

Fieldbus

**NI-FBUS™
Communications
Manager User Manual
for Windows 95 and
Windows NT**

July 1997 Edition
Part Number 321287B-01

© Copyright 1996, 1997 National Instruments Corporation.
All Rights Reserved.



Internet Support

support@natinst.com

E-mail: info@natinst.com

FTP Site: ftp.natinst.com

Web Address: <http://www.natinst.com>



Bulletin Board Support

BBS United States: (512) 794-5422

BBS United Kingdom: 01635 551422

BBS France: 01 48 65 15 59



Fax-on-Demand Support

(512) 418-1111



Telephone Support (U.S.)

Tel: (512) 795-8248

Fax: (512) 794-5678



International Offices

Australia 03 9879 5166, Austria 0662 45 79 90 0, Belgium 02 757 00 20,
Canada (Ontario) 905 785 0085, Canada (Québec) 514 694 8521, Denmark 45 76 26 00,
Finland 09 725 725 11, France 01 48 14 24 24, Germany 089 741 31 30,
Hong Kong 2645 3186, Israel 03 5734815, Italy 02 413091, Japan 03 5472 2970,
Korea 02 596 7456, Mexico 5 520 2635, Netherlands 0348 433466,
Norway 32 84 84 00, Singapore 2265886, Spain 91 640 0085, Sweden 08 730 49 70,
Switzerland 056 200 51 51, Taiwan 02 377 1200, United Kingdom 01635 523545

National Instruments Corporate Headquarters

6504 Bridge Point Parkway Austin, TX 78730-5039 Tel: (512) 794-0100-0100

Important Information

Warranty

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

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

National Instruments believes that the information in this manual 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 THEREFORE PAID BY THE CUSTOMER. NATIONAL INSTRUMENTS WILL NOT BE LIABLE FOR DAMAGES RESULTING FROM LOSS OF DATA, PROFITS, USE OF PRODUCTS, OR INCIDENTAL OR CONSEQUENTIAL DAMAGES, EVEN IF ADVISED OF THE POSSIBILITY THEREOF. This limitation of the liability of National Instruments will apply regardless of the form of action, whether in contract or tort, including negligence. Any action against National Instruments must be brought within one year after the cause of action accrues. National Instruments shall not be liable for any delay in performance due to causes beyond its reasonable control. The warranty provided herein does not cover damages, defects, malfunctions, or service failures caused by owner's failure to follow the National Instruments installation, operation, or maintenance instructions; owner's modification of the product; owner's abuse, misuse, or negligent acts; and power failure or surges, fire, flood, accident, actions of third parties, or other events outside reasonable control.

Copyright

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

Trademarks

NI-FBUS™ is a trademark of National Instruments Corporation.

Product and company names listed are trademarks or trade names of their respective companies.

WARNING REGARDING MEDICAL AND CLINICAL USE OF NATIONAL INSTRUMENTS PRODUCTS

National Instruments products are not designed with components and testing intended to ensure a level of reliability suitable for use in treatment and diagnosis of humans. Applications of National Instruments products involving medical or clinical treatment can create a potential for accidental injury caused by product failure, or by errors on the part of the user or application designer. Any use or application of National Instruments products for or involving medical or clinical treatment must be performed by properly trained and qualified medical personnel, and all traditional medical safeguards, equipment, and procedures that are appropriate in the particular situation to prevent serious injury or death should always continue to be used when National Instruments products are being used. National Instruments products are NOT intended to be a substitute for any form of established process, procedure, or equipment used to monitor or safeguard human health and safety in medical or clinical treatment.

*Table
of
Contents*

About This Manual

How to Use the Manual Set.....	ix
Organization of This Manual.....	x
Conventions Used in This Manual	xi
Related Documentation	xi
Customer Communication	xii

Chapter 1 Introduction

Fieldbus Overview.....	1-1
What Is Fieldbus?.....	1-1
Benefits of Fieldbus	1-1
The Fieldbus Foundation.....	1-2
FOUNDATION Fieldbus Technology.....	1-2
FOUNDATION Fieldbus Layers.....	1-3
The Physical Layer	1-3
The Communication Stack	1-4
The User Layer	1-5
The Resource Block	1-5
Function Blocks.....	1-6
Transducer Blocks.....	1-6
Objects.....	1-7
FOUNDATION Fieldbus Concepts.....	1-7
Links	1-7
Types of Devices	1-7
Virtual Field Devices	1-8

Device Descriptions	1-8
NI-FBUS Communications Manager Overview	1-9
How NI-FBUS Communications Manager Helps You.....	1-9
The NI-FBUS Communications Manager Software Components	1-10
Driver.....	1-10
Binaries.....	1-10
Dynamic Link Libraries.....	1-10
Utilities	1-11
Static Library	1-11
Sample Application Files.....	1-11
Header Files.....	1-11
Readme File.....	1-12

Chapter 2

Developing Your Application

Using the NI-FBUS Communications Manager Software	2-1
Using NI-FBUS Functions.....	2-1
Administrative Functions	2-1
Example: How to Use Administrative Functions	2-2
Core Functions.....	2-2
Example: How to Use Core Functions	2-3
Alert and Trend Functions	2-3
Device Description Functions.....	2-4
Using the NI-FBUS Communications Manager Process	2-5
Developing Your NI-FBUS Communications Manager Application	2-6
Choosing Your Level of Communication.....	2-6
Choosing Between Tag-Based and Index-Based Access	2-7
Choosing Between Multi-Threaded and Single-Threaded Access	2-7
Using Single Threading	2-7
Using Multi-Threading	2-8
Using the NI-FBUS Dialog Utility to Communicate with Devices	2-8
Writing Your NI-FBUS Communications Manager Application	2-9
Compiling, Linking, and Running Your NI-FBUS Communications Manager Application	2-10

Chapter 3

NI-FBUS Dialog Utility

NI-FBUS Dialog Utility Overview	3-1
NI-FBUS Dialog Examples	3-2
Getting a Device List.....	3-2
Downloading a Schedule to an Interface.....	3-3
Reading a Parameter Using TAG.PARAM Access.....	3-5
Waiting for a Trend.....	3-6

Chapter 4

NI-FBUS Interface Configuration Utility

Introduction to the NI-FBUS Interface Configuration Utility	4-1
Starting the NI-FBUS Interface Configuration Utility	4-2
Using the NI-FBUS Interface Configuration Utility	4-2
Configuring a New Interface in Windows NT	4-2
Configuring a New PCMCIA-FBUS in Windows 95	4-4
Configuring a New AT-FBUS in Windows 95	4-4
Configuring Port Information.....	4-4
Changing or Deleting Existing Interface Information in Windows NT	4-6
Changing or Deleting Existing Interface Information in Windows 95	4-8

Appendix A

Configuring the Link Active Schedule File

Appendix B

Customer Communication

Glossary

Index

Figures

Figure 1-1.	Fieldbus-Based Control System.....	1-2
Figure 1-2.	The Fieldbus Model Compared to the OSI 7-Layer Communications Model.....	1-3
Figure 1-3.	The User Layer	1-5
Figure 3-1.	Getting a Device List	3-3
Figure 3-2.	Downloading a Link Active Schedule	3-4
Figure 3-3.	Reading a Parameter Using TAG.PARAMETER Access	3-6
Figure 3-4.	Waiting for a Trend	3-7
Figure 4-1.	Assigning Board Information	4-3
Figure 4-2.	Assigning Port Information	4-5
Figure 4-3.	Advanced Stack Configuration Dialog Box	4-6
Figure 4-4.	Choosing a Port	4-7

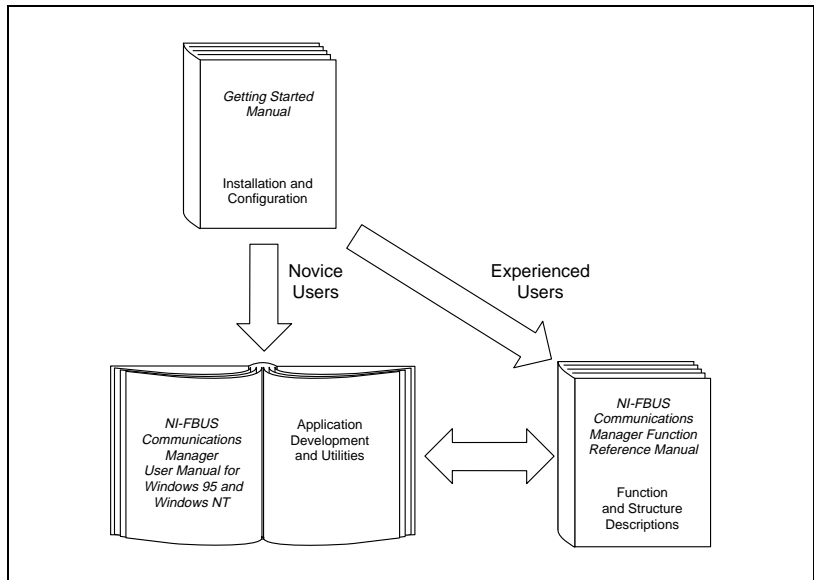
Tables

Table 1-1.	Ten Standard FOUNDATION Fieldbus-Defined Function Blocks.....	1-6
Table A-1.	Valid Variable Names and Values for the Schedule Summary Section	A-2
Table A-2.	Valid Variable Names and Values for the Subschedule Section	A-3
Table A-3.	Valid Variable Names and Values for the Sequence Section	A-3
Table A-4.	Valid Variable Names Including the Variable <i>N</i> and Values for the Sequence Section	A-3

About This Manual

This manual gives an overview of fieldbus, describes the NI-FBUS Communications Manager software, and explains how to use the software. The NI-FBUS Communications Manager software for Windows 95 is meant to be used with the Microsoft Windows 95 operating system. The NI-FBUS Communications Manager software for Windows NT is meant to be used with the Microsoft Windows NT (version 3.5.1 and later) operating system. This manual assumes that you are already familiar with the appropriate Microsoft operating system.

How to Use the Manual Set



This user manual helps you to learn how to employ the NI-FBUS Communications Manager in your application.

Use the getting started manual to install and configure your fieldbus interface and the NI-FBUS Communications Manager software for Windows 95 or Windows NT.

Use the *NI-FBUS Communications Manager Function Reference Manual* to look up specific information about NI-FBUS Communications Manager functions, such as input and output parameters, syntax, and error messages.

Organization of This Manual

This manual is organized as follows:

- Chapter 1, *Introduction*, introduces fieldbus and the NI-FBUS Communications Manager software.
- Chapter 2, *Developing Your Application*, explains how to use the NI-FBUS Communications Manager software for Windows 95 or Windows NT to develop your fieldbus application.
- Chapter 3, *NI-FBUS Dialog Utility*, describes the NI-FBUS Dialog utility and gives examples of how to use it.
- Chapter 4, *NI-FBUS Interface Configuration Utility*, explains how to use `fbconf`, the NI-FBUS Interface Configuration utility.
- Appendix A, *Configuring the Link Active Schedule File*, contains information about how to configure your Link Active Schedule file.
- Appendix B, *Customer Communication*, contains forms you can use to request help from National Instruments or to comment on our products and manuals.
- The *Glossary* contains an alphabetical list and description of terms used in this manual, including abbreviations, acronyms, metric prefixes, mnemonics, and symbols.
- The *Index* contains an alphabetical list of key terms and topics in this manual, including the page where you can find each one.

Conventions Used in This Manual

This manual uses the following conventions:

»	The » symbol leads you through nested menu items and dialog box options to a final action. The sequence File»Page Setup»Options»Substitute Fonts directs you to pull down the File menu, select the Page Setup item, select Options , and finally select the Substitute Fonts options from the last dialog box.
bold	Bold text denotes parameters, menus, menu items, dialog box buttons or options, and error messages.
<i>bold italic</i>	Bold italic text denotes a note, caution, or warning.
bold monospace	Bold text in this font denotes the messages and responses that the computer automatically prints to the screen. This font also emphasizes lines of code that are unique.
<i>italic</i>	Italic text denotes emphasis, a cross reference, or an introduction to a key concept. This font also denotes text for which you supply the appropriate word or value.
<i>italic monospace</i>	Italic text in this font denotes that you must supply the appropriate words or values in the place of these items.
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 for statements and comments taken from programs.

