Selected** or **Read All** button with MODE_BLK selected. If it does not go to Auto, refer to Appendix F, *Troubleshooting*, for more information.

Once the block goes to `Auto`, it is fully operational. You can adjust the Set Point by writing a value between 0 and 100 (percent) to SP. The current flow through the heating element will vary accordingly.

If you are only interested in making the FP-3000 output current and not interested in closed loop control, you are finished with this example.

Close any open block configuration windows before you proceed to the next section.

# PID Control

## Create a PID Block

Now that your input and output blocks are functioning, you can "close the loop" by creating a control block and putting the loop under automatic control. To create a PID block, follow these steps:

1. Right-click on the FP-3000 in the configuration tree.
2. Select **Instantiate New Block**. This causes a dialog box to appear that lists all of the blocks supported by the FP-3000 and allows them to be instantiated.
3. Select **Pid Block** from the list, then click on the **OK** button.

## Assign a Tag to the New Block

By default, new blocks are created without a tag. To assign a tag, follow these steps:

1. Right-click on **--- No tag --- (PIDC)** in the configuration tree, then select **Set Tag**.
2. Enter the tag you choose in the dialog box. The tag can be up to 32 characters in length and should not contain the dot (".") character.
3. Make sure **Set to OOS Mode** is checked.
4. Click on **Set**.

## Scale the PID

Double-click on the PID block you created to launch the block configuration window. Set the PV_SCALE parameter on the **Scaling** tab. The PV_SCALE should match the output scale (OUT_SCALE) of the AI block because you will wire the output of the AI block to the IN (i.e., process variable) of the PID block. Enter the following:

```
PV_SCALE
    EU_100       1775
    EU_0         -273
    Units_Index  °C
    Decimal      2
```

When you set the PV_SCALE for the AO block, you set it to take 0 to 100 percent, so adjust the PID block to output that range. To set the OUT_SCALE parameter of the PID, enter the following:

```
OUT_SCALE
    EU_100       100
    EU_0         0
    Units_Index  %
    Decimal      2
```

## Set Up Scheduling

Before the block will operate, you need to schedule the block to execute. All Fieldbus function blocks (including function blocks on the FP-3000) execute according to a schedule. You can specify the order of function blocks in the schedule and the rate at which the schedule is repeated. If the Function Block Application Editor window is not already open, double-click on **Function Block Application** in the configuration tree of the NI-FBUS Configurator.

### Connect the PID to the AI and AO Blocks

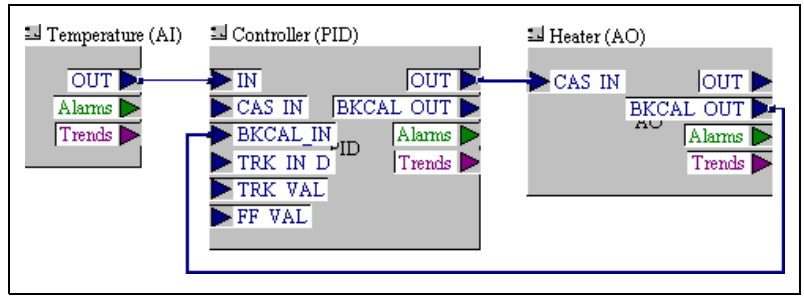
Drag your AI, AO, and PID blocks from the configuration tree to the Function Block Application Editor window, if you have not done so already. The NI-FBUS Configurator automatically generates a schedule for the blocks that causes them to run every second (refer to the documentation that came with your NI-FBUS Configurator for information about changing the execution period).

At this point, you could either use a template or wire your blocks manually. To use a template:

1. On an empty Function Block Application Editor window, right-click on the background of the Function Block Application Editor and select **FBAP Templates»PID Feedback Control**. When templates are initially placed, the template blocks are grayed out because you have not assigned a function block to the template block.
2. Assign a function blocks to the template block. Double-click on the template block to view all the blocks that match this block type in your project.
3. Select the desired block from the list that appears.
4. When prompted to use saved values, click on **No**.
5. Replace all the template blocks with function blocks from your project.

To wire manually:

1. Using the wiring tool, connect the `OUT` parameter from the AI to the `IN` parameter of the PID.
2. Connect the `OUT` parameter of the PID to the `CAS_IN` parameter of the AO.
3. Connect the `BKCAL_OUT` parameter of the AO to the `BKCAL_IN` parameter of the PID. The following figure shows what your connections should look like.



## Download and Bring the Loop into Auto

1. To download this schedule to the device, select **Configure»Download Configuration**, or click on the **Download** button. This establishes all the linkages that you "wired" in the previous step. It also schedules the PID function block, which has not yet been scheduled to execute. A dialog box appears that enables the configuration to be downloaded. Select the **Clear Devices** and **Automatic Mode Handling** checkboxes, then click on the **Download** button. Go through the download process, as described in the documentation that came with your configurator. When the download is finished, click on **Close**.
2. Double-click on the PID block to open the block configuration window.
3. Go to the `MODE_BLK` parameter on the **Process** tab of the PID block you created and set the `TARGET` to `Auto`, or click on the **Auto** button in the block configuration window.
4. Read the `MODE_BLK` parameter. The `ACTUAL` mode should read `IMan`, for Initialization Manual. This means the PID is not able to enter `Auto` because the AO block is not in Cascade mode.
5. Go to the `MODE_BLK` parameter on the **Process** tab of the AO block you created and set the `TARGET` to `Cas` and `Auto` (check both the `Cas` and `Auto` boxes). This tells the AO to operate in Cascade if possible, and otherwise, to fall back to `Auto`.
6. If the **Periodic Updates** checkbox is checked, the `MODE_BLK.ACTUAL` parameter should change after a few seconds. Otherwise, re-read the parameter by clicking on the **Read Selected** or **Read All** button with `MODE_BLK` selected. The `ACTUAL` field should be `Cas`. If it does not, refer to Appendix F, *Troubleshooting*, for more information.

Now, your loop should be running under automatic control. Verify this by reading the `ACTUAL` mode of the PID block. If it is `Auto`, the PID is trying to control the temperature. You can change the desired temperature by changing the Set Point (SP) parameter of the PID. Remember that the units of SP are the same as `PV_SCALE` for the PID, which in our example is degrees Celsius.

# Tune the PID

Adjust the PID tuning constants to match the dynamics of your temperature process. A general description of tuning PID loops is beyond the scope of this document. The parameters to change in the PID block are `GAIN`, `RESET`, and `RATE`. You can adjust these constants, change the PID Set Point, and watch how the temperature changes over time.

Typical values for GAIN, RESET, and RATE are:

| | GAIN | RESET | RATE |
|---|---|---|---|
| Pressure control | 1.2 | 3.33 | 0.8 |
| Temperature control | 3.0 | 25.0 | 10.0 |
| Flow control | 0.33 | 1.11 | 0.0 |
| Level control | 1.9 | 16.67 | 2.7 |

## Set Alarms

In the above example, it might be convenient to have the FP-3000 generate an alarm whenever the temperature goes above 40° C. This behavior can be configured from the PID block or the AI block. This example uses the PID block.

To set a high limit alarm, follow these steps:

1. Open the PID block and locate the HI_PRI parameter on the **Alarms** tab. This is the priority of the high limit alarm.
2. Set the HI_PRI parameter to 2. Fieldbus alarms can range in priority from 0-15, with 0 being disabled and 1 meaning that the alarm is detected but not reported. All other priorities cause the alarm to be reported.
3. Set the HI_LIM parameter to 40. This is the high limit that triggers the high limit alarm. The units are defined to be the same as PV_SCALE, which is degrees Celsius.
4. Set up an interface board to receive the alarm. From the NI-FBUS Configurator, drag the icon that represents your interface card (it might be named something like “interface0-0”) onto the Function Block Application Editor window. Connect the **Alarms** output of the PID to the **Alarms** input of the interface card.
5. You must download the configuration because you have changed the function block application. The PID now detects a high limit alarm whenever the temperature exceeds 40° C, and the alarm is transmitted to the interface on your PC. You need a separate program (such as the Lookout HMI package from National Instruments) to receive, display, and acknowledge the alarms. You can verify that the alarms are being detected by the PID block by reading the HI_ALM parameter. The Alarm State changes, and the Alarm Timestamp is set when the alarm goes active.

This concludes the examples. If you want, you can save your example files by selecting **File»Save**. Close any open files by selecting **File»Close**.

# 4

# Block Reference

This chapter describes the function blocks and the parameters supported by the FP-3000. For more information on blocks, refer to the *FOUNDATION Fieldbus Overview* document.

## Block Overview

A block is a predefined software module that runs on an FP-3000 and acts as a fundamental component of a control system. Each block has parameters that can be adjusted to configure that part of the system. These are referred to as contained parameters. In addition to contained parameters, some blocks have I/O parameters. *Input* parameters receive data from another block. *Output* parameters send data to another block. I/O parameters of different blocks can be connected together to establish communications between blocks. There are three types of blocks: function blocks, transducer blocks, and the resource block.

### Function Blocks

Function blocks implement the basic control algorithms that make up a control strategy. The FOUNDATION Fieldbus specification defines a set of ten fundamental (or elementary) function blocks and a set of nineteen advanced function blocks. The function blocks encapsulate a significant part of the control system behavior, thereby relieving a host (computer) of such tasks. The FOUNDATION Fieldbus specification defines the parameters of each function block, how to make each parameter accessible to host system, parameters for configuring function blocks, and I/O parameters that can communicate with other function blocks in the system. For example, an Analog Input (AI) function block has parameters to scale a transducer value to engineering units. It also has alarm limits that can be configured by a configurator or by an operator using a host application. The AI block detects and reports process alarms such as `HI_HI`, `HI`, `LO`, and `LO_LO`. The Fieldbus specification does not specify the algorithm for function blocks. For example, the specification does not define the actual equations to use in a PID function block. However, it does define all the parameters needed for configuration and operation of the PID, such as `RATE`, `GAIN`, `RESET`, and `MODE`. The execution of function blocks and the

communication between function blocks on different devices are scheduled deterministically.

A PID control loop consists of one of each of the following function blocks: an Analog Input (AI) block to read the process variable in a device (such as a transmitter), a Proportional–Integral–Derivative (PID) block to compare the process value to the setpoint and make control decisions, and an Analog Output (AO) block to move an actuator in a device (such as a valve). The PID can be located (and execute) in the transmitter, valve, or any other device (such as a controller). The execution of the AI, PID, and AO blocks is precisely scheduled on a schedule. The communication of the process value from the AI to the PID and the communications between the PID and AO blocks are also scheduled and synchronized with the block executions.

The FP-3000 function blocks conform to the standard function blocks defined by the FOUNDATION Fieldbus. In addition, the FP-3000 contains certain enhancements to the standard function blocks, such as AI, AO, DI, and DO, to permit easy configuration and diagnostics. The FP-3000 also uses National Instruments defined function blocks. All the vendor-specific blocks and enhancements are defined using Device Descriptions to interoperate with other hosts and devices.

## Transducer Blocks

Transducer blocks read from physical sensors into function blocks. Transducer blocks decouple the function blocks from the hardware details of a given device, allowing generic indication of function block input and output.

The transducer block knows the details of I/O devices and how to actually read the sensor or change the actuator. The transducer block performs the digitizing, filtering, and scaling conversions needed to provide the sensor value to the function blocks and/or makes the change in the output as dictated by the function block. Generally, there will be one transducer block per device channel. In some devices, multiplexors allow multiple channels to be associated with one transducer block.

Manufacturers can define their own transducer blocks. For some devices, including the National Instruments FP-3000, the functionality of the transducer block is included in the function block. You will see no separate transducer blocks for such devices.

**Note** There are many parameters that can be changed to modify the I/O functionality.

### Resource Block

The resource block, defined by the FOUNDATION Fieldbus specification, contains general information about the device. It also contains parameters to control the device as a whole, such as restarting the device or taking the device off-line. The resource block in the FP-3000 contains some enhancements to the standard resource block definition. For example, it includes a software version parameter that lists the version information for the FP-3000 firmware and the version of the FOUNDATION Fieldbus specifications supported.

## Types of Function Blocks

The following types of blocks are supported by the FP-3000.

### AI (Analog Input)

The AI block reads data from a single analog input channel. This block performs simple filtering and scaling of the raw data to engineering units from the input channel and supports limit alarming.

### AO (Analog Output)

The AO block writes data to an analog output channel. This block supports cascade initialization to allow upstream control blocks to switch smoothly from manual to automatic mode. It also has a faultstate behavior that allows the block to react if communications fail between itself and the upstream block.

### PID (Proportional–Integral–Derivative)

The PID block implements a PID control algorithm. When at least one PID block is present in the device, the Process LED reflects the state of the PID(s) present. In Fieldbus, a PID block must be connected to an upstream block (such as an AI block) and a downstream block (such as an AO block) before it can be used for control. These software connections are established by using host Fieldbus configuration software, such as the NI-FBUS Configurator.

### DI (Discrete Input)

The DI block reads data from discrete input channels. This block performs simple filtering and processing of the raw data from the input channel and supports limit alarming.

## DO (Discrete Output)

The DO block writes to a discrete output channel. This block supports cascade initialization to allow upstream control blocks to determine the current state of the process before assuming control. It also has a faultstate behavior that allows the block to react if communications fail between itself and the upstream block.

## CDO (Complex Discrete Output)

The CDO block serves the same purpose as the DO block and adds a number of parameters to support interlocking at three levels of priority.

**Table 4-1.** CDO Block Interlock Priorities

| Input (Descending Priority) | Notes |
|---|---|
| Safeguard Close (SAFEGUARD_CL) | Safeguard Close takes priority over any other interlock input. |
| Safeguard Open (SAFEGUARD_OP) | Safeguard Open takes priority over every other interlock input and is used to force the block to an open state (Discrete_State_1). |
| Binary Open/Close (BINARY_OP/BINARY_CL) | BINARY_OP only functions when ENABLE_OP has a value of Discrete_State_1. BINARY_CL only functions when ENABLE_CL has a value of Discrete_State_1. If both BINARY_OP and BINARY_CL are set and enabled, neither one takes effect. |
| Operator Command (OP_CMD_CXO) | OP_CMD_CXO is a contained bit string parameter that has a bit for Open and a bit for Close. Open only functions when ENABLE_OP has a value of Discrete_State_1. Close only functions when ENABLE_CL has a value of Discrete_State_1. If both Open and Close are set and enabled, neither one takes effect. |

## LOG (FieldPoint Log Block) (FP-3000 Specific)

The LOG block contains a log of error conditions and events detected by the device as it operates.

## STAT (FieldPoint Statistics Block) (FP-3000 Specific)

The STAT block contains a set of parameters that can be used to examine how the device is performing. It contains statistics describing the performance of local function blocks and the network interface.

# Expression Block (FP-3000 Specific)

The Expression block in the FP-3000 is a "programmable" block into which you can enter your own algorithm for the block in the form of a C-language style expression.

The FP-3000 Expression block includes the following features:

1. Simple discrete control that was not possible with the standard FOUNDATION Fieldbus specified blocks.
2. Custom analog alarming/interlocks.
3. Input and output signal characterization.
4. Expression processing encapsulated into a convenient module with a clean interface, so that it can be easily be incorporated wherever desired.

The architecture of the expression language in the FP-3000 is to take a textual representation of an expression and produce the appropriate code for it.

## Data Types Supported

The fundamental data types manipulated with expressions are FOUNDATION Fieldbus value/status pairings. These carry an analog (IEEE-754 floating point) or discrete (32-bit signed integer) value, as well as a status that describes the quality of the data. The discrete data type supported by the expression language is an extension of the Fieldbus discrete value status in that it is a full 32-bit integer, as opposed to a number from 0-16. This improves the flexibility of the data type and does not interface with compatibility. The FP-3000 automatically converts between discrete values and floating point values as needed. In the event that a floating point being converted to discrete results in an overflow, the value is reported as 0xFFFFFFFF.

The presence of both the value and the status in the data types allows numerical expressions to automatically calculate a status for the result of the expression. In simple cases, this keeps you from having to implement status handling logic. In the event that a value comes from a source that does not provide status information, the status information will be marked as invalid and ignored in status calculations.

**Table 4-2.** Expression Block Data Types

| Type Name | Description |
|---|---|
| vs_float | A FOUNDATION Fieldbus Value/Status record for floating point values. |
| vs_discrete | A FOUNDATION Fieldbus Value/Status record for discrete values. This discrete will be considered to be a 32-bit integer. The least significant four bits of the 32-bit integer map to the 16 Fieldbus discrete states. (As such, the discrete values may be used for computational purposes for numbers greater than 16. For the purposes of publishing on the bus, these will still be the standard one byte FOUNDATION Fieldbus discrete values.) |

## Status Calculation Rules

To simplify status calculation for simple cases, the Expression block performs all arithmetic operations on value/status pairings and manipulates statuses according to the following rules:

- Constant values and intrinsic variables have no status.
- If two statuses are being composed, all substatus values are discarded and the results have a NonSpecific Substatus.
- For two operand operators, the results have quality equivalent to the lowest quality operand (i.e., the worst status propagates to the output). The order of quality is as follows: Good, Good_NonCascade, Uncertain, and Bad.
- The limit bits reflect the effect of the operator on the limit bits of the operand(s). For example, a high-limited value that is negated would become low-limited. The result of adding two high-limited values would still be high-limited.
- Assignment operators and variable bindings leave the status entirely unchanged.
- Conditional statements ignore the statuses and only use the values.