Related Documentation

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

- *Foundation Specification: System Architecture*
- *Foundation Specification: System Management*

- *Foundation Specification: Network Management*
- *Foundation Specification: Fieldbus Message Specification*
- *Foundation Specification: Function Block Application Process, Parts 1 and 2*
- *Foundation Specification: Device Description Services User Guide*
- *Foundation Specification: 31.25 kbit/s Physical Layer Profile*
- *Foundation Specification: Device Description Services User Guide*
- *Device Description Language Specification*

Customer Communication

National Instruments wants to receive your comments on our products and manuals. We are interested in the applications you develop with our products, and we want to help if you have problems with them. To make it easy for you to contact us, this manual contains comment and configuration forms for you to complete. These forms are in Appendix B, *Customer Communication*, at the end of this manual.

Introduction

This chapter introduces fieldbus and the NI-FBUS Communications Manager software.

Fieldbus Overview

This section gives a short overview of fieldbus. Refer to the *Glossary* for more explanation of fieldbus terms and concepts.

What Is Fieldbus?

The term *fieldbus* generally refers to an all-digital, two-way communication system that connects control systems to instrumentation.

Benefits of Fieldbus

Fieldbus offers important benefits over some other protocols. Fieldbus supports digital encoding of data, two-way communication, many types of messages, and multiple devices, all on the same set of wires.

Most control systems today have a combination of analog, hybrid, and proprietary digital networks.

Fieldbus solves the problem of proprietary networks by using standardized networks to connect systems and devices, as Figure 1-1 shows.

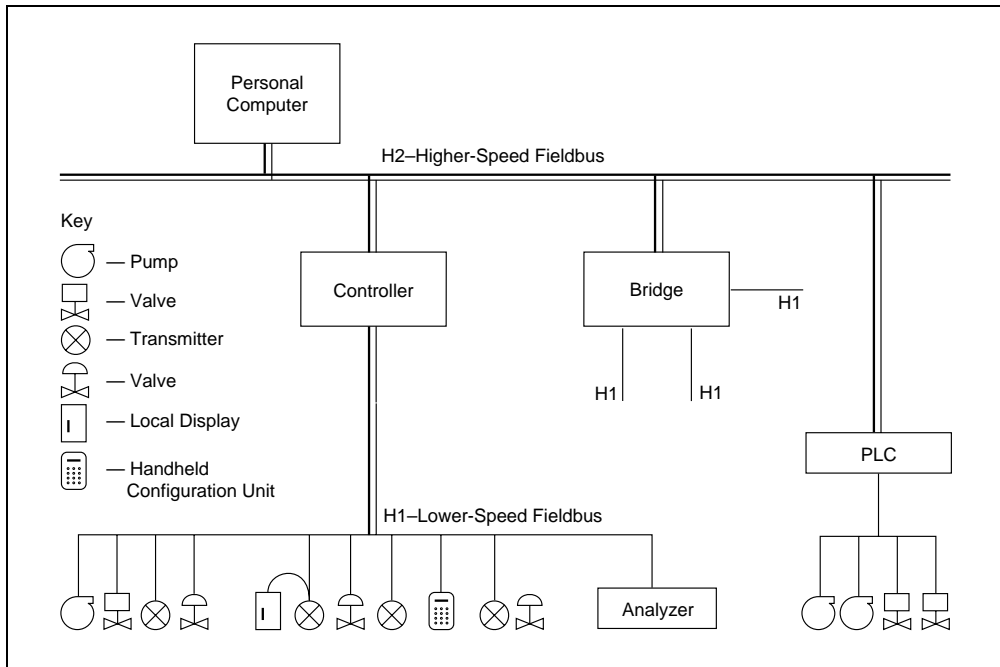


Figure 1-1. Fieldbus-Based Control System

The Fieldbus Foundation

The *Fieldbus Foundation* is an organization that developed a fieldbus network specifically based upon the work and principles of the ISA/IEC standards committees. The goal of the Fieldbus Foundation and its members is not simply to produce a standard, but to help create products that use a robust industrial network based on existing standards and other proven technologies.

FOUNDATION Fieldbus Technology

FOUNDATION Fieldbus, which is the communications network the Fieldbus Foundation created, specifically targets the need for robust, distributed control in process control environments. FOUNDATION Fieldbus technology consists of the Physical Layer, the Communication Stack, and the User Layer. Figure 1-2 shows a diagram of the fieldbus layers compared to the Open Systems Interconnect (OSI) layered communication model. Notice that the OSI model does not define a User Layer.

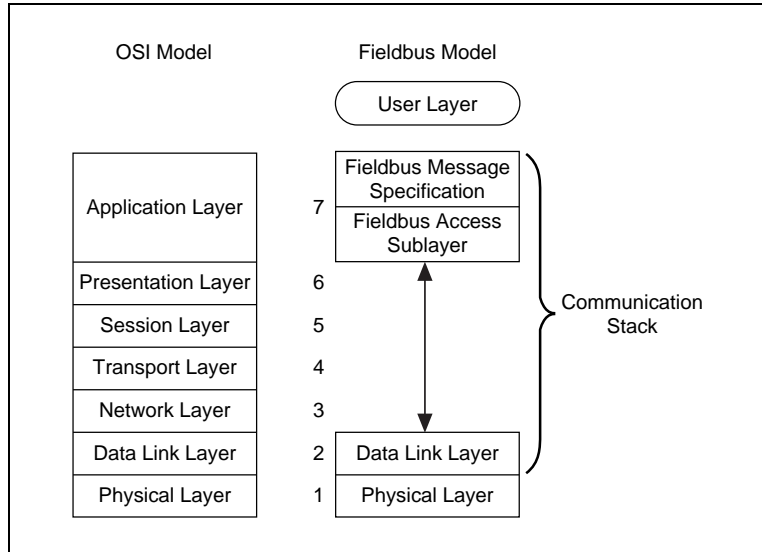


Figure 1-2. The Fieldbus Model Compared to the OSI 7-Layer Communications Model

FOUNDATION Fieldbus does not implement layers 3, 4, 5, and 6 of the OSI model shown in Figure 1-2 because the services of these layers are not required in a process control application. A very important part of FOUNDATION Fieldbus is the defined User Layer, often referred to as Layer 8.

Refer to the FOUNDATION Fieldbus specifications for more information on the layers of the FOUNDATION Fieldbus network.

FOUNDATION Fieldbus Layers

The Physical Layer

The Physical Layer converts digital Fieldbus messages from the communication stack to physical signals on the Fieldbus transmission medium and vice versa. See the *Foundation Specification: 31.25 kbit/s Physical Layer Profile* for the FOUNDATION Fieldbus Physical Layer Specifications.

The Communication Stack

The Communication Stack performs the services required to interface the User Application to the Physical Layer. As shown in Figure 1-2, the Communication Stack consists of three layers: the Fieldbus Message Specification, the Fieldbus Access Sublayer, and the Data Link Layer.

The Data Link Layer manages access to the Fieldbus through the LAS by splitting data into frames to send on the physical layer, receiving acknowledgment frames, and re-transmitting frames if they are not received correctly. It also performs error checking to maintain a sound virtual channel to the next layer.

The Fieldbus Access Sublayer (FAS) layer of the stack provides an interface between the Data Link Layer and Layer 7. The FAS provides communication services such as client/server, publisher/subscriber and event distribution.

The Fieldbus Messaging Specification (FMS) layer of the stack defines a model for applications to interact over the Fieldbus. The Object Dictionary (OD) and the Virtual Field Device (VFD) are important in this model. The OD is a structure in a Fieldbus device that describes data that can be communicated on the Fieldbus. You can think of the OD as a lookup table that gives information such as data type about a value that can be read from or written to a device. The VFD is a model for remotely viewing data described in the object dictionary. The services provided by FMS allow you to read and write information about the OD, read and write the data variables described in the OD, and perform other activities such as uploading/downloading data, and invoking programs inside a device.

In addition, there are two management layers called System Management (SM) and Network Management (NM). SM assigns addresses and physical device tags, maintains the Function Block Schedule for the FBs in that device, and distributes application time. You can also locate a device or a Function Block tag through SM.

Network Management contains objects that other layers of the communication stack use, such as Data Link, FAS, and FMS. You can read and write SM and NM objects over the Fieldbus using FMS Read and FMS Write services.

The User Layer

The User Layer defines blocks and objects that represent the functions and data available in a device. Rather than interfacing to a device through a set of commands, like most communication protocols, a FOUNDATION Fieldbus user interacts with devices through a set of blocks and objects that define device capabilities in a standardized way. The User Layer shown in Figure 1-2 consists of the Resource Block, and one or more Transducer Blocks and Function Blocks, as illustrated in Figure 1-3.

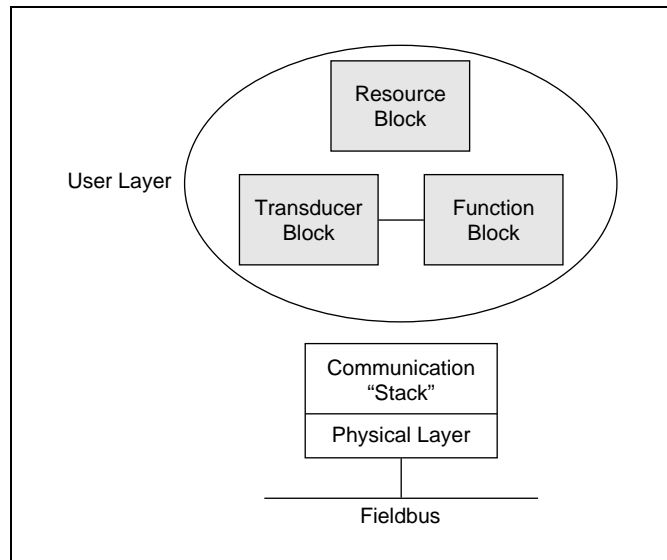


Figure 1-3. The User Layer

The Resource Block

The resource block describes general characteristics, such as manufacturer and device name. Only one resource block per VFD is allowed.

Function Blocks

Function blocks are the core components of a control system. FOUNDATION Fieldbus defines a standard set of function blocks. There are ten function blocks for the most basic control and I/O functions are shown in Table 1-1. Manufacturers can define their own function blocks.

Table 1-1. Ten Standard FOUNDATION Fieldbus-Defined Function Blocks

Function Block Name	Symbol
Analog Input	AI
Analog Output	AO
Bias/Gain	BG
Control Selector	CS
Discrete Input	DI
Discrete Output	DO
Manual Loader	ML
Proportional/Derivative	PD
Proportional/Integral/Derivative	PID
Ratio	RA

You can connect the input and output of individual function blocks to specify communication of data between blocks. Also, you can precisely schedule the execution of a function block and the transmission of its output data using the communication stack, which means you can control loops directly over the network.

Transducer Blocks

Transducer blocks interface to the sensing hardware in the device. They also perform the digitizing, filtering, and scaling conversions needed to present input data to function blocks or convert 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. Various profile groups have defined transducer blocks for different types of devices. Manufacturers can define their own transducer blocks.

Objects

In addition to the blocks described, the User Layer of a FOUNDATION Fieldbus device also contains four types of objects. Link objects define connections between function block input and output across the network. Trend objects accumulate values of function block parameters for access over the network. Alert objects report alarms and events. View objects are predefined groupings of parameter sets that Man-Machine Interface (MMI) applications use.

FOUNDATION Fieldbus Concepts

This section discusses basic concepts of fieldbus architecture that you must understand to use the NI-FBUS Communications Manager.

Links

A fieldbus network consists of one or more links. Each link is configured with a unique link identifier.

Each link on the fieldbus network consists of one or more physical devices. The devices can be field devices (temperature transmitter, valve, and so on) or host devices (PCs, Digital Control Systems). Each physical device is configured with a physical device tag, an address, and a device ID. The physical device tag must be unique within a fieldbus system, and the address must be unique within each link. The device manufacturer assigns a device ID that is unique to the device.

Types of Devices

There are three types of physical devices from a protocol point of view—Link Masters, basic devices, and bridges. Each link has one device called a Link Active Scheduler (LAS) that executes the Link Active Schedule, circulates tokens, distributes time, and probes for new devices. A Link Master device can become the LAS, but a basic device cannot become the LAS. A bridge device connects two or more links together.

The LAS controls access to the bus. For cyclic communications, the LAS maintains the list of scheduled communication times (the Link Active Schedule) and compels a device to transmit data at the appropriate time, making communication very deterministic. For acyclic communication, the LAS grants permission to a device to use the fieldbus by passing a token to the device.

Virtual Field Devices

Each physical device on the fieldbus can have one or more Virtual Field Devices (VFDs). A network configuration application can assign each VFD a tag that is unique within the device. Most devices have only one VFD. Each VFD has one resource block and one or more function blocks and transducer blocks. Each block should be assigned a tag that is unique within the fieldbus system.

Device Descriptions

A key objective for FOUNDATION Fieldbus is interoperability—the ability to build systems comprised of devices from a variety of manufacturers and take full advantage of both the standard and unique capabilities of every device.

Instead of requiring device manufacturers use only a given set of functions in a device in order to ensure that a system can always communicate with a new device, FOUNDATION Fieldbus uses *Device Descriptions (DDs)*, which describe all the functions in a device. The DD defines the parameters of the blocks. It also defines attributes of parameters and blocks like help strings in different languages, ranges of values for parameters, and so on. Using the Device Description, the host in a control system can obtain the information needed to create an interface that configures parameters, calibrates, performs diagnostics, and accomplishes other functions on the device.

The developer of a fieldbus device uses Device Description Language to create the Device Description for a device. The DDL is compiled using the Fieldbus Foundation-supplied Tokenizer, which creates a binary form of the code to ship to the end user with the instrument. The output files are made available to the host devices. The host devices can access information in these files through Device Description Services (DDS). The Fieldbus Foundation supplies the DDL for the standard function blocks and the resource block.

NI-FBUS Communications Manager Overview

The NI-FBUS Communications Manager implements a high-level Application Programmer Interface (API) you can use to interface with the National Instruments FOUNDATION Fieldbus (FF) communication stack and hardware. The main purpose of the NI-FBUS Communications Manager is to make the details of the fieldbus communication protocols transparent by offering you an API that supports *TAG.PARAMETER* access. You need a general knowledge of the fieldbus architecture (outlined in the *Fieldbus Overview* section of this chapter) to understand and use the NI-FBUS Communications Manager.

How the NI-FBUS Communications Manager Helps You

The NI-FBUS Communications Manager interface makes fieldbus protocols transparent. The NI-FBUS Communications Manager interfaces between the communication stack and the user application. It handles the details of interfacing to the Fieldbus Messaging Specification (FMS) and lower layers of the communications stack. The NI-FBUS Communications Manager also hides the low-level details of Virtual Communication Relationships (VCRs), connection management, addresses, and Object Dictionary indices, and offers name access to physical devices, Virtual Field Devices (VFDs), function blocks, transducer blocks, and parameters.

The NI-FBUS Communications Manager API is independent of National Instruments fieldbus hardware and the operating system. With the NI-FBUS Communications Manager, you can plug multiple National Instruments fieldbus interfaces of any type into the same PC, and use them through the NI-FBUS Communications Manager API.

The NI-FBUS Communications Manager is interface independent because it does not require you to specify which fieldbus interface to use in NI-FBUS Communications Manager calls. It determines which interface to send certain fieldbus messages over. The NI-FBUS Communications Manager allows you to write applications that are as independent as possible of the actual configuration of your fieldbus interfaces.

The NI-FBUS Communications Manager is useful for developing host applications. Typical examples are configurators and applications for monitoring a fieldbus network, diagnosing a network, and developing interfaces to Man-Machine Interface (MMI) packages.

NI-FBUS Communications Manager Software Components

This section lists the important elements of the NI-FBUS Communications Manager software for Windows 95 and Windows NT, and describes the function of each element.

Driver

In Windows NT, `nifb.sys` is a Windows NT kernel-mode driver that interfaces between the NI-FBUS Communications Manager and the fieldbus interface. The NI-FBUS Communications Manager software installer installs it in your standard Windows NT drivers directory.

In Windows 95, `nifb.vxd` is a Windows 95 virtual device driver that interfaces between the NI-FBUS Communications Manager and the fieldbus interface. The NI-FBUS Communications Manager software installer installs it in your Windows 95 `System` directory.

Binaries

`nifb.exe` is an executable that constitutes the NI-FBUS Communications Manager process. This process must be executing in order for your applications to work.

`ffstack.bin` is the binary stack file that the NI-FBUS Communications Manager downloads to a fieldbus interface device (such as the National Instruments AT-FBUS board.) This file is a binary image of the Fieldbus Foundation communication stack.

Dynamic Link Libraries

`nifb.dll` is the dynamic link library that is installed in your Windows NT directory. `nifb.dll` is necessary for your application to communicate with the `nifb` process.

`drvintf.dll` is another dynamic link library that is installed in your Windows NT or Windows 95 directory. The `nifb` process needs `drvintf.dll` to communicate with the kernel-mode driver.

Utilities

`nifbldg.exe` is an interactive dialog utility that you can use to communicate with the fieldbus network devices. It helps you to learn the NI-FBUS Communications Manager routines.

In Windows NT, `fbconf.exe` is a configuration program that lets you add fieldbus interfaces or change the hardware configuration parameters for your fieldbus interface devices (such as the AT-FBUS boards). It also lets you assign a logical name to each port on the fieldbus interface and provide the NI-FBUS Communications Manager with Device Description location information.

In Windows 95, you can view hardware configuration parameters with `fbconf`, but must use the Windows 95 Device Manager to modify them. However, you can still assign logical names to your interface ports with the Windows 95 version of `fbconf`.

Static Library

`nifb.lib` is the static library that your application must link with in order to communicate with the `nifb` process. `nifb.lib` is compiled with Microsoft C. `nifb_bor.lib` is provided for Borland C users.

Sample Application Files

`nifbtest.c`, `nifb_mt.c`, and `nifbdd.c` are source code files that you can use with the `nifb.lib` library to make a sample application. All of these files ship with the NI-FBUS Communications Manager software, but you must use your own compiler to create the sample application.

`sched.ini` is a sample Link Active Schedule file. Refer to Appendix A, *Configuring the Link Active Schedule File*, for information about how to configure this file.

Header Files

`nifbus.h` is an include file that contains some data type declarations, error code declarations and function prototypes.

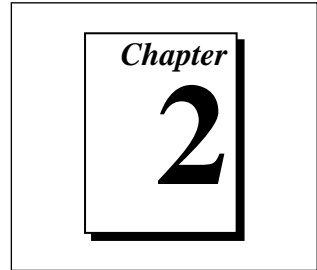
In addition to `nifbus.h` there are fourteen other header files that `nifbus.h` includes for you. Therefore, your application only has to include `nifbus.h`.

Readme File

`readme.txt` is a documentation file that contains important information about the NI-FBUS Communications Manager software, and a list of known bugs or anomalies in the software.

Proceed to Chapter 2, *Developing Your Application*, to develop your fieldbus application.

Developing Your Application



This chapter explains how to use the NI-FBUS Communications Manager software for Windows 95 or Windows NT to develop your fieldbus application.

Using the NI-FBUS Communications Manager Software

This section discusses considerations about using the NI-FBUS Communications Manager software for Windows 95 or Windows NT.

Using NI-FBUS Functions

The NI-FBUS functions are classified in four categories:

- Administrative Functions
- Core Functions
- Alert and Trend Functions
- Device Description Functions

Administrative Functions

You can use the administrative functions to get the list of physical devices in a link, get a list of VFDs in a physical device, and get a list of blocks (resource, function, transducer) from a VFD. The administrative functions include `nifGetDeviceList`, `nifGetVfdList`, and `nifGetBlockList`. Typically, you have to call these before you call a core, alert, or another administrative function.

Because you can use the NI-FBUS Communications Manager to communicate with each of the FOUNDATION Fieldbus entities, such as

link, physical device, VFD, and block, there are `nifOpen` calls for you open and get a descriptor to each of these entities.

Example: How to Use Administrative Functions

Suppose you want to get a descriptor to a block with `nifOpenBlock` before you read or write the block parameters. Then you want to open a block using the block's tag.

To open a block with the tag `TI101_Analog_Input`, you invoke `nifOpenBlock(sessionDesc, "TI101_Analog_Input", &blockDesc)`, where `sessionDesc` is the descriptor of the session that you have established with the NI-FBUS Communications Manager. The NI-FBUS Communications Manager returns the descriptor of the block that you have opened in `blockDesc`. From then on, you can use this descriptor for calls associated with this block.

Core Functions

Core NI-FBUS functions are the functions that deal with processing function block parameters—primarily the `nifReadObject` and `nifWriteObject` functions, which read and write block parameters. The NI-FBUS Communications Manager encapsulates the Device Description Services (DDS) with the core function `nifGetObjectAttributes`, which gives you the Device Description attributes of any parameter. A function to get the object's data size, `nifGetObjectSize`, is also a core function.

Function blocks contain view or display objects. As the name implies, these objects are a collection of parameters in function blocks that are typically displayed in an operator console. Four view objects are defined for each of the 10 standard function blocks in the FOUNDATION Fieldbus specification.

The following examples are a good summation of the NI-FBUS Communications Manager, because they demonstrate that details such as VCRs, indices, and connections are hidden by the `TAG.PARAMETER` access the NI-FBUS Communications Manager provides. However, to correctly write an application using the NI-FBUS Communications Manager, you must be familiar with the *Foundation Specification: Function Block Application Process Parts 1 and 2*—the standard blocks, their parameters and their syntax—and have an idea of the architecture of

fieldbus. Refer to the *Fieldbus Overview* section in Chapter 1, *Introduction*, in this manual for an outline of fieldbus architecture.

Example: How to Use Core Functions

Suppose the object VIEW_1 for a PID function block consists of GAIN, RATE, SP, CAS_IN, MODE, and ALARM_SUM parameters of the PID function block. You want to get the values of all these parameters using a single read of the VIEW_1 object. If the tag of a PID function block is TIC101_PID, you can read the VIEW_1 object by executing the following function call:

```
nifReadObject(sessionDesc, "TIC101_PID.VIEW_1", buffer,
&cnt)
```

Notice that it is not necessary to have a block descriptor to read the object's parameters. If you *do* have the block descriptor, you can read the object with the following call:

```
nifReadObject(blockDesc, "VIEW_1", buffer, &cnt)
```

You can get the block descriptor using nifOpenBlock, which returns blockDesc.

If you wanted to change the set point of the preceding PID block, you could do so with the following call:

```
nifWriteObject(sessionDesc, "TIC101_PID.SP", buffer,
cnt)
```

Alert and Trend Functions

When a properly configured device detects an alarm condition, it broadcasts the data. A host device should receive the alarm and send a communication acknowledgment and an operator acknowledgment to the field device. The field device can also collect trends based on a configured sample type and interval. When it collects 16 samples, it broadcasts the trend data on the fieldbus. Any number of interested hosts can collect this data. For more details, refer to the *Foundation Specification: Function Block Application Process Part 1*.

With a program such as the National Instruments Fieldbus Network Configuration utility, you can configure the FOUNDATION Fieldbus field devices to broadcast alert and trend data.

The NI-FBUS Communications Manager has functions to receive trends and alerts from configured devices and to perform operator acknowledgment on alerts. `nifWaitAlert` lets you wait for an alert from any device in a link, any function block in a physical device, or a specific function block, depending on the type of descriptor that you pass it. When the NI-FBUS Communications Manager receives an alert, it returns a structure containing information about the alert. The NI-FBUS Communications Manager sends the communication acknowledgment to the device automatically. The NI-FBUS Communications Manager provides a separate function, `nifAcknowledgeAlarm`, to send the operator acknowledgment.

Similarly, `nifWaitTrend` lets you wait for a trend from any device in a link, any function block in a physical device, or a specific function block, depending on the type of descriptor you pass it. When the NI-FBUS Communications Manager receives a trend, it returns a structure containing information about the trend, along with the trend data itself.

`nifWaitAlert` and `nifWaitTrend` wait until an alert or trend is received before returning, so you might want to have separate threads invoke these functions.

Device Description Functions

The NI-FBUS Communications Manager gives your applications access to Device Descriptions, which are binary files that describe the characteristics of blocks and parameters. Your application can use the NI-FBUS function `nifGetObjectAttributes` to decode attributes of parameters including data type, data size, help strings, and other attributes defined in the *Device Description Language Specification*. In addition, Device Description symbol files are used automatically to assist in allowing your applications to access parameters by name.

The NI-FBUS Communications Manager is shipped with Device Descriptions for all standard Fieldbus Foundation function blocks. The NI-FBUS Communications Manager can provide attributes for all parameters of all standard function blocks, even if the device manufacturers for your devices did not provide Device Descriptions. However, to get the attributes of parameters of nonstandard (not FOUNDATION Fieldbus defined) blocks, the NI-FBUS Communications Manager requires that the device manufacturer provide the Device Description.