## Syntax Rules

Each expression may only consist of the following types of lines:

- Comments—Fragments of text in the expression that are not used to generate any code. These are delimited with matching `/*` and `*/`, or by `//` and `\n` (newline character).
- Symbols—Strings of alphanumeric characters beginning with a letter or underscore.

- C-language style program constructs—Program statements that define the logic of the expression with the use of conditional statements, variable assignments, etc. Each valid statement ends with a semi-colon.
- Empty lines for vertical spacing—You can add empty lines to improve readability.

## Program Constructs

Program constructs in the Expression block are C-language like. They are comprised of keywords, symbols, constants, operators, conditional statements for flow control, and delimiters (for the most part).

### Reserved Symbols

- a_in_0, a_in_1, a_in_2, a_in_3, a_in_4, a_in_5, a_in_6, a_in_7

  These symbols correspond to the EXPR block's analog input parameters A_IN_0, A_IN_1, A_IN_2, A_IN_3, A_IN_4, A_IN_5, A_IN_6, and A_IN_7, respectively.
- d_in_0, d_in_1, d_in_2, d_in_3

  These symbols correspond to the EXPR block's discrete input parameters D_IN_0, D_IN_1, D_IN_2, and D_IN_3, respectively.
- a_state_0, a_state_1, a_state_2, a_state_3

  These are symbols for the EXPR block's parameters that you can use as local variables in an expression. They correspond to the EXPR block parameters A_STATE_0, A_STATE_1, A_STATE_2, and A_STATE_3, respectively.
- a_out_0, a_out_1, a_out_2, a_out_3

  These symbols correspond to the EXPR block's analog output parameters A_OUT_0, A_OUT_1, A_OUT_2, and A_OUT_3, respectively.
- d_out_0, d_out_1, d_out_2, d_out_3, d_out_4, d_out_5, d_out_6, d_out_7

  These symbols correspond to the EXPR block's discrete output parameters D_OUT_0, D_OUT_1, D_OUT_2, D_OUT_3, D_OUT_4, D_OUT_5, D_OUT_6, and D_OUT_7, respectively.
- x_time

  Provides read-only access to the FP-3000's sense of application time in the Expression block. Application time is expressed in the number of seconds since January 1, 1972.

**Note** Use the spreadsheet `fp3kexpr.xls` provided with the FP-3000 package to generate the expression text if you have a need for the block to execute at specified times during the day.

- `x0`, `x1`, `x2`, `x3`, `x4`, `x5`, `x6`, `x7`

  Provides eight unsigned 32-bit integer variables. These variable values are persistent across block executions but are not stored in non-volatile memory. As such, power cycling the FP-3000 will leave them with an unknown value unless they are initialized by the user in such cases.

- `last_oos`

  This symbol defines a boolean read-only variable in the EXPR block, to represent whether the EXPR block's actual mode was Out Of Service (`OOS`) the previous time it was scheduled to execute.

User-defined variables are not allowed in the Expression block language. However, you may (for readability reasons) choose to rename any one of the symbols listed above using the keyword "rename."

Keywords

- rename

  This is used to rename one of the symbols listed above to a user defined name.

  For example, `rename a_state_0 counter`
  will rename the symbol `a_state_0` to `counter`. You can then use the name `counter` in the rest of your expression. Such user-defined names must be declared before use in the expression.

## Constants

These are the constants typed in by the user in an expression.

For example, `a_state_0 = 0;`

In the above example, the `0` on the right-hand side of the assignment operator (`=`) is the constant value.

User-defined constants of the form `<#define XXX 123>` are not supported in the Expression block language.

## Operators

Operators let you manipulate the Expression block variables. Operators supported by the Expression block language are listed in the following table:

**Table 4-3.** Expression Block Operators

| Class of Operation | Operation | Operators |
|---|---|---|
| Logical | Logical AND | <op1> **&&** <op2> |
| | Logical OR | <op1> **||** <op2> |
| | Logical NOT | **!**<op1> |
| Bitwise | Bitwise AND | <op1> **&** <op2> |
| | Bitwise OR | <op1> **|** <op2> |
| | Bitwise NOT | **~**<op1> |
| | Bitwise XOR | <op1> **^** <op2> |
| Arithmetic | Add | <op1> **+** <op2> |
| | Subtract | <op1> **-** <op2> |
| | Multiply | <op1> ***** <op2> |
| | Divide | <op1> **/** <op2> |
| | Modulo Divide | <op1> **%** <op2> |
| | Negate | **–**<op1> |
| Comparison | Equivalence | <op1> **==** <op2> |
| | Inequivalence | <op1> **!=** <op2> |
| | Greater Than | <op1> **>** <op2> |
| | Greater Than or Equal | <op1> **>=** <op2> |
| | Less Than | <op1> **<** <op2> |
| | Less Than or Equal | <op1> **<=** <op2> |
| Assignment | Simple Assignment | <l-val> **=** <op1> |
| Grouping | NA | ( <expr> ) |

## Flow Control

Flow control and grouping constructs supported by the expression language will be a variant of the C language designed to eliminate the possibility for endless loops and other pathological cases. This variant includes standard if...then...else statements and compositions.

**Table 4-4.** Flow Control and Group Constructs

| Operation | Syntax |
|---|---|
| Conditional | if (condition) <*statement(s) to be executed for true case*><br>[ else <*statement(s) to be executed for false case*> ]<br><br>for loop and while loop constructs are not supported |
| Composition | { <statement> ; ... } |

Example:

```
//Conditional statement with an if...then clause
if (a_state_0 > 5)  { //begin a composition
   A_state_0 = 0;
   A_state_1 = 5;
}                     //end composition
else //The else clause is optional and not required
   a_state_0 = a_state_0 + 1; //else clause is not
   required to be a composition.
```

## Delimiters

Every valid program statement must be terminated using the C-language style delimiter ‘;’.

For example,

```
a_in_0 = 0.0;
a_in_1 = 0.0;
```

## Functions

### Intrinsic Functions

The Expression block language provides a set of standard functions that implement some higher math functions, as well as functions for controlling the status of a value.

**Table 4-5.** Function Operators

| Class of Operation | Operation | Operators |
|---|---|---|
| Math | Square Root | sqrt(<op1>) |
| | Exponential | e<op1> |
| | Logarithm (base e) | log(<op1>) |
| | Absolute Value | abs(<x>) |
| Status | Status checking<br>is good cascade status<br>is bad status<br>is good non-cascade status<br>is uncertain | <br>is_good(<op1>)<br>is_bad(<op1>)<br>is_gnc(<op1>)<br>is_unc(<op1>) |
| | Status manipulator | set_status(<op1>, <stat>) |
| | Status retriever | get_status(<op1>) |
| Value manipulation | Fraction to engineering unit value converter | to_scaled (<value>, <op1>, <op2>) |
| | Engineering unit value to fraction and value | to_unity (<value>, <op1>, <op2>) |

### Function Descriptions

| Function Name | `is_bad` |
|---|---|
| Function Syntax | `Uint32 is_bad(vs_float F) OR`<br>`Uint32 is_bad(vs_discrete D)` |
| Input | Any symbol that is a value status combination |
| Purpose | This function checks to see if the quality of status of the input parameter is Bad. If it is, the function returns TRUE, otherwise it returns FALSE. |
| Return value | Unsigned 32-bit integer TRUE (value 1) or FALSE (value 0) |

| Function Name | `is_good` |
|---|---|
| Function Syntax | `Uint32 is_good(vs_float F) OR`<br>`Uint32 is_good(vs_discrete D)` |
| Input | Any symbol that is a value status combination. |
| Purpose | This function checks to see if the status of the input parameter is Good Cascade. If it is, the function returns TRUE, otherwise it returns FALSE. |
| Return value | Unsigned 32-bit integer TRUE (value 1) or FALSE (value 0) |

| Function Name | `is_gnc` |
|---|---|
| Function Syntax | `Uint32 is_gnc(vs_float F) OR`<br>`Uint32 is_gnc(vs_discrete D)` |
| Input | Any symbol that is a value status combination |
| Purpose | This function checks to see if the status of the input parameter is Good Non-Cascade. If it is, the function returns TRUE, otherwise it returns FALSE. |
| Return value | Unsigned 32-bit integer TRUE (value 1) or FALSE (value 0) |

| Function Name | `is_unc` |
|---|---|
| Function Syntax | `Uint32 is_unc(vs_float F) OR`<br>`Uint32 is_unc(vs_discrete D)` |
| Input | Any symbol that is a value status combination |
| Purpose | This function checks to see if the status of the input parameter is Uncertain. If it is, the function returns TRUE, otherwise it returns FALSE. |
| Return value | Unsigned 32-bit integer TRUE (value 1) or FALSE (value 0) |

| Function Name | `set_status` |
|---|---|
| Function Syntax | `vs_discrete set_status(vs_discrete, status) OR`<br>`vs_float set_status(vs_float, status)` |
| Input | Any symbol that is a value status combination |
| Purpose | This function sets the status of the input parameter to the input status and returns the input parameter with the old value and newly set status. |
| Return value | `vs_discrete` or `vs_float` input parameter with the old value but new status |

| Function Name | `get_status` |
|---|---|
| Function Syntax | `unsigned char get_status(vs_discrete) OR`<br>`unsigned char get_status(vs_float)` |
| Input | Any symbol that is a value status combination |
| Purpose | This function retrieves the status of the input parameter and returns it. |
| Return value | Status of the input parameter |

| Function Name | `to_scaled` |
|---|---|
| Function Syntax | `vs_float to_scaled( float valueToScale,`<br>`float engUnit0,`<br>`float engUnit100)` |
| Input | `engUnit0` is the low value of the scale range, `engUnit100` is the high value of the scale range, and `valueToScale` is the floating point value (implicitly between 0 and 1) that needs to be scaled (i.e., 0 <= `valueToScale` <= 1) |

| | |
|---|---|
| Purpose | Scales the floating point value, `valueToScale`, to a percentage in the scale range specified by `engUnit0` and `engUnit100`. |
| Return value | The scaled floating point value status pair.<br>`engUnit0 <= vs_float.value returned.value <=engUnit100` |

| | |
|---|---|
| Function Name | `to_unity` |
| Function Syntax | `vs_float to_unity(float euValue, float engUnit0, float engUnit100)` |
| Input | `engUnit0` is the low value of the scale range, `engUnit100` is the high value of the scale range, and `euValue` is the value that needs to be converted from engineering units to an implicit 0 to 1 scale.<br>`EngUnit0 <= euValue <= engUnit100` |
| Purpose | Un-scales the floating point value input in `euValue` parameter to a value using the scale range specified by `engUnit0` and `engUnit100` for the conversion. |
| Return value | The floating point value status pair, with floating point value being between 0 and 1.<br>`0 <= vs_float_value returned.value <= 1` |

| | |
|---|---|
| Function Name | `to_discrete` |
| Function Syntax | `vs_discrete to_discrete(vs_float)` |
| Input | A symbol that is a value status float |
| Purpose | Converts a value status floating point variable to a discrete value status variable. |
| Return value | A discrete value status record. |

### User Functions

User functions are not supported. You cannot define sub-routines within an expression block.

# Supported FieldPoint Modules and Channels

The FP-3000 supports a wide variety of I/O channels, each with different types of configuration information. For example, a thermocouple channel includes parameters for thermocouple type and has ranges for different temperatures; a current loop channel includes parameters for filter frequency and has a different group of available ranges to chose from. Because of these differences in parameters, the FP-3000 has a block specific to each type of channel it supports. For example, to use a thermocouple connected to an I/O channel on an FP-TC-120 module, the FP-3000 provides a FP-TC-120 AI Block. This block is a standard Analog Input block augmented with parameters specific to the thermocouple channel on the FP-TC-120.

The FP-3000 does not support all FieldPoint I/O modules. In particular, the counter module is one module that is often requested, but not supported. Support for other modules might be added in later firmware revisions.

The following table shows the function block that corresponds to each supported FieldPoint I/O module.

**Table 4-6.** FieldPoint Modules

| Channel Type | Supported Module | Block Type |
|---|---|---|
| Analog Input | FP-AI-100 | Fp Ai 100 |
| | FP-AI-110 | Fp Ai 110 |
| | FP-AI-111 | Ap Ai 111 |
| | FP-TC-120 | Fp Tc 120<br>Fp Tc 120 C (cold junction compensation) |
| | FP-RTD-122 | Fp Rtd 122 |
| Analog Output | FP-AO-200 | Fp Ao 200 |
| | FP-AO-210 | Fp Ao 210 |
| | FP-PWM-520 | Fp Pwm 520 |

**Table 4-6.** FieldPoint Modules (Continued)

| Channel Type | Supported Module | Block Type |
|---|---|---|
| Discrete Input | FP-DI-300 | Fp Di 300 |
| | FP-DI-301 | Fp Di 301 |
| | FP-DI-330 | Fp Di 330 |
| Discrete Output | FP-DO-400 | Fp Do 400<br>Fp Cdo 400 (interlock logic) |
| | FP-DO-401 | Fp Do 401<br>Fp Cdo 401 (cold junction) |
| | FP-DO-403 | Fp Do 403<br>Fp Cdo 403 |
| | FP-DO-410 | Fp Do 410<br>Fp Cdo 410 |
| | FP-RLY 420 | Fp Rly 420<br>Fp Crly 420 (cold junction) |
| | FP-RLY-422 | Fp Rly 422<br>Fp Crly 422 |

# PID Control

In a Fieldbus network, a PID control loop is composed of three function blocks: an Analog Input (AI) block, a Proportional Integral Derivative (PID) block, and an Analog Output (AO) block. The PID block's `IN` parameter is connected to the AI block's `OUT` parameter. The PID uses this linkage to determine the current value of the process variable it is controlling. The PID uses linkage from the PID's `OUT` parameter to the AO block `CAS_IN` parameter to adjust the AO block's setpoint. To allow the cascade to be correctly initialized, a third back calculation linkage is created that allows the AO block to send its current setpoint back up to the PID block. These linkages are established using configuration software, such as the NI-FBUS Configurator. For more information, refer to *Example 2: Temperature Control with the FP-3000* in Chapter 3, *Example Applications*.

## PID Loop Execution Time

The FP-3000 can run approximately 50 PID loop iterations per second. For loops with all three function blocks contained in the FP-3000, a single loop iteration can run as quickly as 15–20 ms. You can choose how to distribute these iterations—running 1 PID loop 50 times per second, or 50 PID loops once per second (actually at most 49 PID loops can run on a single FP-3000, see below). You can also choose a combination between these two extremes.

When deciding how fast to run PID loops, keep in mind the A/D update rate of the FieldPoint I/O modules. For example, the FP-AI-100 has an update rate of 2.8 ms, but the FP-AI-110 has an update rate of 170 ms to 1500 ms. It does not make sense to run a PID loop at a faster rate than the update rate. The A/D converter cannot supply new information faster than its update rate. For example, attempting to run a PID loop every 100 ms with an FP-AI-110 will cause stale readings to be processed. The update rate of each I/O module is documented in the operating instructions of each module and in the National Instruments catalog.

## PID Loop Execution Time Considerations

Up to 150 function blocks can be instantiated on an FP-3000. One of these must be a resource block. This means that you can instantiate up to 149 other function blocks. Each PID loop requires 3 blocks (AI, PID, AO). Thus, at most 49 PID loops can run on a single FP-3000.

Because function blocks must execute serially, there is a trade-off between how many PID loops you run and how fast you can run them. To see this relationship, calculate the total execution time for your desired function blocks. Also note that in your schedule, you generally want to leave 30% of your macrocycle for unscheduled communications (to allow alarm information to be passed between devices, HMI communications, etc.). The function blocks used in a PID loop have maximum execution times (worst-case times) as shown below. (Note: You can find this information in the block information tab for the desired function block. Under the **View** menu, choose **Preferences**. Click on the **Block View** tab. Check the **Show Block Information** checkbox. Now when you double-click on a function block, a new tab called **Block Information** will appear. Look at the parameter `EXECUTION_TIME` on the **Block Information** tab for the individual function block. If the number is a hexadecimal number, you can change it to decimal by right-clicking and unchecking **Hexadecimal Data**. This decimal value of this parameter is the execution time in 1/32 ms. Dividing this number by 32 will give the maximum execution time in

milliseconds). The NI-FBUS Configurator does not allow function blocks to be scheduled any closer together than these maximum execution times.