Using the NI-FBUS Communications Manager Process

For any of your applications that relate to the NI-FBUS Communications Manager to run correctly, you must successfully launch the NI-FBUS Communications Manager process. The NI-FBUS Communications Manager process is the medium by which your application communicates with the devices on the fieldbus network. The NI-FBUS Communications Manager process receives requests from your application and passes them on to the specified fieldbus device through the fieldbus interface connected to your machine. Refer to the *Begin to Use the NI-FBUS Communications Manager Software* chapter in the getting started manual for instructions on how to start the NI-FBUS Communications Manager process.

At startup time, the NI-FBUS Communications Manager process downloads the Fieldbus Foundation communication stack file `ffstack.bin` (provided in your kit) to the fieldbus interfaces connected to your machine. It then downloads the communication stack configuration parameters, such as the fieldbus network address for the interface device, and so on, to each interface device. You can edit these parameters using the NI-FBUS Interface Configuration utility by clicking the **Advanced** button on the dialog box for the **Edit Port** information.

You must make sure to specify a unique, non-default fieldbus network address for the NI-FBUS Communications Manager to work properly. You can use a default address if another entity on the fieldbus assigns your interface a non-default address. You can change the address from the NI-FBUS Interface Configuration utility in the **Edit Port** dialog box. You must restart the NI-FBUS Communications Manager for any changes you make to take effect.

The NI-FBUS Communications Manager process features non-volatile storage of all network parameters, including the last known Link Active Schedule. After network parameters, including the Link Active Schedule, are stored, the NI-FBUS Communications Manager automatically reloads them to the interface on startup.

At installation time, the non-volatile copy of the schedule is empty, but you can make the NI-FBUS Communications Manager store the non-volatile Link Active Schedule by downloading it to your fieldbus interface. To download a Link Active Schedule to your fieldbus interface, you can use the NI-FBUS Dialog utility. Refer to Chapter 3, *NI-FBUS*

Dialog Utility, in this manual for an example of how to download the Link Active Schedule to your fieldbus interface. You can also use the National Instruments Fieldbus Network Configuration utility to download a Link Active Schedule to your fieldbus interface.

At any point in the NI-FBUS Communications Manager process start-up, if the NI-FBUS Communications Manager process is unable to find information it needs to start up, error messages will appear. You may ignore these messages and continue; however, this will result in your application not being able to communicate with the interface devices for which the error messages appeared. These messages tell you the information that the NI-FBUS Communications Manager process is looking for, but cannot find.

Developing Your NI-FBUS Communications Manager Application

This section contains information to help you develop your NI-FBUS Communications Manager application.

Choosing Your Level of Communication

While a few functions require a specific type of descriptor (for example, `nifGetDeviceList` requires a link descriptor), many functions (such as the Core functions, and the Alert and Trend functions) allow you to communicate using any type of descriptor. With these functions, the descriptor type you should choose depends on what is most convenient for you in designing your application, because there is no significant difference in performance between the different types.

For example, if it is convenient for your application to use only a session descriptor, and to keep track of tags for each block, so that you refer to all parameters in `BLOCKTAG.PARAMNAME` format, then you should write your application this way. If it is easier for you to keep track of a descriptor for each block rather than a tag for each block, then you should open a block descriptor for each block you are communicating with, keep track of that descriptor value, and access parameters by `PARAMNAME` using the block descriptor.

Choosing Between Tag-Based and Index-Based Access

The NI-FBUS Communications Manager supports access by name or by index for all block parameters. National Instruments recommends that you access all variables by name. Although access by index might be slightly faster in some cases, an application cannot always reliably determine indices.

The NI-FBUS Communications Manager can convert the parameter name you specify to the final index that Foundation protocols must use to access the parameter over the network. The NI-FBUS Communications Manager converts the name to an index using standard Fieldbus Foundation-specified methods, which include a check to the device at run time to verify the index. If you hard code indices, you will have to modify them when the devices they are accessing become replaced, upgraded, or have new blocks created on them.

Choosing Between Multi-Threaded and Single-Threaded Access

All NI-FBUS functions are synchronous, meaning that the calling function is blocked until the NI-FBUS call completes. A fieldbus device usually takes tens of milliseconds to respond to a block parameter read or write. It takes longer if any communication errors occur. The NI-FBUS Communications Manager uses the protocol connections to communicate with the devices. If a connection is lost, the NI-FBUS Communications Manager tries to reestablish the connection. When a connection is lost, an NI-FBUS read or write call may take several seconds to complete.

Using Single Threading

If potential delays like the ones discussed in the previous paragraph are acceptable for your application, you can write your application or the fieldbus access part of your application as a single thread. Single-threaded applications are easier to develop, debug, and test, because you do not have to consider exclusion between threads. If you are writing an application for testing, monitoring, or configuring a single device, a single-threaded application might be adequate.

Using Multi-Threading

If your application monitors or tests several devices at a time, communication delays might affect the throughput of your application and therefore be unacceptable. If so, you can develop a multi-threaded application to improve the performance of your application. There are several ways to multi-thread your application.

If you are accessing information from function blocks or transducer blocks, you might want to create a thread for each block. Each block's thread will read and write information from that block. If creating a thread for each block is excessive, you might consider an architecture in which you have a set of threads dedicated to fieldbus I/O. Your application can then interface with I/O threads through a shared queue in which threads put their I/O requests. When the I/O completes, the I/O threads can inform the application by passing a message or some other synchronization scheme.

If your application performs trending or alarm handling, you might want to have separate threads that perform these functions. You can make a thread wait for a trend or alarm with the `nifWaitTrend` or `nifWaitAlert` function and then process the trend or alarm when it arrives. If you are monitoring the live list (the current list of devices on the bus), you may have a dedicated thread that calls `nifGetDeviceList`, because the call will not return until the live list changes.

Using the NI-FBUS Dialog Utility to Communicate with Devices

The NI-FBUS Dialog utility (`nifbdlg.exe`) helps you perform simple tests of your whole fieldbus setup, including the NI-FBUS Communications Manager, your interface board(s), and any devices you have. The NI-FBUS Dialog utility graphical interface has dialog boxes that call the NI-FBUS Communications Manager API, and allows you to specify parameters and make NI-FBUS calls. For example, you can use the NI-FBUS Dialog utility to get a list of devices on your network, as well as view and set parameters in each device. For more details, see Chapter 3, *NI-FBUS Dialog Utility*.

Writing Your NI-FBUS Communications Manager Application

Use the following guidelines to make sure your application uses the NI-FBUS Communications Manager interface properly.

- Always call `nifOpenSession` early in your program and check the return value of the call. This check verifies that the NI-FBUS Communications Manager process is running, which is a prerequisite for your application to access the fieldbus network. If this call fails, your application should inform the user that the fieldbus is currently inaccessible.
- Always close any descriptors that you open before your program exits, including session descriptors. The NI-FBUS Communications Manager requires that your application closes all descriptors that it opens.
- Always check the return values from NI-FBUS calls. The NI-FBUS Communications Manager is a high-level API, and performs many operations that can fail because of incorrect parameters, incorrect bus configuration, or communication failures. An application that fails to check return values might use output parameters from NI-FBUS calls that are NULL or uninitialized, leading to incorrect behavior or a program crash.
- If you plan to call any of the indefinitely-blocking functions including `nifGetDeviceList`, `nifWaitAlert`, and `nifWaitTrend`, you should probably use a separate descriptor for these calls. To terminate these calls early, you have to close the descriptor. Having a separate descriptor will ensure that terminating these calls does not affect any other NI-FBUS calls your application has pending.
- If the NI-FBUS Communications Manager stops for any reason, any outstanding calls in your application complete with the error `E_SERVER_CONNECTION_LOST`. At this point, all of the descriptors that you have (including the session) are invalid. If you restart the NI-FBUS Communications Manager, your application should recover by opening a new session to the NI-FBUS Communications Manager and opening all new descriptors. After this recovery procedure, your application should be fully operational.

Compiling, Linking, and Running Your NI-FBUS Communications Manager Application

To compile, link, and execute your application, you must carry out the following steps:

- Add the line `#include "nifbus.h"` to any of your source files that make NI-FBUS calls. The `nifbus.h` file is located in the `includes` subdirectory of your installation. Also, make sure that the `includes` subdirectory is included in your project's settings.
- Link your application with `nifb.lib` (or `nifb_bor.lib`, if you are using Borland's compiler), which is located in the `libs` subdirectory of your installation.
- Make sure that `nifb.dll` is present in your Windows 95 or Windows NT directory. `nifb.dll` is an interface DLL required to interface to the NI-FBUS Communications Manager process. `nifb.dll` must be present when your application runs.
- Make sure that the NI-FBUS Communications Manager process has started up and is entirely initialized before your application makes its first NI-FBUS call.
- Make sure your compiler has the structure padding or alignment parameter set to 8 bytes. This will allow proper communication of data structures.
- The `nifbus.h` header file and `nifb.lib` library have been compiled and linked with Microsoft Visual C, versions 4.0, 4.1, and 4.2. The `nifbus.h` header file and `nifb_bor.lib` library have been compiled and linked with Borland C++, version 5.0 only.

NI-FBUS Dialog Utility

This chapter describes the NI-FBUS Dialog utility and gives examples of how to use it.

NI-FBUS Dialog Utility Overview

The NI-FBUS Dialog utility allows you to interact with your devices over the fieldbus by opening descriptors, making single NI-FBUS calls, and viewing the results. You might want to use the NI-FBUS Dialog utility to verify installation and device operation, or to learn the NI-FBUS Communications Manager API.

To run the NI-FBUS Dialog utility under Windows 95, select **Start»Programs»NI-FBUS»NI-FBUS Dialog**. Under Windows NT, double-click on the **NIFBus Dialog** icon in your **NI-FBUS** program group.

When you open the Dialog utility, a window appears containing a single item called **Open Descriptors**. This icon is the root of a graphical tree of icons for each of the NI-FBUS descriptors you open using the NI-FBUS Dialog utility. The area below the icon remains empty until you make an NI-FBUS call to open a descriptor. When you open a descriptor, the NI-FBUS Dialog utility adds an icon representing that descriptor in a tree structure on this area of the screen.

You can use the Dialog Utility to perform operations on the descriptors you have opened. Select the operation you want to perform on a descriptor by right-clicking on the descriptor icon and choosing an item on the popup menu that appears, or by selecting the icon with a single click and choosing an item on the **Actions** menu. The choices that appear on the menu depend on the type of descriptor you have selected.

The first thing you should do when using the NI-FBUS Dialog utility is open a session to the NI-FBUS Communications Manager. This creates a session descriptor icon in the tree structure on the screen. You can right-click on this session descriptor to open other descriptors or access NI-FBUS functions.

NI-FBUS Dialog Examples

Getting a Device List

Follow these steps to practice using the NI-FBUS Dialog utility to get a device list.

1. Open the NI-FBUS Dialog utility by double-clicking on the **NIFBus Dialog** icon in your **NI-FBUS** program group.
2. Click on the **Actions** menu and select **Open Session**.

or

Right-click on the **Open Descriptors** icon.

3. On the pop-up menu that appears, select **Open Session**.
4. In the Open Session dialog box that appears, click on the **Open Session** button. The NI-FBUS Dialog utility makes an `nifOpenSession` call to the NI-FBUS Communications Manager process. This call opens a session descriptor, which represents your connection to the NI-FBUS Communications Manager process.

If the call succeeds, the NI-FBUS Communications Manager process is running and responding to requests, and a new session descriptor is created under the **Open Descriptors** icon. If the call fails, make sure that your NI-FBUS Communications Manager process is running, and that it has not displayed any error message boxes during startup. You can check this by maximizing and looking at the `nifb.exe` console window.

5. Right-click on the session descriptor icon to see its pop-up menu.

or

Click on the **Session** icon and then select the **Actions** menu.

The list that appears represents the NI-FBUS Communications Manager API calls you can make with a session descriptor.

6. Choose the **GetInterfaceList** function from the list of choices. This choice displays the logical name of all known interfaces.
7. Highlight the interface name of your choice and click on the **OpenLink** button.
or
Open a link by choosing the **OpenLink** function and entering the interface name.
8. Right-click on the **Link** icon and choose **GetDeviceList**. The NI-FBUS Dialog utility displays a list of active devices on your fieldbus link. Your fieldbus interface board is also included in this list, as shown in Figure 3-1.

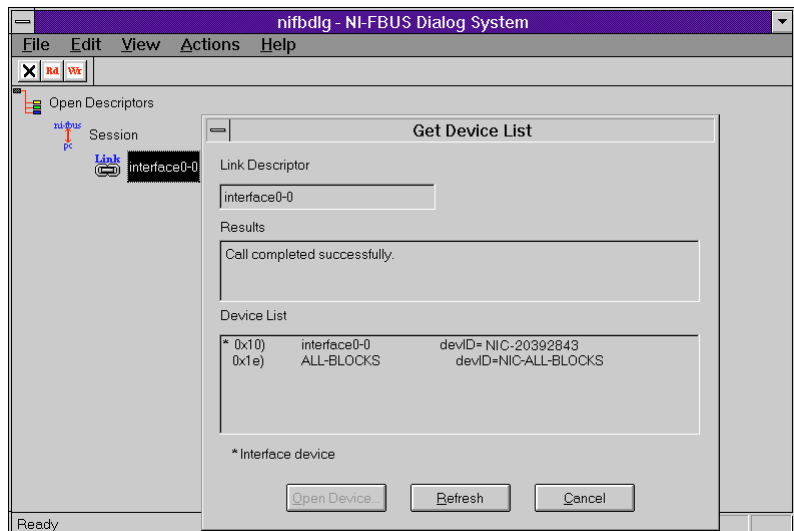


Figure 3-1. Getting a Device List

Downloading a Schedule to an Interface

Follow the steps in this section to use the NI-FBUS Dialog utility to download a schedule to an interface.

1. Complete all the steps of the previous example, *Getting a Device List*.
2. Select an interface board by clicking on an entry in the device list that has an asterisk (*) on its left.

3. Click on the **Open Device...** button. A new dialog box appears with the identifying information for the interface board already filled in.
4. Click on the **Open Device** button on the new dialog box. If the call completes successfully, a new icon for the device descriptor appears in the tree structure on the screen.
5. Right-click on the new device icon, and select the **DownloadLASSched** menu option. A new dialog box, shown in Figure 3-2, appears with identifying information for the device already filled in.
6. In the new dialog box, click **Browse** to locate your .ini file that contains the LAS schedule you want to download, or enter the full path to the file, if you know it.
7. Click the **Download** button. The NI-FBUS Communications Manager downloads the schedule to the interface board and activates it immediately.

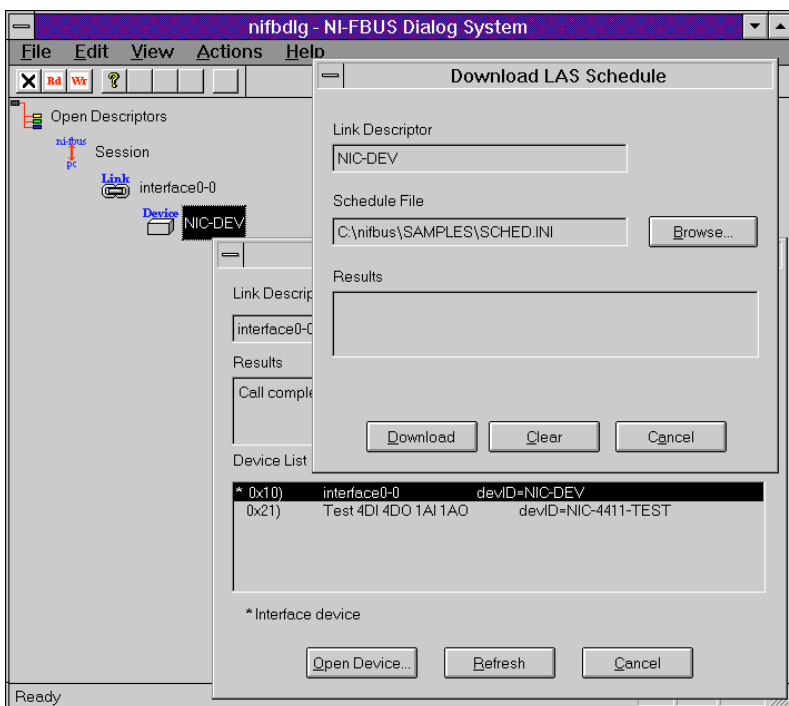


Figure 3-2. Downloading a Link Active Schedule

Reading a Parameter Using TAG.PARAM Access

Follow the steps in this section to read a parameter using *TAG.PARAM*-based access with the NI-FBUS Dialog utility.

1. Open the NI-FBUS Dialog utility.
2. Click on the **Actions** menu and select **Open Session**.
3. Click on the **Open Session** button. If the call succeeds, the NI-FBUS Communications Manager process is running and responding to requests, and a new session descriptor is created under the **Open Descriptors** icon.
4. Right-click on the session descriptor icon to see its popup menu.
5. Select the **ReadObject** menu item.
6. In the dialog box that appears (shown in Figure 3-3), enter the name of the parameter to read in the *BLOCKTAG.PARAM* format, where *BLOCKTAG* is the tag of the block containing the parameter, and *PARAM* is the name of the parameter. For example, to read the *out* parameter of an Analog Input block called FT-201, enter *FT-201.OUT*.
7. Click on the **Read** button to perform the read operation. If the call completes successfully, the NI-FBUS Dialog utility automatically determines the type of the data and displays it in the **Data** box. If the call fails, the error message appears in the **Result** box.

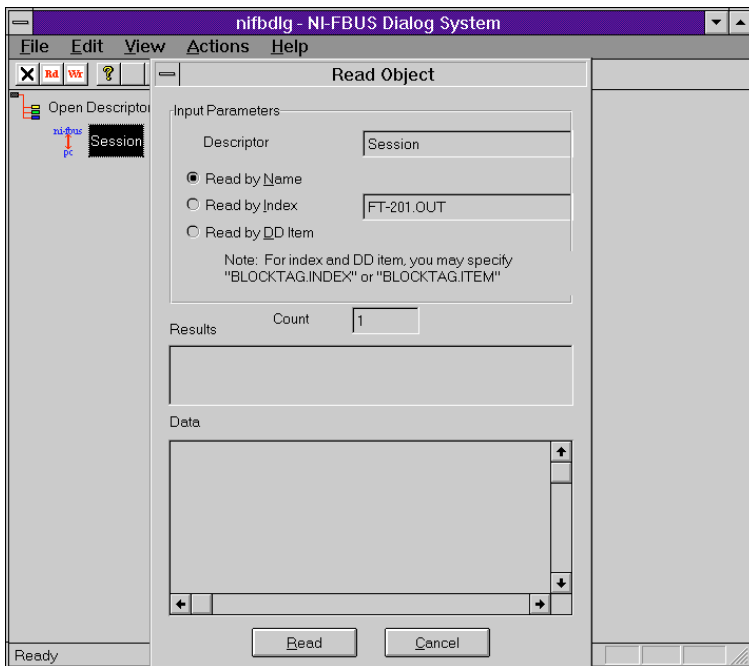


Figure 3-3. Reading a Parameter Using TAG.PARAMETER Access

Waiting for a Trend

Follow the steps in this section to wait for a trend using the NI-FBUS Dialog utility.



Note:

You will not be able to receive any trends unless you have configured a device to generate them, and configured an interface to receive them. You can use the NI-FBUS Configurator to accomplish both of these tasks.

1. Open the NI-FBUS Dialog utility.
2. Click on the **Actions** menu and select **Open Session**.
3. Click on the **Open Session** button. If the call succeeds, the NI-FBUS Communications Manager process is running and responding to requests, and a new session descriptor is created under the **Open Descriptors** icon.
4. Right-click on the session descriptor icon to see its popup menu.
5. Select the **WaitTrend** menu item.

6. The dialog box shown in Figure 3-4 appears. This dialog box waits until the NI-FBUS Communications Manager receives a trend from any device on the bus. The trend data is displayed in the **Results** box when the trend is received. The **Trend** dialog continues to wait for and display trends as they are received until you close it with the **Cancel** button.

You can wait on trends from all types of descriptors, not just session descriptors. For example, if you wait on a trend from a device descriptor, the dialog box only displays trends coming from the device that the specified descriptor represents. The same is true of link, VFD, and block descriptors.

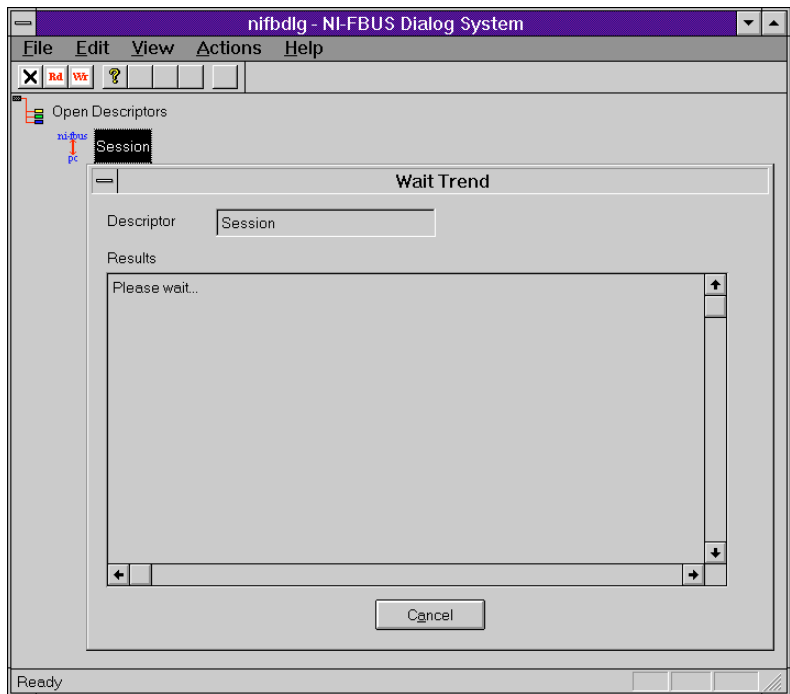
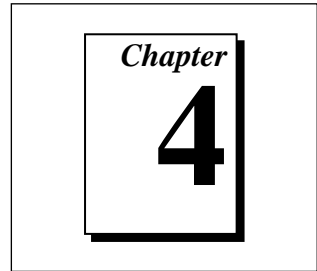


Figure 3-4. Waiting for a Trend

To exit the NI-FBUS Dialog utility, select **Exit** from the **File** menu.

NI-FBUS Interface Configuration Utility



This chapter explains how to use `fbconf`, the NI-FBUS Interface Configuration utility.

Introduction to the NI-FBUS Interface Configuration Utility

The NI-FBUS Interface Configuration utility is a screen-oriented, interactive program you can use to view the hardware configuration parameters for your fieldbus interfaces. In Windows NT, it is also the place where you edit your hardware configuration parameters, and add and delete interfaces. In Windows 95, these tasks are accomplished using various Windows 95 tools. You can use the Add New Hardware wizard in the Control Panel to add interface boards in Windows 95, and you can use the Device Manager under **Control Panel»System** to remove interface boards and change hardware configuration parameters.

In Windows NT and Windows 95, you can use the NI-FBUS Interface Configuration Utility to set certain low-level network software parameters as well.

The hardware configuration information is used primarily by the OS level driver; that is, `nifb.sys` for Windows NT, and `nifb.vxd` for Windows 95.

Starting the NI-FBUS Interface Configuration Utility

The NI-FBUS Interface Configuration utility automatically starts when the installer installs the NI-FBUS Communications Manager software. You can enter configuration information for the NI-FBUS Communications Manager software and the interface board(s) at installation time and after installation. To start the NI-FBUS Interface Configuration utility after installation, do the following:

- If you are using Windows 95 or Windows NT 4.0, select **Start»Programs»NI-FBUS»Interface Config**.
- If you are using Windows NT 3.51, double-click on the **NI-FBUS Interface Config** icon, which is part of the **NI-FBUS** program group, created in your **Program Manager** during installation.
- To use the command prompt, enter the command `fbconf.exe` to start the NI-FBUS Interface Configuration utility executable, which is located in the `utils` subdirectory of your NI-FBUS installation directory.

Using the NI-FBUS Interface Configuration Utility



Note:

If you are using an AT-FBUS board, you must know the base address and IRQ settings of your AT-FBUS hardware before you configure your NI-FBUS Communications Manager software so that you can configure these settings to match. To read your base address from your board switch settings, refer to the installation and configuration chapter in your getting started manual. To read your IRQ level from your board, look at the number printed on the board next to the jumper.