PID: 8 ms
AI: 6 ms
AO: 6 ms

For one PID loop:

Total time scheduled for function block execution = 20 ms (8 ms + 6 ms + 6 ms). Multiplying this number by 1.5 will allow 30% unscheduled time = 30 ms

No unscheduled time = 1/0.020 s = 50 iterations per second (for one loop)

30% unscheduled time = 1/0.030 s = about 33 iterations per second (for one loop)

For 49 PID loops:

Total time scheduled for function block execution = 980 ms (49*(8 ms + 6 ms + 6 ms)). Multiplying this number by 1.5 will allow 30% unscheduled time = 1470 ms

No unscheduled time = 1/0.980 s = about one iteration per second (for each of 49 loops)

30% unscheduled time = 1/1.470 s = about one iteration per 1.5 seconds (for each of 49 loops)

# Alarming

The Fieldbus network supports event notification messages from field devices like the FP-3000. Fieldbus function blocks use event notification messages to implement alarms and events. Alarms are used to report conditions that can either be active or inactive. An example of an alarm is when the measured value of an AI block exceeds the user-defined alarm limit. The function block sends an event notification alarm to the host each time conditions transition between active and inactive. Events are notifications of one-time events as they are detected by the field device. An example of an event is the update event that is reported as a host application or operator modifies configuration parameters of the device.

# Alarm Parameters

Each block contains a fixed set of alarms it can report, such as High Alarm or Deviation Alarm. Each alarm parameter is a record describing the current state of that particular alarm or event. It contains a number of fields the device uses to reveal the current state of the alarm. Following is a list of the meaning of each field in an alarm record.

## UNACKNOWLEDGED

The UNACKNOWLEDGED subfield indicates the acknowledgment state of the alarm or event. A host application typically acknowledges the unacknowledged alarm when an operator sees and acknowledges the alarm.

## ALARM_STATE/UPDATE_STATE

The current state of the alarm or event can be determined through the ALARM_STATE (for alarms) or UPDATE_STATE (for events) field of the alarm or event parameter. This parameter shows the active/clear state of alarms and the reported/unreported state of both alarms and events.

The first piece of state information in the state field is the active/clear state of the alarm. An alarm is considered to be active when the alarm condition is detected to be true. In the case of a limit alarm, the alarm is active when the process variable, such as the temperature being measured, is beyond the limit. The alarm state clears when the process variable returns within the limit, adjusted for any hysterisis factor specified in the ALARM_HYS parameter.

All blocks have one alarm known as the Block alarm. The Block alarm is considered active when any block error conditions (in the BLOCK_ERR parameter) are true. The Block alarm clears when the last block error condition clears.

For all alarms, the alarm condition is checked during each block execution. Events, on the other hand, are not considered to be active or clear, but simply one-time notifications.

The second piece of state information in the state field is the reported status of the alarm or event. When an alarm or event condition is reported to the host computer, an event notification message is broadcast on the bus if the alarm has a priority greater than 1. For alarms without priority parameters (Block alarms and events), the priority defaults to 2 and is always reported. To confirm the receipt of the event notification, the host responds with a confirmation message (different from the acknowledgment message

discussed above). The confirmation message confirms that the message has made it across the bus to the host. It does not indicate acknowledgment by the operator. Until the device receives the confirmation message, the alarm or event is considered to be unreported. In the case where a block has no alarm linkage, the device waits to report the alarm or event until the linkage is established. If an alarm is unable to be reported to a host, the Active/Clear state of the alarm stays constant until the alarm can be reported.

### TIME_STAMP

The time the alarm was detected by the FP-3000. In FOUNDATION Fieldbus, all devices share a common sense of time, as published by the time master on the bus. This shared sense of time is used to timestamp alarm conditions as they occur, rather than when they are reported.

### SUBCODE

For Block alarm, the subcode of the last error condition detected. The Block alarm, shared by all FOUNDATION Fieldbus function blocks in the FP-3000, is unique in that there are multiple conditions that can cause it to go active and clear. Any error condition reported in the BLOCK_ERR parameter of the block causes the block alarm to go active. The alarm does not clear until the last error condition in BLOCK_ERR has been resolved. To indicate which error condition the BLOCK_ALM is reporting, the SUBCODE subfield of the parameter is set to indicate the block error condition causing the fault. If additional error conditions are detected, the SUBCODE and TIME_STAMP are updated to reflect the latest condition detected, but the alarm will not be reported again until every error condition has been resolved and a new condition triggered.

### VALUE

For limit alarms, the value of the parameter causing the alarm condition (the Process Variable). For update events, the index of the modified static parameter.

## Status and Mode Handling Overview

Status and mode handling are crucial aspects of developing a distributed control application. Mode refers to the mode of operation of a function block. Allowable modes depend on the type of block, but generally include Out of Service mode, Manual mode, and Automatic mode.

Status refers to the quality of a variable communicated between two blocks. When a block receives a variable with bad status, it can affect its current mode of operation.

## Status Handling

Parameters that can be communicated between blocks are composed of a value and a status. The value is the data to be communicated, and the status describes the quality of the data. When two blocks are logically connected over the Fieldbus using host configuration software, the block that is sending data is called the publisher. The block receiving the data is called the subscriber. When communication is established from a publisher to a subscriber, the subscriber takes the value and the status of the published variable. If communication is not established between the publisher and the subscriber, the subscriber has a status that reflects the lack of communication. Statuses themselves are composed of three subfields: the quality, the substatus, and the limit.

### Quality

Table 4-7 describes the `Quality` subfields.

**Table 4-7.** Quality Values

| Value | Meaning |
|---|---|
| `Bad` | The value is bad, the sensor is defective, or communication has not been established. The value should not be trusted by the receiver. |
| `Uncertain` | The quality of the data is unknown. This can be caused by errors or a lack of calibration in the physical I/O transducer. Blocks can generally be configured to treat values of `Uncertain` quality as either `Bad` or `Good` with the `STATUS_OPTS` parameter. |
| `Good_NonCascade` | The value is good and from a block that does not support cascade initialization. This status is also used when an alarm is active. |
| `Good_Cascade` | The value is good and from a block that supports cascade initialization. |

## Substatus

The Substatus field is used to describe more specifically the cause of the given quality. For example, a status with the quality of Bad might have a substatus of Device Failure, indicating that the value should not be trusted because it is from a device that has failed. Another common substatus is Non-specific. The Non-specific substatus is used when no other substatus applies. There are numerous substatuses.

## Limit

Table 4-8 describes the Limit subfield values.

**Table 4-8.** Limit Values

| Value | Meaning |
|---|---|
| None | The value is not limited. |
| Low | The value is at a lower limit. This can be caused by a transducer limitation or setpoint limits. |
| High | The value is at a high limit. This can be caused by a transducer limitation or setpoint limits. |
| Constant | The value is at a fixed value and cannot move. This results from the block supplying the value being in manual mode. |

# MODE_BLK Parameter and Mode Handling

The block's mode behavior is controlled with the MODE_BLK parameter. The MODE_BLK parameter contains four fields that allow the current mode of the block to be read and the desired mode for the block to be written. The four fields are TARGET mode, ACTUAL mode, PERMITTED mode, and NORMAL mode.

## TARGET Mode

The desired mode of execution for the block. An operator or process engineer normally writes this to put the block in the desired mode of operation.

## ACTUAL Mode

An indicator of the current mode of execution of the device. This is a read-only parameter. Normally, the actual mode of the block is equal to the target mode of the block. However, configuration errors or other conditions can cause it to differ from the target mode.

## PERMITTED Mode

A list describing the modes into which the block may be target. For example, in order to set a `TARGET` mode of `Man`, the `Man` mode bit must be set in the `PERMITTED` mode field. The `PERMITTED` mode field also has an effect on the way output blocks shed modes. For a description of mode shedding, refer to Appendix D, *Advanced Function Block Behavior*.

## NORMAL Mode

The block's normal mode of operation is stored in the `NORMAL` field. This field is not used internally by the device, but is a guide for an operator.

**Table 4-9.** Modes

| Mode | Meaning | Bit Set in TARGET Mode |
|---|---|---|
| Out of Service<br>`OOS` (0x80) | The block is idle and does not execute. If the resource block is out of service, all other blocks in the device are also out of service. | Out of Service, and optionally any other valid target mode |
| Manual<br>`Man` (0x10) | The output of the block is set by an operator through a write to the output parameter. No block processing other than writing to the I/O channel is performed. | Manual |
| Initialization Manual<br>`IMan` (0x40) | The upstream block in a cascade loop is preparing to enter `Auto` mode. This mode cannot be set as a target mode. It is used internally by control blocks as they establish cascade loops. | Not applicable |
| Local Override<br>`LO` (0x20) | The block's faultstate or interlock capability is causing the block to override its normal output value. This mode cannot be set as a target mode. | Not applicable |
| Automatic<br>`Auto` (0x08) | The block operates normally with a setpoint specified manually through a write to the setpoint parameter. | Automatic |

**Table 4-9.** Modes (Continued)

| Mode | Meaning | Bit Set in TARGET Mode |
|---|---|---|
| Cascade<br>`Cas` (0x04) | The block operates normally with a setpoint specified automatically through a connection from an upstream block to the `CAS_IN` (cascade input) parameter. Before the block can enter this mode, the cascade is initialized automatically to avoid windup. | Cascade and Automatic |
| Remote Cascade<br>`RCas` (0x02) | The block operates normally with a setpoint specified automatically through a write from a host computer to the `RCAS_IN` (remote cascade input) parameter. Before the block can enter this mode, the cascade is initialized automatically to avoid windup. | Remote Cascade and Automatic. For more information on Remote Cascade operation, refer to the section *Fault State and Mode Shedding* in Appendix D, *Advanced Function Block Behavior*. |
| Remote Output<br>`ROut` (0x01) | The output of the block is set manually through a write to the `ROUT_IN` parameter. No block processing other than writing to the I/O channel is performed. Before the block can enter this mode, the cascade is initialized automatically to avoid windup. | Remote Output and Automatic. For more information on Remote Output operation, refer to the section *Fault State and Mode Shedding* in Appendix D, *Advanced Function Block Behavior*. |

# A

# Configuring the FP-3000

The FP-3000 has three configuration switches accessible from an opening in the top of the module. These switches are shown in Figure A-1.

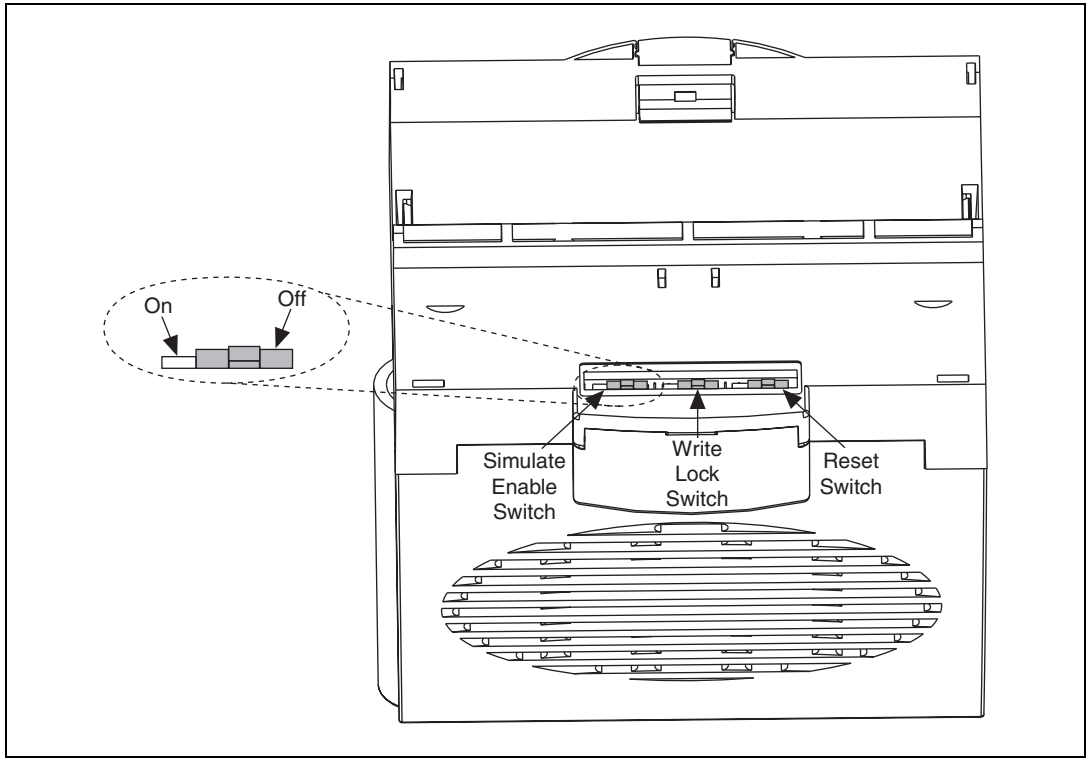



**Figure A-1.** Configuration Toggle Switches

## Simulate Enable

When On, this switch allows simulation on I/O blocks to be enabled. The status of this jumper is shown in the Simulate Active bit in the resource block's BLOCK_ERR parameter. If the switch is On, the bit is set, and the device allows simulation to be enabled on I/O blocks.

# Write Lock

When On, the device rejects writes to block configuration parameters. Linkages between blocks still function correctly.

# Reset

When On, this switch causes the device to reset all configuration information to factory defaults on power up. To resume normal operation, this switch must be switched off and power to the device must be cycled a second time.

# B

# Fieldbus Parameters

## Fieldbus Parameters (In Alphabetical Order)

### ACK_OPTION (Alarming)

Allows alarms to be automatically acknowledged by the block with no outside intervention. This is useful if you are not interested in acknowledging certain alarms from a block.

### ALARM_HYS (Alarming)

The amount a value must move off an alarm limit, in percent of scale, for the alarm to be considered clear. This helps prevent alarms from constantly "toggling" on and off when the process value is near the configured alarm limit.

### ALARM_SUM (Alarming)

A summary of the status of alarms in the block. Allows alarms to be disabled.

### ALERT_KEY (Alarming)

A user-assigned identification number reported in alarm messages from the block that allows HMI applications to sort and filter alarms and events. This parameter is set for each function block to indicate which physical unit the function block is associated with.

### BAL_TIME (Tuning)

Time, in seconds, for the bias or ratio to change from the internal working value to the operator set value. Also, the time constant used by the integral term of the PID to obtain balance when the output is limited and the block is in `Auto`, `Cas`, or `RCas` mode.

## BIAS (Tuning)

The bias value, in engineering units, used to calculate the function block output.

## BKCAL_HYS (Limiting)

The amount a block's output value must move off a limit, in percent of scale, for the limit status to be turned off.

## BKCAL_IN (Limiting, Process)

Will be linked to a downstream block's `BKCAL_OUT` or `SELECT_OUT` parameter. This is used to initialize a control loop through cascade initialization. Cascade initialization allows smooth transfer for a control block from manual to automatic mode. To bypass cascade initialization, this parameter can be left unwired and manually set to a status of `Good, non-cascade`.

## BKCAL_OUT (Process)

A back-calculation value published to the `BKCAL_IN` of an upstream block in a control loop. The `BKCAL_OUT` parameter has the value of the block's current output. Before a cascade loop is initialized, the upstream block can use this value to smoothly transfer to loop control.

## BKCAL_OUT_D (Process)

An output value published to an upstream discrete block. The upstream block can use this value to smoothly transfer to loop control.

## BLOCK_ALM (Alarming, Diagnostic)

An alarm parameter used to report error conditions detected within the block, such as block Out of Service.

## BLOCK_ERR (Diagnostic)

A list of error conditions for hardware and software components associated with the block.

**Table B-1.** Error Codes

| Error Code | Code | Description |
|---|---|---|
| Other | 0x0001 | Undefined block error condition. |
| Block Configuration Error | 0x0002 | The block has detected an error in its configuration. This usually indicates a static parameter has been left uninitialized. |
| Link Configuration Error | 0x0004 | The logical connection between this block and another block is misconfigured. |
| Simulate Active | 0x0008 | For I/O function blocks, this indicates that simulation is enabled. For the resource block, this indicates that the simulate jumper has been set, allowing simulation to be enabled in other blocks. |
| Local Override | 0x0010 | The block has locally overridden the output value. This can be the result of an interlock or faultstate. |
| Device Faultstate Set | 0x0020 | The block's faultstate behavior is active. |
| Device Needs Maintenance Soon | 0x0040 | The device is reporting performance degradation that will soon require maintenance. |
| Input Failure/BAD PV Status | 0x0080 | Either the input transducer channel has reported a failure, or the input parameter from an upstream block has reported a failure. For an AI block, this could be caused by an open circuit being detected on the FP-AI-100 input module. |
| Output Failure | 0x0100 | The output transducer channel has reported a failure. For an AO block, this could indicate that the FP-AO-200 cannot drive the current request, perhaps due to an open circuit. |
| Memory Failure | 0x0200 | The storage for nonvolatile and static parameters was corrupted. |
| Lost Static Data | 0x0400 | The device was unable to restore the values of static parameters after a restart. |

**Table B-1.** Error Codes (Continued)

| Error Code | Code | Description |
|---|---|---|
| Lost NV Data | 0x0800 | The device was unable to restore the values of nonvolatile parameters after a restart. |
| Readback Check Failed | 0x1000 | The value read back from the output channel does not match the value the output channel was set to. |
| Device Needs Maintenance Now | 0x2000 | The device needs to be maintained now. |
| Power-Up | 0x4000 | The device has just powered up. |
| Out of Service | 0x8000 | The block is currently out of service. |

## BYPASS (Scaling, Tuning)

Allows the normal control algorithm to be bypassed if the `CONTROL_OPT` parameter's Bypass Enable option is selected. If control is bypassed, the PID uses its setpoint value in percent of scale as its output value and does not attempt to do any PID control.