If you are using a PCMCIA-FBUS card, there are no physical hardware settings to match to NI-FBUS Interface Configuration utility settings; you can use any nonconflicting system resources.

Configuring a New Interface in Windows NT

To configure a new interface in Windows NT with the NI-FBUS Interface Configuration utility, complete the following steps:

1. Start the NI-FBUS Interface Configuration utility as described in the previous section, *Starting the NI-FBUS Interface Configuration*

Utility. On startup, the NI-FBUS Interface Configuration utility displays a dialog box with the **NI-FBUS** icon and buttons for **DD Info** and **Add a Board**.

2. Click on the **DD Info** button to add information about the base directory for the device descriptions.
3. Enter the full name, including the path, for the standard text dictionary file (which usually has a .dct extension). If you leave both of these fields at their default, the NI-FBUS Communications Manager uses a default Device Description that ships with the NI-FBUS Communications Manager software.
4. Click on the **Add a Board** button to configure hardware information for a single fieldbus interface connected to your machine. The hardware information you must fill in includes the physical address of your board, the board IRQ and the system bus type, as shown in Figure 4-1. ISA support and PCMCIA support are provided.

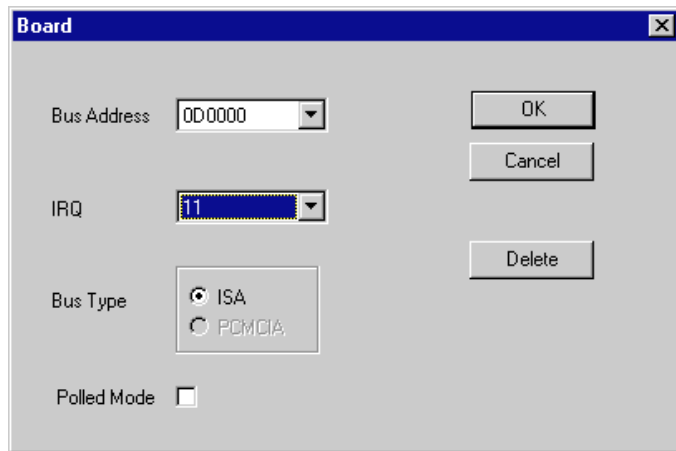


Figure 4-1. Assigning Board Information

You can also choose to use polled mode, without interrupts. In polled mode, the NI-FBUS Communications Manager periodically polls the board for service instead of using interrupts. Polled mode consumes more processor time, so you should only use it if there are no free IRQs in your system for the fieldbus hardware to use.

5. After checking the information you just entered, click on the **OK** button.

6. The dialog box for entering port information for the fieldbus interface appears. Proceed to the *Configuring Port Information* section, later in this chapter.

Configuring a New PCMCIA-FBUS in Windows 95

To configure a new PCMCIA-FBUS card in Windows 95, complete the following steps:

1. Install the hardware, as described in your getting started manual.
2. Reboot your computer.
3. Start the NI-FBUS Interface Configuration utility by selecting **Start»Programs»NI-FBUS»Interface Config**. The NI-FBUS driver automatically recognizes the newly added PCMCIA-FBUS card and displays it in the NI-FBUS Interface Configuration utility.
4. Proceed to the *Configuring Port Information* section, later in this chapter, for information on configuring the communication parameters and interface names .

Configuring a New AT-FBUS in Windows 95

To configure a new AT-FBUS for Windows 95, you must use the Add New Hardware Wizard to add your AT-FBUS and the Device Manager to configure resources for the AT-FBUS. You can then use the NI-FBUS Interface Configuration utility to configure the communication parameters and interface name as described in the following sections. Refer to Chapter 2, *Installation and Configuration*, in your getting started manual for detailed instructions on adding a new AT-FBUS and configuring resources, then proceed to the next section in this chapter, *Configuring Port Information*.

Configuring Port Information

After you add a new interface, configure the port information. In Windows NT, the port configuration dialog box appears when you click on the **OK** button after adding a new interface in the NI-FBUS Interface Configuration utility. In Windows 95, the port configuration dialog box appears when you select the port you want to configure in the NI-FBUS Interface Configuration utility and click on the **Edit** button.

Figure 4-2 shows the port configuration dialog box.

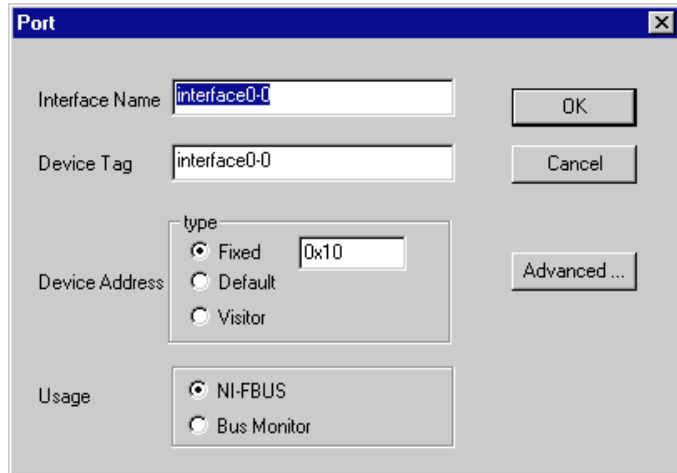
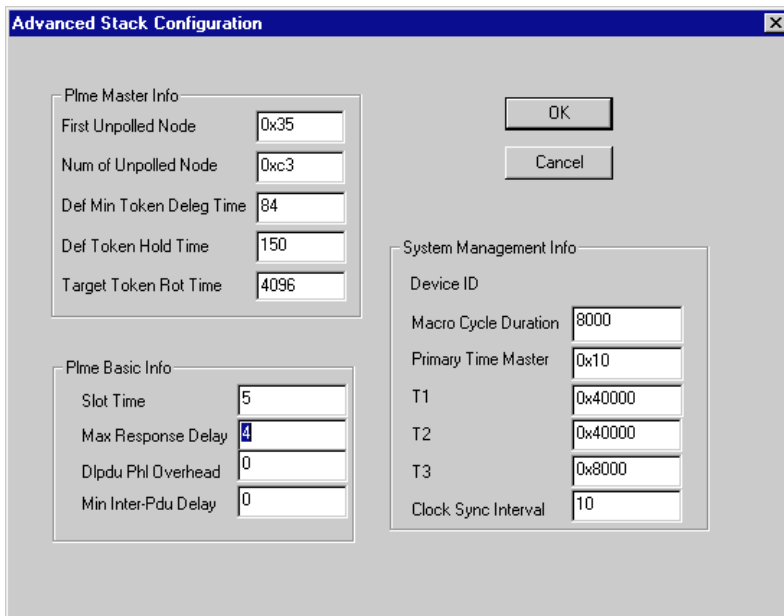


Figure 4-2. Assigning Port Information

Port configuration information you must enter includes the following:

- Interface (logical) name that the NI-FBUS Communications Manager software will use to refer to this port
- Interface device tag
- Interface device address
- Software that this port is for: the NI-FBUS Communications Manager or the NI-FBUS Monitor
- Other communication configuration parameters

To change these other configuration parameters, click on the **Advanced** button on the dialog box for the port you want to configure. Figure 4-3 shows the **Advanced Stack Configuration** dialog box.



The dialog box is titled "Advanced Stack Configuration" and contains several input fields and buttons. It is organized into three main sections:

- Prime Master Info:**
 - First Unpolled Node: 0x35
 - Num of Unpolled Node: 0xc3
 - Def Min Token Deleg Time: 84
 - Def Token Hold Time: 150
 - Target Token Rot Time: 4096
- Prime Basic Info:**
 - Slot Time: 5
 - Max Response Delay: 4
 - Dlpdu Pbl Overhead: 0
 - Min Inter-Pdu Delay: 0
- System Management Info:**
 - Device ID: (empty)
 - Macro Cycle Duration: 8000
 - Primary Time Master: 0x10
 - T1: 0x40000
 - T2: 0x40000
 - T3: 0x8000
 - Clock Sync Interval: 10

Buttons for "OK" and "Cancel" are located in the upper right area of the dialog.

Figure 4-3. Advanced Stack Configuration Dialog Box

Check the information you entered and then click on the **OK** button to save this configuration information for the board and port. The NI-FBUS call `nifGetInterfaceList` returns a list of the logical interface names you enter here.

You can connect multiple interface boards to your machine and configure them by repeating the procedure just described.

Under Windows NT, if you configure a new interface, you must stop and restart the `nifb.sys` driver so that the new configuration will take effect. You can stop the driver by typing `net stop nifb` at the command prompt, and start the driver by typing `net start nifb` at the command prompt.

Changing or Deleting Existing Interface Information in Windows NT

To delete or change information about any interface that you have already entered, complete the following steps:

1. Click on the relevant **Board x** icon, where x refers to the board number of the board you want to configure.

- Choose the **Edit** option from the dialog box that appears. You can edit the board configuration information that you entered earlier, or delete this board entirely.

If you delete an interface, the NI-FBUS Interface Configuration utility renumbers all the remaining interfaces. For example, if you delete **Board0**, it appears that you deleted the last interface, because all the remaining interface numbers are decreased by one.

To change information about a port on an interface that you have already configured, complete the following steps:

- Click on the relevant port icon, as shown in Figure 4-4.

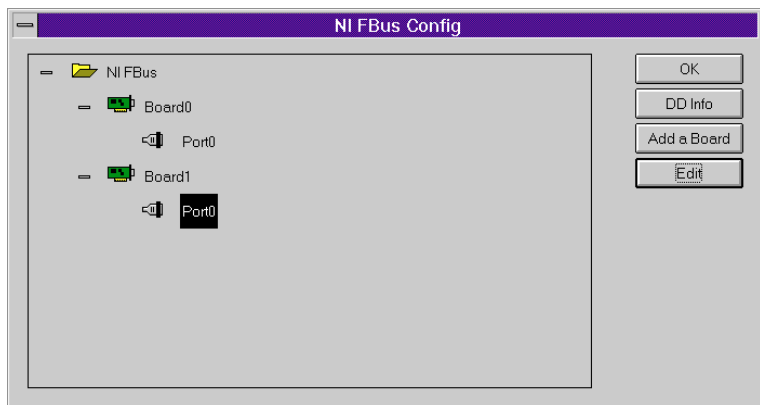


Figure 4-4. Choosing a Port

- Choose the **Edit** option. You can edit only the port configuration option. There is no option to delete a port after you configure it.

To exit the NI-FBUS Interface Configuration utility, click **OK** on the main screen of the utility.



Note:

Under Windows NT, if you change or delete the interface configuration, you must stop and restart the `nifb.sys` driver so that the new configuration will take effect. You can stop the driver by typing `net stop nifb` at the command prompt, and start the driver by typing `net start nifb` at the command prompt.

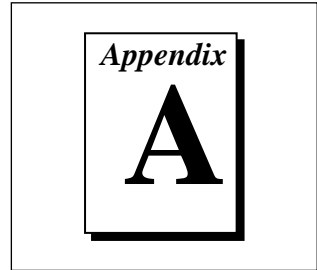
Changing or Deleting Existing Interface Information in Windows 95

To change the hardware resources in Windows 95, you must use the Windows 95 Device Manager rather than the NI-FBUS Interface Configuration utility. For more information, refer to the *Changing the Resources to Eliminate Conflicts* section in the installation and configuration chapter of your getting started manual.

To delete a National Instruments fieldbus interface in Windows 95, complete the following steps.

1. Double-click on the **System** icon under **Start»Settings»Control Panel**.
2. Click on the **Device Manager** tab.
3. Click on the name of your interface under the heading **Fieldbus Adapters**.
4. Click on the **Remove** button.
5. Run the NI-FBUS Interface Configuration utility once to make sure the interface has been removed.

Configuring the Link Active Schedule File



This appendix contains information about how to configure your Link Active Schedule file.

If you want to do scheduling and use publishers and subscribers, you must follow the instructions in this appendix; otherwise, you do not need this information.

Introduction to the Link Active Schedule File

You must download the Link Active Schedule file to your fieldbus interface before the board can have Link Active Scheduler functionality on the fieldbus network. You may ignore this appendix if there is no schedule, or if the schedule is downloaded over the network to your fieldbus interface.

Save the Link Active Schedule file as a .ini file. You can download this file to your interface board using the NI-FBUS Dialog utility.