## CAS_IN (Process)

A remote setpoint value. If used, will be linked to an output of an upstream block. In some blocks, may be left unlinked, and a local setpoint value (typically SP) will be used instead.

## CAS_IN_D (Process)

A remote setpoint value. If used, will be linked to an output of an upstream block. In some blocks, may be left unlinked, and a local setpoint value (typically `SP_D`) will be used instead.

## CHANNEL (I/O, Process)

Used by I/O function blocks to select a physical I/O channel. This mapping is defined by the manufacturer. In the FP-3000, this parameter is automatically updated when the `FIELDPOINT_MODULE` and `FIELDPOINT_CHANNEL` parameters are modified.

## CLR_FSTATE (Faultstate, Option)

Writing Clear to this parameter causes the device-wide faultstate to be cleared and output blocks to resume normal execution. Also see SET_FSTATE and FAULT_STATE parameters.

## CONFIRM_TIME (Alarming)

The lower bound on the time the device waits to send alert report messages if no confirmation is received from a host.

## CONTROL_OPTS (Option, Scaling)

A list of options used to adjust the way control blocks, such as the PID block, operate.

**Table B-2.** Control Options

| Options | Description |
|---|---|
| Bypass Enable | If set, lets you set the BYPASS parameter and bypass the algorithm's control. |
| SP-PV Track in Man | Causes the setpoint to track the process variable in Man. |
| SP-PV Track in ROut | Causes the setpoint to track the process variable in ROut. |
| SP-PV Track in LO or IMan | Causes the setpoint to track the process variable in LO or IMan. |
| SP Track Retained Target | Causes the setpoint to track the input value of the retained target of the block. The retained target of the block is the lowest priority mode set in the target mode field of the MODE_BLK parameter of the block. For example, if the RCas bit is set in the target mode, the setpoint tracks RCAS_IN. |
| Direct Acting | Defines the relationship between changes to the PV and changes to the output. For example, consider a case with a fixed SP while the process variable varies. When Direct Acting is set, an increase in the process variable causes the control block's output value to be increased. When Direct Acting is clear, an increase in the process variable causes the control block's output value to be decreased. |
| Track Enable | Enables external tracking. If Track Enable is true, and TRK_IN_D is true, TRK_VAL overwrites the value at the output of the block except when Man is the target mode. |
| Track in Manual | Enables tracking in Manual mode. |

**Table B-2.** Control Options (Continued)

| Options | Description |
|---|---|
| Use PV for BKCAL_OUT | When set, uses the process variable as the value for BKCAL_OUT, instead of the setpoint. |
| Obey SP Limits if Cas or RCas | When set, confines the setpoint to values within SP_HI_LIM and SP_LO_LIM, even when the setpoint comes from another function block. |
| No OUT Limits in Man | Unused in FieldPoint. |

## CYCLE_SEL (Tuning)

Identifies the block execution methods available. Unused in National Instruments FP-3000.

## CYCLE_TYPE (Tuning)

Used to select the block execution method. Unused in National Instruments FP-3000.

## DD_RESOURCE (Diagnostic)

Unused in FieldPoint.

## DD_REV (Diagnostic)

The revision of the device description used by the device.

## DEV_REV (Diagnostic)

The revision of the device.

## DEV_TYPE (Diagnostic)

The manufacturer's model number for the device.

## DISC_ALM (Alarming)

The current state of the discrete alarm, along with a time and date stamp.

## DISC_LIM (Alarming)

The discrete input state in which an alarm should be generated.

## DISC_PRI (Alarming)

Priority of the discrete alarm.

## DV_HI_ALM (Alarming)

The current state of the deviation high alarm, along with a time and date stamp.

## DV_HI_LIM (Alarming)

The deviation limit between the PID block setpoint and process value, in engineering units, beyond which the deviation high alarm is considered active.

## DV_HI_PRI (Alarming)

The priority of the deviation high alarm.

## DV_LO_ALM (Alarming)

The current state of the deviation low alarm, along with a time and date stamp.

## DV_LO_LIM (Alarming)

The deviation limit between the PID block setpoint and process value, in engineering units, beyond which the deviation low alarm is considered active.

## DV_LO_PRI (Alarming)

The priority of the deviation low alarm.

## FAULT_STATE (Faultstate, Option)

The current status of the device faultstate. It can be set and cleared with `SET_FSTATE` and `CLR_FSTATE`. If it is set, all output blocks in the device initiate their own faultstate behavior.

## FEATURE_SEL/FEATURES (Diagnostic, Option)

The `FEATURES` parameter lists features supported by the device. Use the `FEATURE_SEL` parameter to manually enable and disable the supported features listed in the `FEATURES` parameter.

**Table B-3.** Feature Parameter Options

| Option | Description |
|---|---|
| Unicode | The device supports strings in Unicode format. The FP-3000 does not support this feature. |
| Reports | The device supports event report messages for alarming. If this feature is not selected in the FEATURE_SEL parameter, the FP-3000 continues to detect alarms and events, but does not report them over the bus. In this case, the host must poll the alarm parameters to detect alarm conditions as they change. |
| Faultstate | The device supports Faultstate behavior for output blocks. |
| Soft Write Lock | The device supports locking of configuration of parameters with the WRITE_LOCK parameter in the resource block. With this feature selected and the WRITE_LOCK parameter written to Set, writes to all static configuration parameters are disallowed. |
| Hard Write Lock | The device supports locking of configuration parameters. For the FP-3000, a switch on the back of the device must also be set. If Hard Write Lock is enabled, the switch disallows writes to all configuration parameters in the device, including FEATURE_SEL. |
| Out Readback | The device provides a way for the action of output transducers to be verified through a readback. The FP-3000 does not support this feature. |
| Direct Write | The device provides a manufacturer-specific way to directly write to I/O channels. The FP-3000 does not support this feature. |

## FF_GAIN (Scaling, Tuning)

The gain by which the feed-forward input is multiplied before it is added to the output value of the control block.

## FF_SCALE (Scaling)

The scaling parameter used by the feed-forward value of the block.

## FF_VAL (Process, Scaling, Tuning)

The feed-forward value.

## FIELD_VAL (Process, Scaling, Tuning)

The value from the input channel, in percent of scale.

## FIELD_VAL_D (Process, Scaling, Tuning)

The value from the discrete input channel.

## FREE_SPACE (Diagnostic, Process)

The percentage of memory available on the device. This can be used when instantiating blocks to determine the remaining capacity of the FP-3000. This value will be zero in preconfigured devices since they do not allow user configuration.

## FREE_TIME (Diagnostic, Process)

Percentage of block processing time that is available to process additional blocks. Unused in FieldPoint.

## FSTATE_TIME (Faultstate, Option)

Time (in seconds) to delay from the detection of loss of communications with the host for the output block remotes setpoint until the enaction of the fault state output.

## FSTATE_VAL (Faultstate, Option)

The setpoint value to be used on failure.

**Note** The I/O option **Failsafe to value** must be selected.

## FSTATE_VAL_D (Faultstate, Option)

The discrete setpoint value to be used on failure.

**Note** The I/O option **Failsafe to value** must be selected.

## GAIN (Tuning)

The gain constant used by the PID in calculating the proportional component of the output.

## GRANT_DENY (Option)

Allows HMI applications to determine access privileges for block parameters.

**Note** The device does not use this parameter to restrict parameter access itself. It is only for the benefit of host applications.

## HARD_TYPES (I/O, Process)

A list of available channel types. As I/O modules are inserted and removed from the FP-3000 bank, bits in this field change to reflect the presence or absence of types of I/O channels.

**Table B-4.** Hard Types

| Bitmask | Description |
|---|---|
| `Analog Input` | This bit is set if the FP-3000 has analog input channels available. |
| `Analog Output` | This bit is set if the FP-3000 has analog output channels available. |
| `Discrete Input` | This bit is set if the FP-3000 has discrete input channels available. |
| `Discrete Output` | This bit is set if the FP-3000 has discrete output channels available. |

## HI_ALM (Alarming)

The current state of the high alarm, along with a time and date stamp.

## HI_HI_ALM (Alarming)

The current state of the high-high alarm, along with a time and date stamp.

## HI_HI_LIM (Alarming)

The limit, in PV units, beyond which the high-high limit alarm is considered active.

## HI_HI_PRI (Alarming)

The priority of the high-high limit alarm.

## HI_LIM (Alarming)

The limit, in PV units, beyond which the high limit alarm is considered active.

## HI_PRI (Alarming)

The priority of the high limit alarm.

## IN (Process, Scaling, Tuning)

The primary input of the block.

## IN_1 (Process, Scaling, Tuning)

The secondary input of the block.

## IO_OPTS (I/O, Options, Scaling)

A bitmask used to adjust the way I/O blocks (AI, DI, AO, and DO) operate.

**Table B-5.** Operation Bitmasks

| Bitmask | Description |
|---|---|
| Invert | In discrete blocks, this maps a physical state of Discret_State_0 to Discret_State_1 and maps every other physical transducer state to Discret_State_0. |
| SP-PV Track in Man | Causes the setpoint to track the process variable in Man. |
| SP-PV Track in LO or IMan | Causes the setpoint to track the process variable in LO or IMan. |
| SP Track Retained Target | Causes the setpoint to track the input value of the retained target of the block. The retained target of the block is the lowest priority mode set in the target mode field of the MODE_BLK parameter of the block. For example, if the RCas bit is set in the target mode, the setpoint tracks RCAS_IN. |
| Increase to Close | Remaps the block's scaling so that as the input increases, the output decreases. |
| Faultstate to Value | When set, the block's faultstate behavior sets the output value to the value in FSTATE_VAL. When clear, the block's faultstate behavior leaves the output value at its current setting. |
| Use Faultstate Value on Restart | When set, causes the output value of output blocks to go to faultstate value immediately after a device restart. When clear, uses the value in nonvolatile memory. |
| Target to Man if Faultstate Active | When set, sets the target mode of the block to manual mode when faultstate goes active. |

**Table B-5.** Operation Bitmasks (Continued)

| Bitmask | Description |
|---|---|
| Use PV for BKCAL_OUT | When set, uses the process variable as the value for BKCAL_OUT, instead of the setpoint. |
| Low Cutoff | When set, enables the AI low cutoff parameter. |

## ITK_VER

The version of the Interoperability Test Kit with which this device was tested.

## L_TYPE (Scaling)

The linearization type. This parameter affects the way the value from the transducer is linearized in the analog input block before it is presented as the block output. In all cases, the FIELD_VAL parameter behaves as follows:

```
FIELD_VAL = 100 * (transducer_value – XD_SCALE.EU0) /
(XD_SCALE.EU100 – XD_SCALE.EU0)
```

FIELD_VAL can be simply described as the percentage of span reading from the transducer, and therefore its units are percent.

**Table B-6.** Linearization Types

| Type | Description |
|---|---|
| Direct | The block output is directly taken from the transducer value:<br>`OUT = transducer_value` |
| Indirect | The block output is scaled according to OUT_SCALE from the value in FIELD_VAL:<br>`OUT = OUT_SCALE.EU0 + ((FIELD_VAL/100) * (OUT_SCALE.EU100 – OUT_SCALE.EU0))` |
| Indirect Square Root | The block output is scaled according to OUT_SCALE from the value in FIELD_VAL. Before the field value is rescaled, the square root is taken.<br>`OUT = OUT_SCALE.EU0 + (SQRT(FIELD_VAL / 100) * (OUT_SCALE.EU100 – OUT_SCALE.EU0))` |
| Uninitialized | An invalid setting. The device reports a configuration error with an Uninitalized L_TYPE. |

## LIM_NOTIFY (Alarming)

A limit on the number of unconfirmed alarm/event notification messages the device can have active at once. This must be less than or equal to MAX_NOTIFY.

## LO_ALM (Alarming)

The current state of the low alarm, along with a time and date stamp.

## LO_LIM (Alarming)

The limit, in PV units, beyond which the low limit alarm is considered active.

## LO_LO_ALM (Alarming)

The current state of the low-low alarm, along with a time and date stamp.

## LO_LO_LIM (Alarming)

The limit, in PV units, beyond which the low-low limit alarm is considered active.

## LO_LO_PRI (Alarming)

The priority of the low-low limit alarm.

## LO_PRI (Alarming)

The priority of the low limit alarm.

## LOW_CUT (I/O, Option, Scaling, Tuning)

With an L_TYPE of Indirect Square Root, this can be used to establish a floor (in percent of scale) for values from the transducer. Values below this floor are considered to be zero. This feature must first be enabled by setting Low Cutoff in the IO_OPTS parameter.

## MANUFAC_ID (Diagnostic)

The ID of the manufacturer of the device. For National Instruments devices, it is 0x4E4943. The parameters MANUFAC_ID, DEV_TYPE, DEV_REV, and DD_REV are used in combination for a host tool to locate the Device Description for this device.

## MAX_NOTIFY (Alarming)

The maximum number of unconfirmed alarm/event notification messages the device supports.

## MEMORY_SIZE (Diagnostic)

Unused by FieldPoint.

## MIN_CYCLE_T (Diagnostic, Process)

The length of the shortest macrocycle the device supports.

## MODE_BLK (Diagnostic, Process)

Sets the operational and permitted modes of the block. The following table describes the operational and permitted modes of the block.

**Table B-7.** Field Modes

| Field Mode | Description |
|---|---|
| TARGET | The desired mode of operation of the block. This field is writable. Several bits may be set in this field, and typically, the highest priority bit that is set will be considered to be the target mode. OOS is the highest priority bit. |
| ACTUAL | A bit reflecting the current state of operation of the block. This is a read-only field. Only one bit will be set at a time by the block. ACTUAL mode is a function of the target mode and the current conditions in which the block is executing. Several conditions (such as cascade initialization or fault state conditions) can cause the ACTUAL mode to differ from the Target mode. |
| PERMITTED | A bitmask indicating which modes are permitted target modes and which are not. This field is writable. This could be used by the plant operator to disallow certain modes the block would normally be permitted to have as a Target mode. |
| NORMAL | Not used by the block. This can be used by an operator to store the normal mode of operation for the block in normal plant operations. This field is writable. Used by the NI-FBUS Configurator to set the TARGET mode after download when the Automatic Mode Handling feature is in effect. |

**Table B-8.** Operational Modes

| Operational Mode | Description |
|---|---|
| Out of Service (OOS) | The block is out of service, block execution is suspended, and all output parameters take a status of Bad::OutOfService. |
| Initialization Manual (IMan) | The block is in the process of initializing a cascade. This is used for upstream (control) blocks when they are initializing for smooth transfer into Automatic mode. This is not a valid TARGET mode, but it is a valid ACTUAL mode. |
| Local Override (LO) | Faultstate or an interlock is active and causing the output value of the block to be overridden. This is not a valid TARGET mode, but is a valid ACTUAL mode. |
| Manual (Man) | The output value of the block is set by the user. |
| Automatic (Auto) | The output value of the block is set by the block algorithm, and the block is using a local value for its setpoint. |
| Cascade (Cas) | The setpoint for the block is taken from the CAS_IN parameter, which is typically connected to the output of another block. This mode cannot be entered before cascade initialization takes place. When Cascade is desired as a TARGET mode, the Auto bit is also set in the TARGET. |
| Remote Cascade (RCas) | Like Cascade mode, in Remote Cascade mode the setpoint of the block comes from an outside data source. Unlike Cascade mode, in Remote Cascade mode the setpoint is sourced from the *RCAS_IN* parameter, which is written by a host application and not another function block. |
| Remote Output (ROut) | Remote Output mode is analogous to Remote Cascade mode, except that the remote host application directly sets the output of the block and not the setpoint. In the case of an analog output block, this bypasses setpoint rate and absolute limiting. |

## NV_CYCLE_T (Diagnostic)

The regular time interval, in milliseconds, at which nonvolatile parameters are committed to nonvolatile storage. A value of zero means that the parameters are never written to nonvolatile memory. Note that nonvolatile parameters are stored to nonvolatile memory when they are changed by a user over the network. The NV_CYCLE_T parameter sets the rate at which changes caused by the device itself are stored to nonvolatile memory.

## OUT (Process, Scaling, Tuning)

The current output value of the block.

## OUT_D (Process)

The current output value of a discrete block.

## OUT_HI_LIM (Limiting)

A limit for the maximum output value from a block in modes other than manual.