For detailed information about the parameters in the Link Active Schedule file, refer to the *Data Link Layer* section of the Final Specification version of the *FOUNDATION Fieldbus Specification*.

Format of the Link Active Schedule File

Create your Link Active Schedule file according to the following format.

The names of the sections of the Link Active Schedule file are:

[Schedule Summary]

...

[Subschedule 1]

```

...
[Sequence 1-1]
...
[Sequence 1-n]
...
[Subschedule x]
...
[Sequence x-1]
...
[Sequence x-y]
...

```

The general line format for all other lines is:

VARIABLE=VALUE

where the valid variable names and values are defined in Tables A-1 to A-4.

Table A-1. Valid Variable Names and Values for the Schedule Summary Section

Variable Name	Valid Values	Implied Units	Default
encodingVersionNumber	0-7	none	none
versionNumber	0x0-0xffff	none	none
builderIdentifier	0x100-0xffff	none	none
numSubSchedules	0-255	none	none
maxSchedulingOverhead	0x0-0x3f	octets	none
macroCycle	0x0-0xffffffff	1/32 ms	none

Table A-2. Valid Variable Names and Values for the Subschedule Section

Variable Name	Valid Values	Implied Units	Default
period	0x0- 0xffffffff	1/32 ms	none
numSequence	0-255	none	none

Table A-3. Valid Variable Names and Values for the Sequence Section

Variable Name	Valid Values	Implied Units	Default
maxDuration	0x0-0xffff	1/32 ms	none
numElement	0-255	none	none

For the variables in Table A-4, *N* is an integer between 1 and numElement. Repeat these variables within this subschedule section exactly numElement times.

Table A-4. Valid Variable Names Including the Variable *N* and Values for the Sequence Section

Variable Name	Valid Values	Implied Units	Default
priority N	TIMEAVAILABLE URGENT NORMAL	none	none
address N	Parameter name in <i>TAG.PARAM</i> format, or DLCEP (Data Link Connection End Point) in 0xNNNN format	none	none

Customer Communication

For your convenience, this appendix contains forms to help you gather the information necessary to help us solve technical problems you might have as well as a form you can use to comment on the product documentation. Filling out a copy of the *Technical Support Form* before contacting National Instruments helps us help you better and faster.

National Instruments provides comprehensive technical assistance around the world. In the U.S. and Canada, applications engineers are available Monday through Friday from 8:00 a.m. to 6:00 p.m. (central time). In other countries, contact the nearest branch office. You may fax questions to us at any time.

Electronic Services



Bulletin Board Support

National Instruments has BBS and FTP sites dedicated for 24-hour support with a collection of files and documents to answer most common customer questions. From these sites, you can also download the latest instrument drivers, updates, and example programs. For recorded instructions on how to use the bulletin board and FTP services and for BBS automated information, call (512) 795-6990. You can access these services at:

United States: (512) 794-5422

Up to 14,400 baud, 8 data bits, 1 stop bit, no parity

United Kingdom: 01635 551422

Up to 9,600 baud, 8 data bits, 1 stop bit, no parity

France: 1 48 65 15 59

Up to 9,600 baud, 8 data bits, 1 stop bit, no parity



FTP Support

To access our FTP site, log on to our Internet host, `ftp.natinst.com`, as anonymous and use your Internet address, such as `joesmith@anywhere.com`, as your password. The support files and documents are located in the `/support` directories.



Fax-on-Demand Support

Fax-on-Demand is a 24-hour information retrieval system containing a library of documents on a wide range of technical information. You can access Fax-on-Demand from a touch-tone telephone at (512) 418-1111.



E-Mail Support (currently U.S. only)

You can submit technical support questions to the applications engineering team through e-mail at the Internet address listed below. Remember to include your name, address, and phone number so we can contact you with solutions and suggestions.

support@natinst.com

Telephone and Fax Support

National Instruments has branch offices all over the world. Use the list below to find the technical support number for your country. If there is no National Instruments office in your country, contact the source from which you purchased your software to obtain support.



Telephone



Fax

Australia	03 9879 5166	03 9879 6277
Austria	0662 45 79 90 0	0662 45 79 90 19
Belgium	02 757 00 20	02 757 03 11
Canada (Ontario)	905 785 0085	905 785 0086
Canada (Quebec)	514 694 8521	514 694 4399
Denmark	45 76 26 00	45 76 26 02
Finland	09 725 725 11	09 725 725 55
France	01 48 14 24 24	01 48 14 24 14
Germany	089 741 31 30	089 714 60 35
Hong Kong	2645 3186	2686 8505
Israel	03 5734815	03 5734816
Italy	02 413091	02 41309215
Japan	03 5472 2970	03 5472 2977
Korea	02 596 7456	02 596 7455
Mexico	5 520 2635	5 520 3282
Netherlands	0348 433466	0348 430673
Norway	32 84 84 00	32 84 86 00
Singapore	2265886	2265887
Spain	91 640 0085	91 640 0533
Sweden	08 730 49 70	08 730 43 70
Switzerland	056 200 51 51	056 200 51 55
Taiwan	02 377 1200	02 737 4644
U.K.	01635 523545	01635 523154

Technical Support Form

Photocopy this form and update it each time you make changes to your software or hardware, and use the completed copy of this form as a reference for your current configuration. Completing this form accurately before contacting National Instruments for technical support helps our applications engineers answer your questions more efficiently.

If you are using any National Instruments hardware or software products related to this problem, include the configuration forms from their user manuals. Include additional pages if necessary.

Name _____

Title _____

Company _____

Address _____

Fax (____) _____ Phone (____) _____

Computer brand _____ Model _____ Processor _____

Operating system (include version number) _____

Clock Speed _____ MHz RAM _____ MB Display adapter _____

Mouse ____ yes ____ no Other adapters installed _____

Hard disk capacity _____ MB Brand _____

Instruments used _____

National Instruments hardware product model _____ Revision _____

Configuration _____

National Instruments software product _____ Version _____

Configuration _____

The problem is _____

List any error messages _____

The following steps will reproduce the problem _____

Hardware and Software Configuration Form

Record the settings and revisions of your hardware and software on the line to the right of each item. Complete a new copy of this form each time you revise your software or hardware configuration, and use this form as a reference for your current configuration. Completing this form accurately before contacting National Instruments for technical support helps our applications engineers answer your questions more efficiently.

National Instruments Products

Hardware Revision _____

Interrupt Level of Hardware _____

DMA Channels of Hardware _____

Base I/O Address of Hardware _____

NI-FBUS Communications Manager Software Version _____

Other Products

Computer Make and Model _____

Microprocessor _____

Clock Frequency _____

Type of Video Board Installed _____

Operating System _____

Operating System Version _____

Operating System Mode _____

Programming Language _____

Programming Language Version _____

Other Boards in System _____

Base I/O Address of Other Boards _____

DMA Channels of Other Boards _____

Interrupt Level of Other Boards _____

Documentation Comment Form

National Instruments encourages you to comment on the documentation supplied with our products. This information helps us provide quality products to meet your needs.

Title: *NI-FBUS™ Communications Manager User Manual for Windows 95 and Windows NT*

Edition Date: July 1997

Part Number: 321287B-01

Please comment on the completeness, clarity, and organization of the manual.

If you find errors in the manual, please record the page numbers and describe the errors.

Thank you for your help.

Name _____

Title _____

Company _____

Address _____

Phone (____) _____

Mail to: Technical Publications
National Instruments Corporation
6504 Bridge Point Parkway
Austin, TX 78730-5039

Fax to: Technical Publications
National Instruments Corporation
(512) 794-5678

A rectangular box containing a smaller, slightly offset rectangular box with the word "Glossary" written in a serif font inside it.

Glossary

Prefix	Meaning	Value
n-	nano-	10^{-9}
μ -	micro-	10^{-6}
m-	milli-	10^{-3}

A

A	Amperes.
administrative function	An NI-FBUS function that deals with administrative tasks, such as returning descriptors and closing descriptors.
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 an event.
alert function	A function that receives or acknowledges an alert.
alert objects	Objects used for reporting of alarms and events.
analog network	A network that carries signals in analog form as a continuously varying range of electrical voltage or current.
Application Programmer Interface (API)	A message format that an application uses to communicate with another entity that provides services to it.

B

basic device	A device that can communicate on the fieldbus, but cannot become the LAS.
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 VFD.
block level	The level of an NI-FBUS call that accepts a block descriptor.

C

Communication Stack	The hierarchy of layers in a layered communications model that performs the services required to interface the User Application to the Physical Layer of the fieldbus.
Connection Management	The service the NI-FBUS Communications Manager provides by handling Virtual Communication Management Relationships (VCRs).
control loop	A set of connections between blocks used to perform a control algorithm.
Core Function	The basic functions that the NI-FBUS Communications Manager software performs, such as reading and writing block parameters.

D

Data Link Layer	The second lowest layer, layer two in the ISO seven layer model. The Data Link Layer splits data into frames to send on the physical layer, receives acknowledgment frames, and re-transmits frames if they are not received correctly. It also performs error checking to maintain a sound virtual channel to the next layer.
DCS	Distributed Control System.
DD	See <i>Device Description</i> .

descriptor	A number that the NI-FBUS Communications Manager returns to the application. The descriptor is used to specify a target for future NI-FBUS calls.
deterministic communication	A synonym for scheduled/cyclic communication.
Device Description (DD)	A machine-readable description of all the blocks and block parameters of a device.
Device Description Language (DDL)	A formal programming language that defines the parameters of the blocks. It also defines attributes of parameters and blocks like help strings in different languages, ranges of values for parameters, and so on.
Device Description Services (DDS)	A set of functions that applications use to access Device Descriptions.
device ID	An identifier for a device that the manufacturer assigns. Device IDs must be unique to the device; no two devices can have the same device ID.
device level	An NI-FBUS call that accepts a physical device descriptor.
distributed control	Process control distributed among several devices connected by network.
driver	Device driver software installed within the operation system.
Dynamic Link Library	A library of functions and subroutines that links to an application at run time.

E

entity	A certain thing, such as a process, object, device, or event.
Ethernet	A recognized standard local area network that uses coaxial cable.

F

fieldbus	An all-digital, two-way communication system that connects control systems to instrumentation.
----------	--

Fieldbus Access Sublayer (FAS)	The layer of the communication stack that provides an interface between the DLL and layer 7 of the OSI model. The FAS provides communication services such as client/server, publisher/subscriber and event distribution.
Fieldbus Foundation	The organization that developed a fieldbus network specifically based upon the work and principles of the ISA/IEC standards committees.
Fieldbus Messaging Specification (FMS)	The layer of the communication stack that defines a model for applications to interact over the fieldbus. The services FMS provides allow you to read and write information about the OD, read and write the data variables described in the OD, and perform other activities such as uploading/downloading data, and invoking programs inside a device.
Fieldbus Network Address	Location of a board or device on the fieldbus; the fieldbus node address.
FIP	Factory Instrumentation Protocol.
FOUNDATION Fieldbus	The communications network specification that the Fieldbus Foundation created.
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.

H

hard code	To permanently establish something that should be variable in a program.
header file	A C language source file containing important definitions and function prototypes.

I

IEC Standards Committee	International Electrotechnical Commission Standards Committee. A technical standards committee which is at the same level as the ISO.
-------------------------	---

ISO International Organization for Standardization. A technical standards organization that creates international technical standards for computers and communications. The ISO is composed of national standards organizations in 89 countries. The American National Standards Institute (ANSI) represents the United States in the ISO.

K

kernel The set of programs in an operating system that implements basic system functions.

kernel mode The mode in which device drivers run on Windows NT.

L

link A group of fieldbus devices connected across a single wire pair with no intervening bridges.

Link Active Schedule A schedule of times in the macrocycle when devices must publish their output values on the fieldbus.

Link Active Scheduler (LAS) 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 identifier A number that specifies a link.

Link Master device A device that is capable of becoming the LAS.

link object An object resident in a device that defines connections between function block input and output across the network. Link objects also specify trending connections.

Local Area Network A communications network that is limited in physical spatial area for the purpose of easier connection of computers in neighboring buildings.

M

MMI (Man-Machine Interface) High-level supervisory data acquisition and display programs.