## OUT_LO_LIM (Limiting)

A limit for the minimum output value from a block in modes other than manual.

## OUT_SCALE (Scaling)

The scaling parameter used for the output parameter.

**Table B-9.** OUT_SCALE Parameter

| Subfield | Meaning |
|---|---|
| EU_100 | Engineering units value at 100 percent of scale. |
| EU_0 | Engineering units value at zero percent of scale. |
| UNIT_INDEX | Actual engineering units code (such as mA). |
| DECIMAL | Number of digits a host shows to the right of the decimal for display purposes. |

## OUT_STATE (Process)

Index to the text description of the discrete output state.

## PV (Process, Scaling, Tuning)

The process variable, or primary variable for this block. For AI and control blocks such as PID, this represents a measurement of the state of the process (such as temperature or level). For AO blocks, the process variable is the current setpoint of the block.

## PV_D (Process)

The process variable, or primary variable for this block. For DI and discrete control blocks, this represents a measurement of the discrete state of the process. For DO blocks, the process variable is the current discrete setpoint of the block.

## PV_FTIME (Scaling, Tuning)

The filter time, in seconds, used in input blocks. For analog blocks, it is the time constant for a low pass exponential filter used to damp out rapid oscillations in the input value before using it as the process variable. For discrete blocks, it is the time the PV must remain constant after a change for the change to be reported.

## PV_SCALE (Scaling)

The scaling parameter used by the process variable of the block. Converts from percent of scale to a process variable in engineering units. Contains the same subfields as `OUT_SCALE`.

## PV_STATE (Process)

Index to the text describing the state of a discrete PV.

## RA_FTIME (Tuning)

The filter time constant, in seconds, for the value to be used in the ratio.

## RATE (Tuning)

The time constant for the derivative component of the PID block. A zero disables the derivative term. The units are seconds.

## RCAS_IN (Mode Shedding, Process)

The cascade input for a control or output block set by a remote host. This is propagated to the setpoint of the block when it is in `RCas` mode. If the block is in `RCas` mode and this parameter is not updated in `SHED_RCAS` time (a parameter in the resource block), the block enters mode shedding. Mode shedding allows the block to degrade from `RCas` mode into some higher priority mode.

## RCAS_IN_D (Mode Shedding, Process)

The discrete cascade input for a control or output block set by a remote host. This is propagated to the setpoint of the block when it is in `RCas` mode. If the block is in `RCas` mode and this parameter is not updated in `SHED_RCAS` time (a parameter in the resource block), the block enters mode shedding. Mode shedding allows the block to degrade from `RCas` mode into some higher priority mode.

## RCAS_OUT (Process)

The back calculation output used by the supervisory host when establishing a Remote cascade loop.

## RCAS_OUT_D (Process)

The discrete back calculation output used by the supervisory host when establishing a Remote cascade loop.

## READBACK (Scaling, Tuning)

The valve or actuator position read back from the transducer, in transducer units.

## READBACK_D (Scaling, Tuning)

The transducer state for the actual discrete valve or actuator position.

## RESET (Tuning)

The time constant for the integral component of the PID block. It is measured in seconds per repeat (so larger values have less effect, and `INF` effectively disables the integral term).

## RESTART (Diagnostic, Option)

Allows the user to restart the device remotely.

**Table B-10.** Restart Values

| Value | Behavior |
|---|---|
| `Restart Resource` | Restarts the device. |
| `Restart to Defaults` | Restarts the device, restoring all parameter values to default values. |
| `Restart Processor` | Restarts the device as if the power was cycled. |

**Caution** Using `Restart to Defaults` causes all your configured parameters in the FP-3000 to revert to their factory default settings.

## ROUT_IN (Mode Shedding, Process)

The cascade input set by a remote host. This is propagated to the output of the block when it is in `ROut` mode. If the block is in `ROut` mode and this parameter is not updated in `SHED_ROUT` time (a parameter in the resource block), the block enters mode shedding. Mode shedding allows the block to degrade from `ROut` mode into some higher priority mode.

## ROUT_OUT (Process)

This is the back calculation output used by the host when trying to establish a remote output loop. While the loop is being established, it is the current value of the output channel and can be used by the host to initialize for smooth transfer of control.

## RS_STATE (Diagnostic, Process)

The current state of the device.

**Table B-11.** Device States

| State | Meaning |
|---|---|
| `Start/Restart` | The device has just started a restart cycle. |
| `Initialization` | The device is performing startup diagnostics. |
| `Failure` | A hardware failure has been detected. |
| `On-Line Linking` | The device is online and waiting for new parameter linkages to be established. |
| `On-Line` | The device is online and in service. |
| `Standby` | The device is online, but currently out of service. |

## SEL_1 through SEL_3 (Process, Scaling, Tuning)

Input values for the selector.

## SEL_TYPE (Scaling)

Defines the selector action—High, Medium, or Low.

## SET_FSTATE (Faultstate, Option)

Allows the user to set the device faultstate to active. This, in turn, forces all output blocks into their own faultstate behavior.

## SHED_OPT (Mode Shedding, Option)

Controls the way blocks enter mode shedding. Each option listed below has a companion `No Return` option. The `No Return` shedding options change the target mode of the device to the shed mode and prevent the device from re-entering `RCas` or `ROut` mode after the shed condition has ended.

**Table B-12.** Shed Conditions

| Shed Mode | Behavior |
|---|---|
| `Normal Shed` | The block sheds into the next higher-priority mode set in the permitted mode field of `MODE_BLK`. |
| `Shed to Auto` | The block sheds into automatic mode. |
| `Shed to Manual` | The block sheds into manual mode. |
| `Shed to Retained` | The block sheds to the next higher priority mode set in the target mode field of `MODE_BLK`. |

## SHED_RCAS (Mode Shedding)

The shed time for the `RCAS_IN` parameter. If the block is in `RCas` mode and the `RCAS_IN` parameter has not been updated in `SHED_RCAS` time, the block performs mode shedding as determined by the `SHED_OPT` parameter.

## SHED_ROUT (Mode Shedding)

The shed time for the `ROUT_IN` parameter. If the block is in `RCas` mode and the `ROUT_IN` parameter has not been updated in `SHED_RCAS` time, the block performs mode shedding as determined by the `SHED_OPT` parameter.

## SIMULATE (Option)

Used to bypass the physical I/O channel and allow the block to operate normally, using a simulated I/O channel. For this feature to be enabled on an FP-3000, you must set a switch on the back of the device. To see how to configure the switch, refer to Appendix A, *Configuring the FP-3000*.

## SIMULATE_D (Option)

Used to bypass the physical I/O channel and allow the block to operate normally, using a simulated discrete I/O channel. For this feature to be enabled on an FP-3000, you must set a switch on the back of the device. To see how to configure the switch, refer to Appendix A, *Configuring the FP-3000*.

## SP (Process)

The analog setpoint.

## SP_D (Process)

The discrete setpoint.

## SP_HI_LIM (Limiting, Option)

The upper limit on an operator-entered setpoint for the block. If the operator enters a setpoint that exceeds this value, the setpoint is considered to be `SP_HI_LIM` with a status that indicates that it is limited.

## SP_LO_LIM (Limiting, Option)

The lower limit on an operator-entered setpoint of the block. If the operator enters a setpoint below this value, the setpoint is considered to be `SP_LO_LIM` with a status that indicates that it is limited.

## SP_RATE_DN (Limiting, Option)

In `Auto` mode, the rate, in PV units per second, the setpoint can be moved downwards. If the setpoint moves faster than `SP_RATE_DN`, the block acts as if the setpoint is moving downwards at the maximum rate with a status bit that indicates that it is limited. If set to zero, the setpoint is used immediately.

## SP_RATE_UP (Limiting, Option)

In `Auto` mode, the rate, in PV units per second, the setpoint can be moved upwards. If the setpoint moves faster than `SP_RATE_UP`, the block acts as if the setpoint is moving upwards at the maximum rate with a status bit that indicates that it is limited. If set to zero, the setpoint is used immediately.

## ST_REV (Diagnostic)

`ST_REV` is incremented by one each time a static parameter is modified.

## STATUS_OPTS (Faultstate, Limiting, Option)

A collection of options that affect the status behavior of the block.

**Table B-13.** Status Options

| Option | Meaning |
|---|---|
| IFS if Bad IN | Set the status of the block output to initiate faultstate if the *IN* parameter goes bad. |
| IFS if Bad CAS_IN | Set the status of the block output to initiate faultstate if the CAS_IN parameter goes bad. |
| Use Uncertain as Good | If set, blocks will treat the Uncertain status on an input parameter as if it were a Good status. If clear, Uncertain status is treated as Bad. |
| Propagate Failure Forward | If the status of the IN parameter of the block is Bad::Device_Failure or Bad::Sensor_Failure, the failure will be propagated to the OUT parameter. No alarm will be generated. |
| Propagate Failure Backward | If the status at BKCAL_IN or from the physical I/O channel is bad, the failure will be propagated to the BKCAL_OUT parameter. No alarm will be generated. |
| Target to Manual if Bad IN | Set the target mode of the block to Man if the IN parameter has a bad status. |
| Uncertain if Limited | For input or calculation blocks, the output status will be set to Uncertain if the transducer or calculated value is limited (i.e., at its high or low limit). |
| Bad if Limited | Set the output status to Bad if the transducer value is limited (i.e., at its high or low limit). |
| Uncertain if Manual Mode | Set the output status to Uncertain if the block is in Man mode. |
| Do Not Select if Not Auto Mode | Set the output status to Do Not Select if the block is not in an ACTUAL mode of Auto, CAS or RCas and not initializing. This is useful for blocks upstream of the selector block. |
| Do Not Select if Not Cas Mode | Set the output status to Do Not Select if the block is not in an ACTUAL mode of CAS or RCas and is not initializing. This is useful for blocks connected to a selector block. |

## STRATEGY

Used to identify groupings of blocks. Not used by the block itself.

## TAG_DESC (Diagnostic)

User description for the purpose of the block.

## TEST_RW (Process)

Unused by the block algorithm. Used to test interoperability of reads and writes of different parameter types.

## TRK_IN_D (Scaling)

Used to enable tracking of the output value to TRK_VAL. When this is true, the output value of the block takes on the value specified in TRK_VAL.

## TRK_SCALE (Scaling)

The scaling parameter used for the value specified by TRK_VAL.

## TRK_VAL (Scaling)

The value the block will track when tracking is enabled by TRK_IN_D.

## UPDATE_EVT (Diagnostic)

The current state of the update event, along with a time and date stamp. This event is issued whenever a static parameter is changed and ST_REV is incremented. The index information for the parameter that changed and the new value of ST_REV is included in the alert.

## WRITE_ALM (Alarming)

State of the alert generated if WRITE_LOCK is cleared, along with a time and date stamp.

## WRITE_LOCK (Option)

The software write lock for the device. When this is set to true, writes to all configuration parameters of all blocks are disallowed. The WRITE_ALM block alarm is active when writes are allowed and clear when they are disallowed.

## WRITE_PRI (Alarming, Option)

The priority of the write alarm.

## XD_SCALE (Scaling)

The scaling parameter used to interpret values from the physical I/O channel. This is used to translate from a physical transducer value to a percent of scale.

**Table B-14.** Scaling Parameter Values

| Subfield | Meaning |
|---|---|
| EU_100 | Engineering units value at 100 percent of scale. |
| EU_0 | Engineering units value at zero percent of scale. |
| UNIT_INDEX | Actual engineering units code (such as mA). |
| DECIMAL | Number of digits a host shows to the right of the decimal for display purposes. (Not used by the NI-FBUS Configurator.) |

## XD_STATE (Process)

Index to the text description of the transducer state.

C

# FP-3000 Specific Parameters

## FP-3000 Specific Parameters (In Alphabetical Order)

### A_IN_0—A_IN_7

The Expression Block analog input parameters.

### A_OUT_0—A_OUT_3

The Expression Block analog output parameters.

### A_STATE_0—A_STATE_3

The Expression Block parameters you can use as local variables in an expression.

### ALG_RUN_TIME

The length of time (in milliseconds) the block algorithm takes to run. This time may be less than the total execution time reported by the block; it does not include certain overhead external to the block algorithm itself.

### BINARY_CL

An interlock input. When in `Discrete_State_1`, the output of the block is forced closed (`Discrete_State_0`). This interlock input has priority over all interlock inputs, except `SAFEGUARD_CL` and `SAFEGUARD_OP`. In the event both `BINARY_OP` and `BINARY_CL` are in discrete state 1, both are considered to be in `Discrete_State_0`.

### BINARY_OP

An interlock input. When in `Discrete_State_1`, the output of the block is forced open (`Discrete_State_1`). This interlock input has priority over all interlock inputs, except `SAFEGUARD_CL` and `SAFEGUARD_OP`. In the event both `BINARY_OP` and `BINARY_CL` are in `Discrete_State_1`, both are considered to be in `Discrete_State_0`.

## BLOCK_ALMS_ACT

A more detailed explanation of the active block alarms.

## BLOCK_RESET

Lets you reset the statistics or the configuration of an individual function block.

**Table C-1.** Block Reset Options

| Option | Description |
|---|---|
| Contained Parameters | Sets all the contained parameters (standard and FP-3000 specific) of the function block to default values. This is useful to set a specific block to a known state without affecting the behavior of the other blocks in the FP-3000. The block should be in `OOS` mode to reset the contained parameters. |
| Statistics | Resets the various statistics counts associated with the block. |

## CFG_OPTS

An option in the I/O function blocks to automatically set the scaling or alter the block behavior. Generally, this option can make configuration easier.

**Table C-2.** Configuration Options

| Option | Description |
|---|---|
| Automatically Adjust `XD_SCALE` | Enabling this option allows the block to automatically determine its `XD_SCALE` parameter, based on the physical I/O channel range, which is set by parameters such as `FP_AI_100` range. |
| Ignore Unconnected Interlock Inputs | By default, all interlock inputs on the CDO block must be good, or the block enters fault state. Enabling this option allows inputs with a status of `Bad::Not_Connected` to be ignored. |

## CHECKBACK

A list describing the state of the interlock logic in the CDO block. Use `CHECKBACK` to determine how interlocks are operating.

**Table C-3.** Checkback States

| State | Description |
|---|---|
| Safeguard Open | The SAFEGUARD_OP parameter is in Discrete_State_1, and the block has opened the output. |
| Safeguard Close | The SAFEGUARD_CL parameter is in Discrete_State_1, and the block has closed the output. |
| Binary Open | The BINARY_OP parameter is in Discrete_State_1, and the block has opened the output. |
| Binary Close | The BINARY_CL parameter is in Discrete_State_1, and the block has closed the output. |
| Safeguard Signal (LO) | The block has entered local override mode due to an active interlock. |
| Discrepancy in Open | Unused in FieldPoint. |
| Discrepancy in Close | Unused in FieldPoint. |
| Actuator Open | Unused in FieldPoint. |
| Actuator Close | Unused in FieldPoint. |
| Open Torque Exceeded | Unused in FieldPoint. |
| Close Torque Exceeded | Unused in FieldPoint. |
| Readback Simulated | Unused in FieldPoint. |
| Travel Time Exceeded | Unused in FieldPoint. |
| Local Lockout Active | Unused in FieldPoint. |

## CLEAR_LOG

Write to this parameter to clear all existing log entries.

## D_IN_0–D_IN_3

The Expression Block discrete input parameters.

## D_OUT_0–D_OUT_7

The Expression Block discrete output parameters.

## DEV_OPTS

A list of device-wide options that can be turned on and off at will. The current firmware supports only one option.

**Table C-4.** Device Options

| Options | Description |
|---|---|
| Disable CFG_OPTS in all I/O blocks | This bit disables all block configuration options in CFG_OPTS. It is not recommended that this bit be set since the additional behavior can make configuration easier. |

## EN_CLOSE

Enables the Close command from parameter OP_CMD_CXO. If EN_CLOSE is not set, then OP_CMD_CXO Close commands will be ignored.

## EN_OPEN

Enables the Open command from parameter OP_CMD_CXO. If EN_OPEN is not set, then OP_CMD_CXO Open commands will be ignored.

## EVENT_0—EVENT_19

The last 20 events.

## EVENT_FILTER

Restricts the entries shown in the log of the Log Block. The values are a combination of Configuration Error, Operational Warning, and Operational Error.

## EXECUTION_STATISTICS

A repository containing performance statistics for a given block. Use EXECUTION_STATISTICS to assess the performance of a given configuration and allow appropriate changes to be made.

**Table C-5.** Execution Statistics

| Statistic | Description |
|---|---|
| EXEC_COUNT | The number of times the block executed since the statistics were last reset. |
| EXEC_MISS_COUNT | The number of times the block failed to execute as scheduled since the statistics were last reset. |
| STALE_COUNT | The number of times the block received stale data since the statistics were last reset. |
| EVENT_COUNT | The number of events logged since the statistics were last reset. |
| RESET_TIME_STAMP | The time the statistics were last reset. |

## EXPR_DOMAIN_INDEX

Virtual field device index of the domain associated with this expression.