N

Network Management	A layer of the FOUNDATION Fieldbus communication stack that contains objects that other layers of the communication stack use, such as Data Link, FAS, and FMS. You can read and write SM and NM objects over the fieldbus using FMS Read and FMS Write services.
NI-FBUS Communications Manager process	The Windows 95 or Windows NT task that implements the NI-FBUS Communications Manager API.
non-scheduled/acyclic communication	Communication that occurs at times that are not predetermined.
non-volatile memory	Memory that does not require electricity to hold data.

O

object	An element of an object dictionary.
Object Dictionary (OD)	A structure in a device that describes data that can be communicated on the fieldbus. The OD is a lookup table that gives information such as data type and units about a value that can be read from or written to a device.
operator acknowledgment alarm	The verification an operator performs when he or she receives a fieldbus message.
OSI Model	Open Systems Interconnect Layered Communication Model. A communications protocol standard that the ISO created. It establishes a seven-layered framework for implementing protocols. In the OSI model, control moves from one layer to the next in the following manner: control starts at the top layer in one station, moves through all protocol layers to the bottom layer, then goes over the channel to the next station and moves back up through all protocol layers.

P

parameter	One of a set of network-visible values that makes up a function block.
PC	Personal Computer.
physical device	A single device residing at a unique address on the fieldbus.
physical device tag	A user-defined name for a physical device.
Physical Layer	The layer of the communication stack that converts digital fieldbus messages from the communication stack to actual physical signals on the fieldbus transmission medium and vice versa.
PID	Proportional/Integral/Derivative. A common control function block algorithm that uses proportions, integrals, and derivatives in calculation.
poll	To repeatedly inspect a variable or function block to acquire data.

R

resource block	A block that describes general characteristics of a device, such as manufacturer and device name. Only one resource block per device is allowed.
----------------	--

S

scheduled/cyclic communications	Communication that occurs at the same time during each control cycle.
session	A communication path between an application and the NI-FBUS Communications Manager.
session level	A category of NI-FBUS API calls that accepts a session descriptor.
static library	A library of functions/subroutines that you must link to your application as one of the final steps of compilation, as opposed to a Dynamic Link Library, which links to your application at run time.

System Management A layer of the FOUNDATION Fieldbus communication stack that assigns addresses and physical device tags, maintains the function block Schedule for the function blocks in that device, and distributes application time. You can also locate a device or a function block tag through SM.

T

TCP/IP Transmission Control Protocol/Internet Protocol. The communications protocol used on the Internet.

thread An operating system object that consists of a flow of control within a process. In some operating systems, a single process can have multiple threads, each of which can access the same data space within the process. However, each thread has its own stack and all threads can execute concurrently with one another (either on multiple processors, or by time-sharing a single processor).

Tokenizer A program the Fieldbus Foundation provides that creates a binary form of DDL code to ship to an end user with an instrument.

transducer block A block that is an interface to the 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.

trend function An NI-FBUS call related to trends.

U

User Layer The network layer of the communication stack above layer seven in the OSI. The User Layer defines blocks and objects that represent the functions and data available in a device.

V

VCR	Virtual Communication Relationship. Preconfigured or negotiated connections between virtual Field devices on a network.
view objects	Predefined groupings of parameter sets that MMI applications use.
Virtual Field Device (VFD)	A model for remotely viewing data described in the object dictionary.

A graphic of a book with the word "Index" written on its cover, enclosed in a rectangular border.

Index

A

- administrative functions, 2-1 to 2-2
 - example, 2-2
 - purpose and use, 2-1
- alert functions, 2-3 to 2-4
- alert objects, 1-7
- application development, 2-1 to 2-10
 - communication level choice, 2-6
 - compiling, linking, and running, 2-10
 - multi-threaded vs. single-threaded access, 2-7 to 2-8
 - multi threading, 2-8
 - single threading, 2-7
- NI-FBUS Communications Manager process, 2-5 to 2-6
- NI-FBUS Dialog utility for communicating with devices, 2-8
- NI-FBUS functions, 2-1 to 2-4
 - administrative functions, 2-1 to 2-2
 - alert and trend functions, 2-3 to 2-4
 - core functions, 2-2 to 2-3
 - Device Description functions, 2-4
- sample application files
 - nifbdd.c, 1-11
 - nifb_mt.c, 1-11
 - nifbtest.c, 1-11
 - sched.ini, 1-11

- tag-based vs. index-based access, 2-7
- writing applications, 2-9

B

- binaries, NI-FBUS Communications Manager software
 - ffstack.bin, 1-10
 - nifb.exe, 1-10
- bulletin board support, B-1

C

- communication level, choosing, 2-6
- Communication Stack, 1-4
 - Data Link Layer, 1-4
 - Fieldbus Access Sublayer (FAS), 1-4
 - Fieldbus Messaging Specification (FMS) layer, 1-4
 - Network Management (NM) layer, 1-4
 - Object Dictionary (OD), 1-4
 - System Management (SM) layer, 1-4
 - Virtual Field Device (VFD), 1-4
- compiling NI-FBUS applications, 2-10
- configuration. *See* Link Active Schedule file, configuring; NI-FBUS Interface Configuration utility.
- core functions, 2-2 to 2-3
 - example, 2-3
 - purpose and use, 2-2

customer communication, *xii*, B-1 to B-8

D

Data Link Layer, 1-4

developing applications.

See application development.

Device Description functions, 2-4

Device Description Language (DDL), 1-8

Device Description Services (DDS), 1-8

Device Descriptions (DDs), 1-8

device list, obtaining with NI-FBUS Dialog utility, 3-2 to 3-3

Dialog utility. *See* NI-FBUS Dialog utility.

documentation

conventions used in manual, *xii*

how to use manual set, *ix* to *x*

organization of manual, *x*

related documentation, *xii* to *xiii*

downloading schedule to interface, 3-3 to 3-4

driver

definition, 1-10

nifb.sys, 1-10, 4-1

nifb.vxd, 1-10, 4-1

Dynamic Link Libraries, 1-10 to 1-11

drvintf.dll, 1-11

nifb.dll, 1-10

E

electronic support services, B-1 to B-2

e-mail support, B-2

F

Fax-on-Demand support, B-2

fieldbus. *See also* FOUNDATION Fieldbus;
NI-FBUS Communications Manager
software.

benefits, 1-1

definition, 1-1

Fieldbus Access Sublayer (FAS), 1-4

Fieldbus Foundation, 1-2. *See also*
FOUNDATION Fieldbus.

Fieldbus Messaging Specification (FMS)
layer, 1-4

FOUNDATION Fieldbus

Communication Stack, 1-4

Data Link Layer, 1-4

Fieldbus Access Sublayer
(FAS), 1-4

Fieldbus Messaging Specification
(FMS) layer, 1-4

Network Management (NM)
layer, 1-4

Object Dictionary (OD), 1-4

System Management (SM)
layer, 1-4

components, 1-2

Device Description Language
(DDL), 1-8

Device Descriptions (DDs), 1-8

fieldbus model compared to OSI
7-layer model (figure), 1-3

Link Active Scheduler (LAS), 1-7

Link Masters, 1-7

links, 1-7

overview, 1-2 to 1-3

Physical Layer, 1-3

types of devices, 1-7

User Layer, 1-5 to 1-7

function blocks, 1-6

illustration, 1-5

objects, 1-7

purpose, 1-5

resource block, 1-5

transducer blocks, 1-6

Virtual Field Devices (VFDs), 1-4, 1-8

FTP support, B-1

function blocks, User Layer

definition, 1-6

standard function blocks (table), 1-6
 functions. *See* NI-FBUS functions.

H

header files, NI-FBUS Communications
 Manager software, 1-12

I

index-based access, 2-7
 interface configuration. *See* NI-FBUS
 Interface Configuration utility.

L

Link Active Schedule file, configuring, A-1
 to A-3
 format, A-1 to A-3
 names of sections, A-1 to A-2
 overview, A-1
 valid variable names and values, A-2
 to A-3
 Schedule Summary section
 (table), A-2
 Sequence section (table), A-3
 Subschedule section (table), A-3
 Link Active Scheduler (LAS), 1-27
 Link Masters, 1-7
 link objects, 1-7
 linking NI-FBUS applications, 2-10
 links, 1-7

M

manual. *See* documentation.
 multi-threaded access, 2-8

N

net start nifb command, 4-6, 4-7

net stop nifb command, 4-6, 4-7

Network Management (NM) layer, 1-4

nifAcknowledgeAlarm function, 2-4

NI-FBUS Interface Configuration utility,
 4-1 to 4-6

 AT-FBUS board

 configuring in Windows 95, 4-4
 note, 4-2

 changing or deleting interface

 information, 4-6 to 4-8

 Windows 95, 4-8

 Windows NT, 4-6 to 4-7

 new interface configuration, 4-2 to 4-4

 AT-FBUS in Windows 95, 4-4

 PCMCIA-FBUS in Windows 95,
 4-4

 Windows 95, 4-4

 Windows NT, 4-2 to 4-4

 overview, 4-1

 PCMCIA-FBUS card

 configuring in Windows 95, 4-4
 note, 4-2

 starting, 4-2

 using, 4-2 to 4-8

NI-FBUS Dialog utility, 2-8, 3-1 to 3-7

 application development, 2-8

 downloading schedule to interface,
 3-3 to 3-4

 examples, 3-2 to 3-7

 getting device list, 3-2 to 3-3

 overview, 3-1 to 3-2

 reading parameter using TAG.PARAM
 access, 3-5 to 3-6

 waiting for trends, 3-6 to 3-7

NI-FBUS functions, 2-1 to 2-4

 administrative functions, 2-1 to 2-2

 alert and trend functions, 2-3 to 2-4

 core functions, 2-2 to 2-3

 Device Description functions, 2-4

NI-FBUS Communications Manager
interface
 advantages and uses, 1-9 to 1-10
 configuring. *See* NI-FBUS Interface Configuration utility.
 overview, 1-9
NI-FBUS Communications Manager process, 2-5 to 2-6
NI-FBUS Communications Manager software, 1-10 to 1-11. *See also* fieldbus; FOUNDATION Fieldbus.
 binaries, 1-10
 components, 1-10 to 1-12
 driver, 1-10
 nifb.sys, 1-10
 nifb.vxd, 1-10
 Dynamic Link Libraries, 1-10 to 1-11
 header files, 1-12
 readme.txt file, 1-12
 sample application files, 1-11
 static library, 1-11
 utilities, 1-11
nifGetBlockList function, 2-1
nifGetDeviceList function, 2-1, 2-8, 2-9
nifGetObjectAttributes function, 2-2, 2-4
nifGetObjectSize function, 2-2
nifGetVfdList function, 2-1
nifOpen function, 2-2
nifOpenBlock function, 2-2
nifOpenSession function, 2-9
nifReadObject function, 2-2, 2-3
nifWaitAlert function, 2-4, 2-8, 2-9
nifWaitTrend function, 2-4, 2-8, 2-9
nifWriteObject function, 2-2, 2-3

O

Object Dictionary (OD), 1-4
objects, User Layer, 1-7

P

parameter, reading with TAG.PARAM
 access, 3-5 to 3-6
Physical Layer, 1-3
polled mode, 4-3
port configuration, 4-4 to 4-6
programming. *See* application development.

R

readme.txt file, 1-12
resource block, User Layer, 1-5
running NI-FBUS applications, 2-10

S

sample application files
 nifbdd.c, 1-11
 nifb_mt.c, 1-11
 nifbtest.c, 1-11
 sched.ini, 1-11
schedule, downloading, 3-3 to 3-4
Schedule, Link Active. *See* Link Active Schedule file, configuring; Link Active Scheduler (LAS).
single-threaded access, 2-7
static library, NI-FBUS Communications Manager software, 1-11
System Management (SM) layer, 1-4

T

tag-based access, 2-7
TAG.PARAM access for reading parameters, 3-5 to 3-6
technical support, B-1 to B-2
telephone and fax support, B-2
transducer blocks, User Layer, 1-6
trend, waiting for, 3-6 to 3-7
trend functions, 2-3 to 2-4

trend objects, 1-7

U

User Layer, 1-5 to 1-7

- function blocks, 1-6

- illustration, 1-5

- objects, 1-7

- purpose, 1-5

- resource block, 1-5

- transducer blocks, 1-6

utilities, NI-FBUS Communications Manager

- software

 - fbconf.exe, 1-11. *See also* NI-FBUS

 - Interface Configuration utility;

 - NI-FBUS Dialog utility.

 - nifbldg.exe, 1-11

V

view objects, 1-7

Virtual Field Devices (VFDs), 1-4, 1-8

W

waiting for trends, 3-6 to 3-7