## FIELDPOINT_CHANNEL

The FieldPoint I/O channel the block has been assigned to. Writing to this parameter updates the CHANNEL parameter appropriately.

FP-3000 determines the CHANNEL parameter automatically based on the FIELDPOINT_MODULE and FIELDPOINT_CHANNEL parameters. You do not need to set the CHANNEL parameter.

## FIELDPOINT_MODULE

The FieldPoint I/O module containing the channel the block has been assigned to. FieldPoint modules are numbered, starting with one, at the I/O module closest to the FP-3000.

## FP_AI_100_RANGE

Allows the range of a channel on a FieldPoint FP-AI-100 to be adjusted.

## FP_AI_110_RANGE

Allows the range of a channel on a FieldPoint FP-AI-110 to be adjusted.

## FP_AI_111_RANGE

Allows the range of a channel on a FieldPoint FP-AI-111 to be adjusted.

## FP_AO_200_RANGE

Allows the range of a channel on a FieldPoint FP-AO-200 to be adjusted.

## FP_AO_210_RANGE

Allows the range of a channel on a FieldPoint FP-AO-210 to be adjusted.

## FP_AUTOCONFIGURE

This parameter, present in the resource block, causes the FP-3000 to automatically configure itself. The FP-3000 detects all the I/O modules present and instantiates the appropriate I/O function blocks. It creates a function block for each I/O channel. It tags the function blocks and sets the contained parameters to appropriate defaults. The resource block must be set to `OOS` mode before you set the Autoconfigure option. If Autoconfigure is set on an existing configuration, the FP-3000 deletes all the existing blocks and linkages before creating new blocks.

## FP_CJC_SOURCE

Allows the cold junction compensation to be adjusted on a FP-TC-120 module.

**Note** Cold junction compensation is global to the entire module and affects every channel on the module.

## FP_MOD_LIST

This parameter, present in the resource block, lists all the I/O module types that are currently plugged in.

## FP_MOD_STATUS

The status of the FieldPoint I/O module associated with the function block.

**Table C-6.** Module Status

| Status | Description |
|---|---|
| `No Base` | There is no terminal base in the specified module position. |
| `Base, But No Module` | There is a terminal base in the specified module position, but no module is installed in the base. |
| `Unconfigured Module` | There is a module in the specified position, but the FP-3000 is unable to configure it. |

**Table C-6.** Module Status (Continued)

| Status | Description |
|---|---|
| `Module in Configuration` | There is a module in the specified position, the FP-3000 has attempted to configure the module, and the module is in the process of configuration. |
| `Module Okay` | There is a module in the specified position, and it is configured and operating correctly. |
| `Incorrect Module for Block` | There is a module in the specified position, but the block is incompatible with the module. |

## FP_NOISE_REJECTION

Allows the noise rejection filter of an Analog Input module to be adjusted.

## FP_PWM_520_PERIOD

The period of the pulse width modulated waveform, in milliseconds.

## FP_RTD_122_RANGE

Allows the range of a channel on a FieldPoint FP-RTD-122 to be adjusted.

## FP_RTD_TYPE

Allows adjustment of the RTD type of a channel on an FP-TC-122 module.

## FP_TC_120_CJ_RANGE

Allows the range of the cold junction compensation channel on a FieldPoint FP-TC-120 to be adjusted.

## FP_TC_120_RANGE

Allows the range of a channel on a FieldPoint FP-TC-120 to be adjusted.

## FP_THERMOCOUPLE_TYPE

Allows adjustment of the thermocouple type of a channel on an FP-TC-120 module.

## HI_HI_OUT_D

Discrete output indicating state of Hi-Hi alarm condition.

## HI_OUT_D

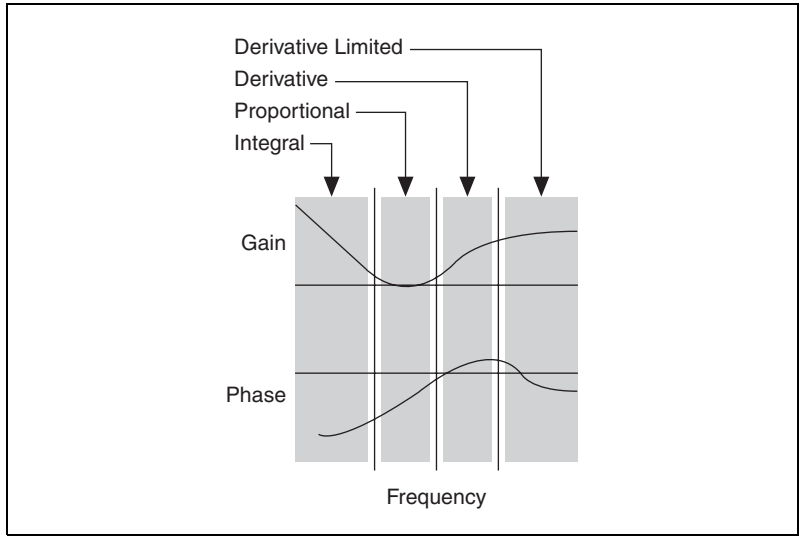
Discrete output indicating state of Hi alarm condition.

## INIT_STATUS

The initialization status of the block (used for debugging).

## LAMBDA

The frequency response of a PID controller is dominated by the Proportional, Integral, and Derivative modes in the three frequency regions shown in the following figure. To limit excessive amplification at high frequencies, a filter is used to create a "Derivative Limited" region. The ratio of the derivative term to the filter time constant is defined as lambda.




## LAST_BLOCK_EVENT

The last logged event detected by the block. Table C-7 lists descriptions of the block events. This parameter is useful for debugging configuration errors because it points to the parameter in error. For example, if an AI block is in `OOS` mode because `L_TYPE` is not set, the `MGS` would say `Linearization Type Uninitialized`.

**Table C-7.** Block Events

| Event | Description |
|---|---|
| CLASS | The type of event detected.<br>**Configuration Error**: An error has been detected in the configuration of the block.This is usually due to an uninitialized static parameter. The block updates its TARGET mode to Out of Service and posts a block alarm.<br>**Operational Warning**: The block detected a non-critical event. The block continues to execute normally.<br>**Operational Error**: The block detected a critical event. The block continues to execute in a higher priority mode.<br>**Internal Error**: The firmware detected an internal error. |
| MSG | A message containing specific details describing the event. |
| BLOCK_IDX | The index of the block causing the event. |
| PARAM_IDX | The index of the parameter causing the event. |
| TARGET_MODE | The TARGET mode of the block when the event was detected. |
| ACTUAL_MODE | The ACTUAL mode of the block when the event was detected. |
| TIME_STAMP | Time when the error was detected. |

## LAST_RUN_ERROR

The last run error detected by the block (used for debugging).

## LO_LO_OUT_D

Discrete output indicating state of Lo-Lo alarm condition.

## LO_OUT_D

Discrete output indicating state of Lo alarm condition.

## LTYPE_DOMAIN_INDEX

Virtual field device index of the domain associated with expression in this AI block.

## NVM_LIFE

Remaining useful life of non-volatile memory (NVM).

## OP_CMD_CXO

The lowest level priority input. This can be used to allow the operator to activate interlock behavior with a write from the host application.

**Table C-8.** Command Parameters

| Value | Description |
|---|---|
| Close | When this flag is set, the output of the block is forced to `Discret_State_0`. This interlock is overridden by every other interlock. If both `OP_CMD_CXO.Close` and `OP_CMD_CXO.Open` are set, they are both considered to be clear. |
| Open | When this flag is set, the output of the block is forced to `Discret_State_1`. This interlock is overridden by every other interlock. If both `OP_CMD_CXO.Close` and `OP_CMD_CXO.Open` are set, they are both considered to be clear. |
| Stop | Unused in FieldPoint. |
| Enable 1, 2, 3 | Unused in FieldPoint. |

## RUN_STATUS

The current run status of the block (used for debugging).

## RUN_TIME

The time the block took to run (including overhead).

## SAFEGUARD_CL

An interlock input. When in `Discret_State_1`, the output of the block is forced closed (`Discret_State_0`). This interlock input has priority over all other interlock inputs.

## SAFEGUARD_OP

An interlock input. When in `Discret_State_1`, the output of the block is forced open (`Discret_State_1`). This interlock input has priority over all other interlock inputs except `SAFEGUARD_CL`.

## SUPPORTED_MODES

The modes supported by the block.

## VERSION_INFORMATION

The revision of the firmware currently in use by the FP-3000. This parameter, present in the resource block, also contains the version numbers of the FOUNDATION Fieldbus specification documents used in the design of the FP-3000.

# D

# Advanced Function Block Behavior

This appendix explains advanced features of function blocks that are unnecessary to establish simple control strategies. Use this information to diagnose problems in control strategies and to develop systems that implement supervisory control by a host computer.

## Cascade Initialization

FOUNDATION Fieldbus provides a protocol called Cascade Initialization that allows a control function block to smoothly transition from `Man` to `Auto` mode. Cascade Initialization allows the PID algorithm to know the current setpoint of the AO block to balance the actual setpoint with the control's setpoint over time. Cascade Initialization is also used to prevent windup in the PID.

### Parameter Connections for Cascade Initialization

Cascade initialization takes place between two blocks: an upstream controlling block, and a downstream controlled block. In a PID loop, the upstream block is the PID block, and the downstream block is the AO block. In the case of cascaded PID blocks, the upstream PID feeds a setpoint into a second PID that is acting as the downstream block. In both cases, the parameter connections are the same. The output (`OUT`) parameter of the upstream block is connected to the cascade input (`CAS_IN`) parameter of the downstream block. This connection controls the setpoint of the downstream block. To allow the upstream block to determine the current setpoint of the downstream block, you must also connect the backward calculation output (`BKCAL_OUT`) parameter of the downstream block with the backward calculation input (`BKCAL_IN`) of the upstream block. The connections are shown in Figure D-1.

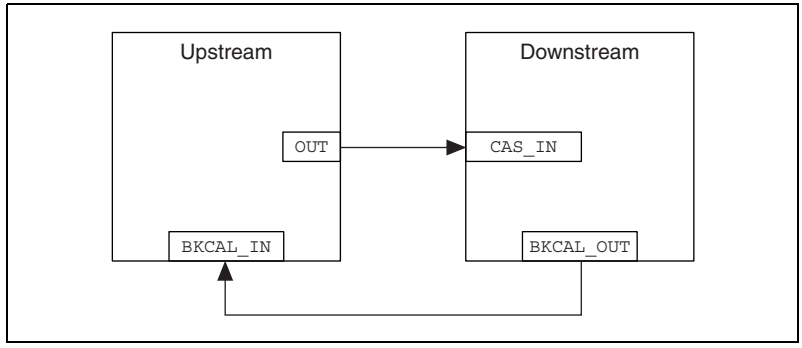



**Figure D-1.** Parameter Connections for Cascade Initialization

## Mode and Status Behavior during Cascade Initialization

Cascade initialization is arbitrated through the status of the backward calculation path from the downstream block and the forward calculation path on the upstream block. If the upstream block publishes a status of `Good, Non-Cascade`, it does not support cascade initialization, and the lower block immediately transitions into a Cascade mode. This happens in the case where an Analog Input (AI) block is acting as the upstream block for an Analog Output (AO) block. Since the AI block does not have a back calculation input, it does not support cascade initialization.

If the upstream block does support cascade initialization, it publishes a status of `Good, Cascade` on its forward calculation output. This signals to the downstream block to begin the cascade initialization process as soon as it is able. If the downstream block is unable to begin cascaded control, it publishes a status of `Good Cascade, Not Invited` on its backward calculation output. This signals to the upstream block that the control path from the downstream block to the process has been broken. As soon as the ability to begin control is established, the downstream block publishes a status of `Good Cascade, Initialization Request` on its backward calculation output. This signals to the upstream block that it should initialize itself for cascade control. While the initialization request status is active, the downstream block is also publishing its current setpoint to the upstream block. This enables the upstream block to prepare for a smooth transfer to automatic control. While the upstream block is initializing itself for automatic control, it enters an actual mode of Initialization Manual (`IMan`). When it is ready to begin control, it publishes a status of `Good Cascade, Initialization Acknowledge` to signal that it is beginning cascade control. The lower block then enters Cascade mode.

To prevent windup, the control loop needs to be aware when it is unable to control the process. If the downstream block can no longer control the process, it reports a status of Bad to the upstream block. This breaks the cascade until automatic control can be resumed, in which case cascade initialization takes place again.

## Remote Cascades

If a host application (rather than another block) provides the setpoint of a block, FOUNDATION Fieldbus provides the Remote Cascade mode. The remote cascade mode is equivalent to Cascade mode, except that the cascade input parameter is RCAS_IN instead of CAS_IN, and the back calculation output is RCAS_OUT instead of BKCAL_OUT. Unlike CAS_IN and BKCAL_OUT, which are input/output parameters, RCAS_IN and RCAS_OUT are contained parameters and can only be written by a host application. To allow the controlled block to enter Remote Cascade mode, the host application must act as the upstream block in the cascade initialization and implement the status handling described above.

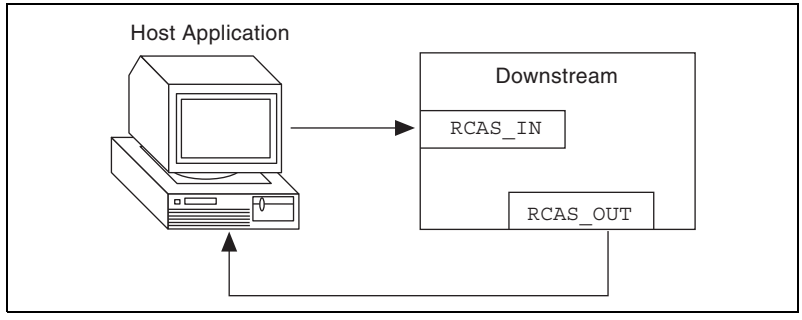



**Figure D-2.** Remote Cascade Model

There is a second remote mode in which a cascade must be initialized: Remote Output (ROut mode). Unlike RCas mode, where the block setpoint is set remotely, in ROut mode the block output is set by the host application. The back calculation output for ROut mode is ROUT_OUT, and the cascade input is ROUT_IN. Otherwise, cascade initialization proceeds normally.

## Bypassing Cascade Initialization

If cascade initialization is unnecessary, you can bypass it by leaving the back calculation input of the upstream block unwired. By manually writing a status of Good, Non-cascade to the back calculation input, the block bypasses cascade initialization and immediately transitions to automatic

control. The upstream block signals to the downstream block that cascade initialization has been bypassed by setting its output to a status of `Good, Non-cascade`.

# Fault State and Mode Shedding

To allow for safe control of a FOUNDATION, even in the event the input sensors or control algorithms fail, FOUNDATION Fieldbus provides fault state and mode shedding. Fault state is used when an output block is in Cascade mode. Mode shedding is used in Remote or Remote Cascade modes.

## Fault State

FOUNDATION Fieldbus output function blocks (AO, DO, and CDO) support a faultstate to deal with the case where the control of the output block has been lost while the block is in Cascade mode. If the block is in Cascade mode, and `CAS_IN` has a bad quality for longer than the time specified in the `FSTATE_TIME` parameter, the block enters faultstate. If the `Faultstate Use Value` option is set in the `IO_OPTS` parameter, the block uses the value in `FSTATE_VAL` as its output value. If the option is not set, it holds the value it had at the time the input went bad until the cascade can be reestablished.

## Mode Shedding

For remote modes in which the cascade input is periodically written by a host application, FOUNDATION Fieldbus provides mode shedding to handle failure of the host application. For the `RCas` mode and `ROut` modes, there are two timeout parameters in the resource block: `SHED_RCAS` and `SHED_ROUT`. If the block is in a remote mode and the block input being used (`RCAS_IN` or `ROUT_IN`) is not updated within the timeout period, the block sheds to a higher priority mode. The action taken by the block when mode shedding occurs is defined by the `SHED_OPT` parameter. The shed condition ends when the host writes the remote input parameter. If a normal return shed option is selected, the block attempts to enter the original remote mode. If a no return shed option is selected, the target mode is changed at the time the block sheds mode, and the block does not attempt to enter the original lower priority mode when the shed condition ends.

# E

# Specifications

This appendix describes the specifications of the FP-3000 network module.

All FieldPoint network modules undergo extensive testing for operating under rugged environmental conditions that exist in industrial applications. FieldPoint network modules are designed and tested for immunity and susceptibility, as well as for emissions.

## Network

FP-3000 ..... FOUNDATION Fieldbus H1

Integrity ..... Checksum

Power supply range ..... 11 to 30 VDC

Power consumption ..... 6 watt + 1.15 * ∑(*I/O Module Consumption*)

Maximum terminal bases per bank ..... 9

Maximum number of banks per Fieldbus link (without repeaters) ..... 32

## Environment

Operating temperature ..... –40 to +60 °C

Storage temperature ..... –55 to +100 °C

Relative humidity ..... 5% to 90% noncondensing

## Compliance

Electrical safety ..... designed to meet IEC 1010

EMI emissions/immunity ..... CISPR 11

# F

# Troubleshooting

The FP-3000 is a powerful and highly flexible tool you can use to solve distributed I/O and control problems. Because of this flexibility, you might encounter problems getting the FP-3000 to perform the functions you want. This chapter helps you diagnose and solve common problems that you might encounter using the FP-3000. Problems you might encounter with the FP-3000 include Fieldbus communication problems, I/O module problems, and software configuration problems.

## Setting Device Tag and Fieldbus Network Address

All Fieldbus devices need a unique Fieldbus network address and a unique device tag before they can be fully operational. The FP-3000 ships without a device tag and only a default network address. A default network address is a network address used temporarily to allow a device tag to be written by host configuration software. Before a permanent network address can be assigned, the device must be assigned a device tag. If you are using the NI-FBUS Configurator, these steps happen automatically when the NI-FBUS Configurator first "sees" the FP-3000 on the bus.

When the NI-FBUS Configurator finds an FP-3000 at a default Fieldbus network address, it creates and assigns it a unique device tag based on FP-3000 and the device's serial number. When the tag has been assigned, it then assigns the FP-3000 a permanent network address. (The address is permanent only in the sense that it is the address where the device is expected to stay. You can change the address.)

## Fieldbus Communication Problems

The interface between the host computer and the FP-3000 is a FOUNDATION Fieldbus network. The network allows you to connect multiple devices (such as FP-3000 network modules or other Fieldbus transmitters) and hosts together, with each device or host having a unique Fieldbus network address and a unique device tag. A link master device on the Fieldbus constantly polls (probes) empty network addresses to check for new devices. A number of problems can arise related to this networking scheme.

Table F-1 describes common Fieldbus communication problems and possible solutions.

**Table F-1.** Fieldbus Communication Problems

| Problem | Solutions |
|---|---|
| FP-3000 is not visible on the Fieldbus from host configuration software. | • Is the power LED on? If not, check your power supply and DC power wiring to the FP-3000.<br>• Is the Fieldbus network LED on? If not, your Fieldbus wiring could be bad.<br>• Check the switches on the back of the FP-3000. Make sure the **Reset** switch is not On. Refer to Appendix A, *Configuring the FP-3000*, to view a diagram of the configuration switches.<br>• Is your host configuration software probing the Fieldbus network address of the FP-3000? If not, the FP-3000 will not be able to get on the network. Check your host configuration software documentation for how to make the host probe more or all network addresses. For NI-FBUS, one way to do this is to use the **Interface Config** program, select **Advanced settings** for the interface port, and set **Num of unpolled nodes** to **0**, then restart NI-FBUS. Refer to the *NI-FBUS Configurator User Manual* for other options and a discussion of the ramifications of setting **Num of Unpolled Nodes** to 0. |
| FP-3000 **NETWORK** LED is red (steady or flashing). | Check your Fieldbus wiring for short circuits or other electrical problems. |
| FP-3000 is visible on the Fieldbus from a host, but no blocks show up. | If you are running NI-FBUS Configurator, and this is your first startup of FP-3000, wait a couple of minutes for the NI-FBUS Configurator to complete setting the device and address tag of the FP-3000. When the process is complete, the resource block will show up and you can instantiate (create) other blocks either using the autoconfiguration function or manually.<br>Otherwise, the FP-3000 might be stuck at a default Fieldbus network address with no tag. Consult your host configuration software documentation on how to set the address and device tag. |

**Table F-1.** Fieldbus Communication Problems (Continued)

| Problem | Solutions |
|---|---|
| All the blocks and configuration information are lost on a power cycle. | Make sure the Reset switch at the back of the FP-3000 is set correctly. Refer to Appendix A, *Configuring the FP-3000*, for details. |
| FP-3000 does not execute the communication schedule when it becomes LAS. | You must download a schedule to the FP-3000. If you are using the NI-FBUS Configurator, make sure FP-3000 is in the list of devices to receive a schedule as described in the *NI-FBUS Configurator User Manual*. |

# I/O Module Problems

FP-3000 offers diagnostic capabilities to help find problems with the I/O modules you have plugged in.

**Table F-2.** I/O Module Problems

| Problem | Solutions |
|---|---|
| The green **READY** LED does not light when the module is plugged in. | Make sure the module is firmly seated. Also, make sure the terminal base is firmly attached to the terminal base on its left. Look for bent pins on the bottom of the module or between terminal bases (remember to power down the bank before removing terminal bases). |
| The red LED is lit on an I/O channel. | • TC module: The thermocouple is not connected or is broken.<br>• AI module: The input wires are not connected or are broken.<br>• AO module: The module cannot source as much current as the FP-3000 is requesting. This might be because the output wires are not connected or are broken, you have not powered the outputs, or there could be some other electrical problem. |
| The power LED does not light when the module is plugged in | • Make sure the module is firmly seated.<br>• Make sure you are not exceeding the amount of current your power supply can provide. Refer to *Bank Power Requirements* in Chapter 2, *Installation and Configuration*, for information on calculating power requirements.<br>• Try removing and/or swapping modules and terminal bases to see if the problem can be narrowed down to a particular terminal base or module. If so, that unit might need to be replaced. Contact National Instruments for help. |

# Software Configuration Problems

The FP-3000 software consists of a number of function blocks that you can instantiate, or create, from host configuration software. Each function block represents an input channel, an output channel, or some control behavior. In addition, each device contains a Resource Block which allows you to configure certain overall characteristics for the device. All of these blocks contain parameters, which are values that you can set to configure things like channel ranges, scaling to engineering units, and failsafe behavior.

Some problems are generic to all blocks, and some problems occur only with certain types of blocks. Table F-3 lists problems that can occur in many types of blocks. Table F-4 lists problems that can occur only in the resource block.

**Table F-3.** Generic Software Configuration Problems

| Problem | Solutions |
|---|---|
| Block will not leave OOS (Out of Service) mode, and BLOCK_ERR does not report any errors. | • Make sure you have written a non-OOS mode to the TARGET mode of the block.<br>• If this is a function block, make sure you have scheduled the block to execute. Refer to your configuration software documentation for information about how to download a configuration, including a schedule, to a device. Function blocks must be scheduled to change modes. If you are using the NI-FBUS Configurator, you can also refer to Chapter 3, *Example Applications*, for information on downloading a configuration.<br>• The resource block might be OOS. This would force all function blocks in that device into OOS mode. Set the MODE_BLK.TARGET parameter in the resource block to Auto, and make sure its ACTUAL mode changes to Auto. You do not need to schedule the resource block to change its mode. |

**Table F-3.** Generic Software Configuration Problems (Continued)

| Problem | Solutions |
|---|---|
| Block will not leave OOS (Out of Service) mode, and BLOCK_ERR parameter reads Block Configuration Error. | • Make sure you have written a non-OOS mode to the TARGET mode of the block.<br>• Look at the MSG field of the parameter LAST_BLOCK_EVENT on the block in question. This contains details on why the block cannot leave OOS mode. If this field is blank, the block might not be scheduled to execute; function blocks must be scheduled to change modes. Refer to your configuration software documentation for information about how to download a configuration, including a schedule, to a device. If you are using the NI-FBUS Configurator, you can also refer to Chapter 3, *Example Applications*, for information on downloading a configuration.<br>• If this field has a message, the message tells you which parameter of the block is misconfigured. Look up the correct use of the parameter in the Chapter 4, *Block Reference*, Appendix B, *Fieldbus Parameters*, or your device documentation, and reconfigure the parameter. |
| Block leaves OOS mode but will not go into the exact TARGET mode. | Your block is correctly configured, but some run-time condition is keeping the block from reaching its target mode.<br>• If your block is connected to other function blocks as part of a function block application, check the status of the input parameters. If you are using the NI-FBUS Configurator, either look at the appropriate parameter (IN, CAS_IN, BKCAL_IN, etc.) in the block configuration window (with periodic updates on) or use the Monitor functionality as described in the *NI-FBUS Configurator User Manual*. If any of these have a quality of Bad or Uncertain, examine the blocks the inputs came from to determine the problem.<br>• If this block is not connected to other function blocks, but is operating standalone, check the BLOCK_ERR parameter to determine the problem. If BLOCK_ERR reports Input Failure or Output Failure, and this is an AI, AO, DI, DO, or CDO block, there is likely a problem with the I/O channel. For more information, refer to the section *I/O Module Problems*. |

**Table F-3.** Generic Software Configuration Problems (Continued)

| Problem | Solutions |
|---|---|
| The actual mode of a block switches values. | A block, such as a PID, may switch modes between `Iman` and `Auto`. This means that there is a communications problem between the PID and the downstream AO block.<br>• Check the schedule. The execution of the PID, AO, and the communication between them might be scheduled too close to each other. Space these events further apart and re-download the configuration.<br>• Check if the stale limit is set correctly for the loop. The stale limit defines how many old values to accept before changing `Status` to `Bad`. If the block that is subscribing the data is executing at twice the rate of a block that is publishing data, the stale limit must at least be set to 2. |
| PID block output seems incorrect. | Make sure that you have set the value of the tuning parameters like `RATE`, `GAIN` and `RESET` correctly.<br>Typical values for `GAIN`, `RESET`, and `RATE` are:<br>`GAIN` `RESET` `RATE`<br>Pressure control 1.2 3.33 0.8<br>Temperature control 3.0 25.0 10.0<br>Flow control 0.33 1.11 0.0<br>Level control 1.9 16.67 2.7 |
| Cannot access or view the parameters added by National Instruments. | Make sure that the Device Descriptions are installed in the correct location. Refer to the section *Import Device Descriptions* in Chapter 2, *Installation and Configuration*. |
| Cannot change the values of certain parameters. | Many configuration parameters of function blocks can be changed only when the block is in `OOS` mode. Set the mode to `OOS` and then change the configuration parameters. |

**Table F-4.** Resource Block Configuration Problems

| Problem | Action |
|---|---|
| Cannot bring resource block into `Auto` mode. | Make sure the Reset switch is disabled. You cannot bring the Resource Block into `Auto` with the Reset switch On (this prevents you from losing your entire configuration if you inadvertently leave the Reset switch On). |
| Cannot set the `WRITE_LOCK` parameter. | Make sure the Write Lock switch is Off. Changing the `WRITE_LOCK` parameter is possible only when the Write Lock switch is Off. If you want to disallow configuration writes to the device, set the Write Lock switch to On, then write `Locked` to this parameter. |

# Common Questions

**What can I do that would cause big problems with my FP-3000?**

Turn off the power while the FP-3000 is writing its non-volatile memory. The FP-3000 writes its non-volatile memory when you are downloading a configuration or writing most parameters. The LED will be solid green after the write starts. Just before the light goes out, the memory is being updated. If the power is turned off during this time, the FP-3000 will lose all blocks and your configuration. If this happens, you need to redownload the configuration.

**I am trying to control a PID loop within the FP-3000. On occasion, the Lookout operator may want to override the output of the PID with a manual output setpoint. How can I implement this?**

1. Double-click on your PID block in the NI-FBUS Configurator to open the block configuration window.
2. On the **Options** tab, click to the right of the `CONTROL_OPTS` parameter. Check the box next to `TRACK ENABLE`. This gives you the option of using a direct value rather than the PID-calculated value. Whether you are using the calculated PID output or just tracking an operator-set value at any given time is determined by the state of `TRK_IN_D`. If it is On, you are tracking the `TRK_VAL`.
3. On the **Scaling** tab, set the quality of the `TRK_IN_D` and `TRK_VAL` to `Good_NonCascade`. `TRK_IN_D` turns tracking on and off. For normal PID operation, you want this value to be zero. To send the `TRK_VAL` directly to the PID output, set this to 1 (On). Connect a switch for the

operator in Lookout to the TRK_IN_D.VALUE of the PID object. Connect the Pot for the operator in Lookout to TRK_VAL.VALUE.

**How long does it take a function block to run on the FP-3000?**

This depends on the function block. The following are approximate maximum execution times in firmware revision 1.

| | |
|---|---|
| AI block | 6 ms |
| DI block | 5 ms |
| DO block | 5 ms |
| AO block | 7 ms |

The maximum execution time is a function block parameter. You can read it from any function block. To view execution times using the NI-FBUS Configurator, select **View»Preferences**. Select the **Block View** tab. Check the **Show Block Information** checkbox. Now when you double-click on a function block, a new tab called **Block Information** will appear. Look at the parameter EXECUTION_TIME on the **Block Information** tab for the individual function block. If the number is a hexadecimal number, you can change it to decimal by right-clicking on it and unchecking Hexadecimal Data. This decimal value of this parameter is the execution time in 1/32 ms. (i.e., dividing this number by 32 will give the maximum execution time in milliseconds).

**How Can I Determine FP-3000 Execution Time?**

If you are running a PID control loop that is entirely contained on the FP-3000 (i.e., the AI, PID, and AO blocks are all created on the FP-3000) with no interaction with an HMI (Human-Machine Interface) software package (such as Lookout or BridgeVIEW), then you can expect a minimum 50 Hz control loop. Add the worst-case function block execution time for each function block in the loop to this minimum. A PID control loop has one AI, one PID, and one AO block. Together they take 20 ms. (The maximum execution time of a function block is a function block parameter you can find in the NI-FBUS Configurator under the Block Information tab for the individual function block.)

For example, if you want to run 10 PID control loops at 20 ms each, this will be a total of 200 ms of execution time (one-fifth of a second). You could run 10 loops as fast as 5 times per second. Be aware that the Expression block takes significantly longer to execute than other function blocks and is dependent on the downloaded expression.

### How Many PID Loop Iterations per Second Can the FP-3000 Run?

The FP-3000 can run approximately 50-60 PID loop iterations per second. You can choose how to distribute these iterations—running one PID loop 50 times per second or 49 PID loops once per second (one loop belongs to the mandatory resource block). You can also choose a point between these two extremes.

**Note** When deciding how fast to run PID loops, keep in mind the update rate of the I/O modules. For example, the AI-100 has an update rate of 2.8 ms, but the AI-110 has an update rate of 170 ms to 1500 ms. It does not make sense to run a PID loop at a faster rate than the update rate. The update rate of the individual I/O modules is documented in the National Instruments catalog and in the operating instructions of each module.

### How Many Channels Per Second Can I Read?

Using OPC with NI-FBUS version 2.3.5—Expect (as a best case) to be able to read 400 analog or 1000 discrete FP-3000 channels per second. Note that this is divided among all the devices on the bus. Performing regular reads with NI-FBUS version 2.3.5 are the same as version 2.3.

NI-FBUS version 2.3—Expect roughly four reads per second from typical third-party transmitters. Expect (as a best case) to be able to read 30 FP-3000 channels per second. (The FP-3000 is a very fast Fieldbus device.)

### How Can I Determine Bandwidth of the Bus?

The bandwidth of the bus affects communications between devices, including the interface board on the host machine. This is important if you are running a loop with function blocks from different devices. For example, a PID loop with the AI block on one device and the PID and AO blocks on another will not run as fast as a PID control loop contained entirely within one device. This is due to the time necessary for the data from the AI to cross the bus. The scheduler in the NI-FBUS Configurator software will take bus traffic into account when scheduling function block execution. The amount of information that can cross the bus in a given period of time depends on several factors including:

- Whether you have adjusted the bus timing parameters to accommodate slower devices
- How many devices are connected to the bus
- Whether you are using OPC or traditional reads and writes and how often you have host communications

Note that if you want to monitor values with an HMI package on the host machine, you may have to slow down the execution rate. This is because HMI communications are considered unscheduled communications by the specification. Scheduled, publisher-subscriber Fieldbus communications are of higher priority than nonscheduled HMI communications. If you do not leave enough unscheduled time in your macrocycle, the HMI communications will never get through.

**How Much Unscheduled Time Should I Allow?**

Start with 30-40% of the total schedule (macrocycle) available for unscheduled communications. With slow devices (NOT the FP-3000) you can often include a large part of the time that function blocks are executing in your calculation of this percentage. This is because on slow devices only a small part of the execution time involves broadcasting information over the bus. During the rest of the execution time, there is no scheduled communication on the bus. With the FP-3000, the function block execution times are so short that there is usually not enough time to send an unscheduled message between the scheduled broadcasts of the function blocks. In this case, you need to include extra time in your macrocycle for unscheduled communication.

# Problems Using Manufacturer-Defined Features

NI-FBUS uses identifying information in the actual device to locate the device description for the device. The identifying information includes four resource block parameters: `MANUFAC_ID`, `DEV_TYPE`, `DEV_REV`, and `DD_REV`. If the identifying information is incorrect, NI-FBUS will not be able to locate the device description for the device. When it has located the device description, NI-FBUS matches the block types in the device description with the actual blocks in the device by using the Item ID of the block characteristics record.

If the blocks in the device do not match the blocks in the description, or if there is no appropriate device description for the manufacturer, device type, device revision and device description revision being returned by the device, then there is a device description mismatch. In either case, NI-FBUS uses only the standard dictionary (`nifb.dct`) and you will be unable to use any manufacturer-supplied functionality.

These parameters can be read from the device's resource block.
The following procedure will help you troubleshoot a `DD_SIZE_MISMATCH_ERROR` by finding out if there is a device description available on your computer that matches what your device expects.

Follow these steps to use the NI-FBUS Dialog utility to check device description files.

1. Start the NIFB process. Wait until the process has finished initializing.
2. Select **Start»Programs»National Instruments FBUS»NI-FBUS Dialog**.
3. Right-click on **Open Descriptors** and choose **Expand All**.
4. After the expansion is complete, click on **Cancel** to close the Expand All window.
5. Right-click on the resource block for your device (it should be under **Open Descriptors»Session»Interface Name»Device Name»VFD Name»Resource Block Name**). Select **Read Object**.
6. Select the **Read by Name** radio button and enter *MANUFAC_ID* as the name. Click on the **Read** button. Write down the hexidecimal number found in parenthesis (0xnumber) in Table F-5.
7. Repeat step 6 for the name *DEV_TYPE*.
8. Repeat step 6 for the name *DEV_REV*.
9. Repeat step 6 for the name *DD_REV*.
10. Repeat steps 5–9 for each device, then close the NI-FBUS Dialog utility.

**Table F-5.** Device Names

| Resource Block Parameter | Name |
|---|---|
| MANUFAC_ID | |
| DEV_TYPE | |
| DEV_REV | |
| DD_REV | |

11. In the Interface Configuration utility, click on the `DD Info` button. Write down the base directory specified for device descriptions. Close the Interface Configuration utility.
12. Use Windows Explorer to view the contents of the base directory specified in the Interface Configuration utility. The Fieldbus specification defines the directory hierarchy for storing device descriptions. There is a different subdirectory for each device manufacturer. Under the base directory, you should see a directory with the number from step 6 for the first device.

13. Under the appropriate manufacturer directory, there is a directory for each device type that you have from that manufacturer. Check to make sure that you see a directory with the number from step 7.
14. Under the appropriate device type directory, there are the individual device descriptions. The device description file name is a combination of the device revision (the number from step 8) and the device description revision (the number from step 9). The device revision is the first two digits, and the device description revision is the second two digits. For example, if your number from step 8 was 2 and from step 9 was 1, you should see files called 0201.ffo and 0201.sym. Device descriptions are backwards compatible. This means that instead of seeing 0201, you might see 0202. This is allowed by the Fieldbus specification. Also, having additional files in this directory is not a problem. The NI-FBUS Configurator will use the most recent device description revision for a given device revision. If you do not have the appropriate `.ffo` and `.sym` files, you must obtain them from the device manufacturer. Be sure to properly import them by clicking on **DD Info** and using the **Import DD** button in the Interface Configuration utility.
15. Repeat steps 12–14 for each device.

The second cause for this problem is when the contents of the file do not accurately describe the device characteristics, even if the device identification information matches the file identification information. This problem is caused when a device manufacturer makes a change to the firmware of the device without incrementing the device revision, in violation of the FOUNDATION Fieldbus recommendation. If this is the case, you must contact your device manufacturer for a resolution.

# G

# Technical Support Resources

## Web Support

National Instruments Web support is your first stop for help in solving installation, configuration, and application problems and questions. Online problem-solving and diagnostic resources include frequently asked questions, knowledge bases, product-specific troubleshooting wizards, manuals, drivers, software updates, and more. Web support is available through the Technical Support section of `www.ni.com`

## NI Developer Zone

The NI Developer Zone at `ni.com/zone` is the essential resource for building measurement and automation systems. At the NI Developer Zone, you can easily access the latest example programs, system configurators, tutorials, technical news, as well as a community of developers ready to share their own techniques.

## Customer Education

National Instruments provides a number of alternatives to satisfy your training needs, from self-paced tutorials, videos, and interactive CDs to instructor-led hands-on courses at locations around the world. Visit the Customer Education section of `www.ni.com` for online course schedules, syllabi, training centers, and class registration.

## System Integration

If you have time constraints, limited in-house technical resources, or other dilemmas, you may prefer to employ consulting or system integration services. You can rely on the expertise available through our worldwide network of Alliance Program members. To find out more about our Alliance system integration solutions, visit the System Integration section of `www.ni.com`

# Worldwide Support

National Instruments has offices located around the world to help address your support needs. You can access our branch office Web sites from the Worldwide Offices section of `www.ni.com`. Branch office web sites provide up-to-date contact information, support phone numbers, e-mail addresses, and current events.

If you have searched the technical support resources on our Web site and still cannot find the answers you need, contact your local office or National Instruments corporate. Phone numbers for our worldwide offices are listed at the front of this manual.

# Glossary

| Prefix | Meanings | Value |
|---|---|---|
| m- | milli- | $10^{-3}$ |
| M- | mega- | $10^{6}$ |

## Numbers

| | |
|---|---|
| 4-20 mA system | Traditional control system in which a computer or control unit provides control for a network of devices controlled by 4-20 mA signals. |

## A

| | |
|---|---|
| A | Amperes. |
| Actuator | A device that translates electrical signals into mechanical actions. |
| A/D | Analog-to-digital converter. |
| Address | Character code that identifies a specific location (or series of locations) in memory. |
| Administrative Function | An NI-FBUS function that deals with administrative tasks, such as returning descriptors and closing descriptors. |
| AI | Analog Input. |
| Alarm | A notification the NI-FBUS Communications Manager software sends when it detects that a block leaves or returns to a particular state. |
| Alarm condition | A notification that a Fieldbus device sends to another Fieldbus device or interface when it leaves or returns to a particular state. |
| Alert | An alarm or event. |
| Analog | A description of a continuously variable signal or a circuit or device designed to handle such signals. |
| AO | Analog Output. |

## B

| | |
|---|---|
| Bandwidth | The range of frequencies present in a signal, or the range of frequencies to which a measuring device can respond. |
| Bank | The combination of one FieldPoint network module and one or more terminal bases and I/O modules. |
| Basic device | A device that can communicate on the Fieldbus, but cannot become the LAS. |
| Bit string | A data type in the object description. |
| Block | A logical software unit that makes up one named copy of a block and the associated parameters its block type specifies. The values of the parameters persist from one invocation of the block to the next. It can be a resource block, transducer block, or function block residing within a virtual field device. |
| Boolean | Logical relational system having two values, each the opposite of the other, such as true and false or zero and one. |
| BridgeVIEW | A program for developing applications that require high channel count datalogging, as well as supervisory control of distributed systems. |
| Buffer | Temporary storage for acquired or generated data. |
| Bus | The group of conductors that interconnect individual circuitry in a computer. Typically, a bus is the expansion vehicle to which I/O or other devices are connected. Examples of PC busses are the ISA and PCI buses. |

## C

| | |
|---|---|
| C | Celsius. |
| Cable | A number of wires and shield in a single sheath. |
| Channel | A pin or wire lead to which you apply or from which you read the analog or digital signal. |
| Circuit | Interconnection of components to provide an electrical path between two or more components. |
| CISPR | International Special Committee On Radio Interference. |

| | |
|---|---|
| Contained parameter | A parameter that does not receive or send data and is contained within a function block. |
| Control loop | A set of connections between blocks used to perform a control algorithm. |
| Control strategy | *See* function block application. |
| Controller | An intelligent device (usually involving a CPU) that is capable of controlling other devices. |
| Current | The flow of electrons through a conductor. |

## D

| | |
|---|---|
| DC | Direct Current. |
| DD | *See* Device Description. |
| Descriptor | A number returned to the application by the NI-FBUS Communications Manager, used to specify a target for future NI-FBUS calls. |
| Device | A sensor, actuator, or control equipment attached to the Fieldbus. |
| Device Description | A machine-readable description of all the blocks and block parameters of a device. |
| Device ID | An identifier for a device that the manufacturer assigns. No two devices can have the same device ID. |
| Device tag | A name you assign to a Fieldbus device. |
| DI | Discrete Input. |
| Digital | Pertaining to data (signals) in the form of discrete (separate/pulse form) integral values. |
| Directory | A structure for organizing files into convenient groups. A directory is like an address showing where files are located. A directory can contain files or subdirectories of files. |
| Distributed control | Process control distributed among several devices connected by network. |
| DO | Discrete Output. |

# E

| | |
|---|---|
| Event | An occurrence on a device that causes a Fieldbus entity to send the Fieldbus event message. |

# F

| | |
|---|---|
| FBAP | *See* Function Block Application. |
| FF | FOUNDATION Fieldbus. |
| Field device | A Fieldbus device connected directly to a Fieldbus. |
| Fieldbus | An all-digital, two-way communication system that connects control systems to instrumentation. A process control local area network defined by ISA standard S50.02. |
| Fieldbus cable | Shielded, twisted pair cable made specifically for Fieldbus that has characteristics important for good signal transmission and are within the requirements of the Fieldbus standard. |
| Fieldbus Foundation | An organization that developed a Fieldbus network specifically based upon the work and principles of the ISA/IEC standards committees. |
| Fieldbus Network Address | Location of a board or device on the Fieldbus; the Fieldbus node address. |
| FOUNDATION Fieldbus specification | The communications network specification that the Fieldbus Foundation created. |
| FP-3000 | National Instruments network interface module for the FieldPoint I/O system. |
| Function block | A named block consisting of one or more input, output, and contained parameters. The block performs some control function as its algorithm. Function blocks are the core components you control a system with. The Fieldbus Foundation defines standard sets of function blocks. There are ten function blocks for the most basic control and I/O functions. Manufacturers can define their own function blocks. |
| Function Block Application | The block diagram that represents your control strategy. |
| Function Block Application Editor window | The middle window of the NI-FBUS Configurator where you create your block diagram. |

## G

| | |
|---|---|
| Ground | An intentional or accidental conducting path between an electrical system or circuit and the earth or some conducting body acting in place of the earth. A ground is often used as the common wiring point or reference in a circuit. |

## H

| | |
|---|---|
| H1 | The 31.25 kbit/second type of Fieldbus. |
| HMI | Human-Machine Interface. A graphical user interface for the process with supervisory control and data acquisition capability. |
| HotPnP | Hot Plug and Play. |
| Hz | Hertz. |

## I

| | |
|---|---|
| IEC | International Electrotechnical Commission. A technical standards committee which is at the same level as the ISO. |
| Index | An integer that the Fieldbus specification assigns to a Fieldbus object or a device that you can use to refer to the object. A value in the object dictionary used to refer to a single object. |
| Input parameter | A block parameter that receives data from another block. |
| I/O | Input/output. |
| ISA | Industry Standard Architecture. |
| Isolation | A type of signal conditioning in which you isolate the transducer signals from the computer for safety purposes. This protects you and your computer from large voltage spikes and makes sure the measurements from the devices are not affected by differences in ground potentials. |

## L

| | |
|---|---|
| LAS | *See* Link Active Scheduler. |
| LED | Light-emitting diode. |

| | |
|---|---|
| Link | A FOUNDATION Fieldbus network is made up of devices connected by a serial bus. This serial bus is called a link (also known as a segment). |
| Link Active Scheduler | The Fieldbus device that is currently controlling access to the Fieldbus. A device that is responsible for keeping a link operational. The LAS executes the link schedule, circulates tokens, distributes time, and probes for new devices. |
| Link master device | A device that is capable of becoming the LAS. |
| Linkage | A connection between function blocks. |
| Lookout | National Instruments Lookout is a full-featured object-based automation software system that delivers unparalleled power and ease of use in demanding industrial measurement and automation applications. |
| Loop | *See* control loop. |

## M

| | |
|---|---|
| Macrocycle | The least common multiple of all the loop times on a given link, or one iteration of a the process control loop. |
| Menu | An area accessible from the command bar that displays a subset of the possible command choices. In the NI-FBUS Configurator, refers to menus defined by the manufacturer for a given block. |
| Method | Methods describe operating procedures to guide a user through a sequence of actions. |
| mm | millimeter. |
| Mode | Type of communication. |

## N

| | |
|---|---|
| Network address | The Fieldbus network address of a device. |
| Nifb.exe | The NIFB process that must be running in the background for you to use your AT-FBUS or PCMCIA-FBUS interface to communicate between the board and the Fieldbus. |
| NI-FBUS API | The NI-FBUS Communications Manager. |

| | |
|---|---|
| NI-FBUS Communications Manager | Software shipped with National Instruments Fieldbus interfaces that lets you read and write values. It does not include configuration capabilities. |
| NI-FBUS Configurator | National Instruments Fieldbus configuration software. With it, you can set device addresses, clear devices, change modes, and read and write to the devices. |
| NI-FBUS Fieldbus Configuration System | *See* NI-FBUS Configurator. |
| NI-FBUS process | Process that must be running in the background for you to use your AT-FBUS or PCMCIA-FBUS interface to communicate between the board and the Fieldbus. |
| Node | Junction or branch point in a circuit. |
| Non-volatile memory | Memory that does not require electricity to hold data. |

## O

| | |
|---|---|
| Object | An element of an object dictionary. |
| OOS | Out of Service mode. |
| OPC | OLE for Process Control. |
| Output parameter | A block parameter that sends data to another block. |

## P

| | |
|---|---|
| Parameter | One of a set of network-visible values that makes up a function block. |
| PC | Personal Computer. |
| PCMCIA | Personal Computer Memory Card International Association. |
| PID | Proportional/Integral/Derivative. A common control function block algorithm that uses proportions, integrals, and derivatives in calculation. |
| Polarity | Term used to describe positive and negative charges. |
| Poll | To repeatedly inspect a variable or function block to acquire data. |
| Port | A communications connection on a computer or remote controller. |
| POST | Power-On Self Test. |

| | |
|---|---|
| Process variable | A common Fieldbus function block parameter representing some value in the process being controlled. |
| Program | A set of instructions the computer can follow, usually in a binary file format, such as a `.exe` file. |
| Publisher | A device that has at least one function block with its output value connected to the input of another device. |
| PV | Process Variable. |

## R

| | |
|---|---|
| RA | Ratio. |
| Repeater | Boost the signals to and from the further link. |
| Resistor | Component made of material that opposes flow of current and therefore has some value of resistance. |
| Resource block | A special block containing parameters that describe the operation of the device and general characteristics of a device, such as manufacturer and device name. Only one resource block per device is allowed. |

## S

| | |
|---|---|
| s | Seconds. |
| Scheduled communications | Communication that occurs at the same time during each control cycle. |
| Segment | *See* Link. |
| Sensor | A device that responds to a physical stimulus (heat, light, sound, pressure, motion, flow, and so on), and produces a corresponding electrical signal. |
| Server | Device that receives a message request. |
| Service | Services allow user applications to send messages to each other across the Fieldbus using a standard set of message formats. |
| Session | A communication path between an application and the NI-FBUS Communications Manager. |

| | |
|---|---|
| Signal | An extension of the IEEE 488.2 standard that defines a standard programming command set and syntax for device-specific operations. |
| Stale | Data that has not been updated for `stale_limit` number of macrocycles, where the stale limit is a parameter of the connection. |
| Subscriber | A device that has at least one function block with its input value connected to the output of another device. |

## T

| | |
|---|---|
| Tag | A name you can define for a block, virtual field device, or device. |
| Terminator | A device used to absorb the signal at the end of a wire. |
| Timeout | A period of time after which an error condition is raised if some event has not occurred. |
| Transducer block | A block that is an interface to the physical, sensing hardware in the device. It also performs the digitizing, filtering, and scaling conversions needed to present input data to function blocks, and converts output data from function blocks. Transducer blocks decouple the function blocks from the hardware details of a given device, allowing generic indication of function block input and output. Manufacturers can define their own transducer blocks. |
| Trend | A Fieldbus object that allows a device to sample a process variable periodically, then transmit a history of the values on the network. |
| Trunk | *See* Homerun. |

## U

| | |
|---|---|
| Unscheduled | Messages sent on the Fieldbus between transmissions of scheduled messages. |
| Upstream | Fewer network hops away from a backbone or hub. For example, a small ISP that connects to the Internet through a larger ISP that has their own connection to the backbone is downstream from the larger ISP, and the larger ISP is upstream from the smaller ISP. |

## V

| | |
|---|---|
| V | Volts |
| VDC | Volts Direct Current. |
| VFD | *See* Virtual Field Device. |
| Virtual Field Device | The virtual field device is a model for remotely viewing data described in the object dictionary. The services provided by the Fieldbus Messaging Specification allow you to read and write information about the object dictionary, read and write the data variables described in the object dictionary, and perform other activities such as uploading/downloading data and invoking programs inside a device. A model for remotely viewing data described in the object dictionary. |

## W

| | |
|---|---|
| Waveform | Multiple voltage readings taken at a specific sampling rate. |

# Index

## Numbers



## A



## B

## C

## D



## E

## F

## G



## H



## I

## L



## M

## N



## O



## P

## Q



## R



## S

## T

## U



## V



## W



## X