ARTISAN®
TECHNOLOGY GROUP

Your **definitive** source for quality pre-owned equipment.

Artisan Technology Group
(217) 352-9330 | sales@artisantg.com | artisantg.com

Full-service, independent repair center
with experienced engineers and technicians on staff.

We buy your excess, underutilized, and idle equipment
along with credit for buybacks and trade-ins.

Custom engineering
so your equipment works exactly as you specify.

- Critical and expedited services
- In stock / Ready-to-ship
- Leasing / Rentals / Demos
- ITAR-certified secure asset solutions

Expert team | Trust guarantee | 100% satisfaction

All trademarks, brand names, and brands appearing herein are the property of their respective owners.

Find the *National Instruments GPIB-ENET/100* at our website: *Click HERE*

ESP-488
Software Reference Manual for the GPIB-ENET

February 1995 Edition

Part Number 320910A-01

© Copyright 1995 National Instruments Corporation.
All Rights Reserved.

National Instruments Corporate Headquarters
6504 Bridge Point Parkway
Austin, TX 78730-5039
(512) 794-0100
Technical support fax: (800) 328-2203
(512) 794-5678

Branch Offices:
Australia (03) 879 9422, Austria (0662) 435986, Belgium 02/757.00.20,
Canada (Ontario) (519) 622-9310, Canada (Québec) (514) 694-8521,
Denmark 45 76 26 00, Finland (90) 527 2321, France (1) 48 14 24 24,
Germany 089/741 31 30, Italy 02/48301892, Japan (03) 3788-1921,
Mexico 95 800 010 0793, Netherlands 03480-33466, Norway 32-84 84 00,
Singapore 2265886, Spain (91) 640 0085, Sweden 08-730 49 70,
Switzerland 056/20 51 51, Taiwan 02 377 1200, U.K. 0635 523545

Limited Warranty

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

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

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

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

failure or surges, fire, flood, accident, actions of third parties, or other events outside reasonable control.

Copyright

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

Trademarks

NI-488.2™ 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.

Contents

About This Manual ix
Organization of This Manual ix
Conventions Used in This Manual x
Related Documentation xi
Customer Communication xi

Chapter 1
Introduction 1-1
What Your Kit Should Contain 1-1
Optional Equipment 1-1
GPIB-ENET Hardware Kits 1-1
Ethernet Cables 1-1
Shielded GPIB Cables 1-2
Hardware Description 1-2
Software Description 1-3
File Description 1-4

Chapter 2
Configuring the Hardware 2-1
Step 1. Determine IP and Ethernet Addresses 2-1
Step 2. Configure the GPIB-ENET Operating Characteristics 2-2
Configure the Slide Switch 2-2
Configure the 8-Bit DIP Switch 2-3
Step 3. Connect the Cables 2-3
Step 4. Switch on Your GPIB-ENET 2-4
Step 5. Configure Subnet Information 2-4

Chapter 3
The ESP-488 C Language Library 3-1
Global Variables 3-1
Status Variable: ibsta 3-2
Error Variable: iberr 3-6
Count Variable: ibcntl 3-11
ESP-488 Function Descriptions 3-11

Chapter 4
ESP-488 Functions 4-1
Function Names 4-1
Purpose 4-1
Input and Output 4-1
Description 4-1
Possible Errors 4-1
Examples 4-1

ibcmd 4-2
ibconfig 4-4
ibfind 4-13
ibonl 4-15
ibrd 4-17
ibrpp 4-19
ibrsp 4-20
ibsic 4-22
ibwait 4-23
ibwrt 4-26

Chapter 5
Using Your ESP-488 Software 5-1
Helpful Source Code Rules 5-1
Programming Considerations 5-2
Compiling a C Program with the ESP-488 Package 5-2
ESP-488 Example 5-3

Chapter 6
Verification and Troubleshooting 6-1
Verify the Hardware Installation 6-1
Troubleshooting Hardware Problems 6-1
Common Questions 6-2

Appendix A
Multiline Interface Messages A-1

Appendix B
Hardware Specifications B-1

Appendix C
GPIB-ENET 8-Bit DIP Switch C-1
8-Bit DIP Switch Descriptions C-1

Appendix D
GPIB-ENET Configuration Utilities D-1
IPassign Utility D-1
Update Utility D-2
IPsetup Utility D-3

Appendix E
READY LED Signaling E-1
READY LED Overview E-1
Step 1. Count the Long Flashes E-1
Step 2. Count the Short Flashes E-2
Step 3. Record Your Status Code Number E-2

Appendix F
Customer Communication F-1

Glossary G-1

Index I-1

Figures

Figure 1-1. GPIB-ENET Top Panel and LEDs 1-2

Figure 2-1. GPIB-ENET Bottom Panel Identification Label 2-1
Figure 2-2. GPIB-ENET Back Panel Switches 2-2
Figure 2-3. Default Switch Setting for Ethernet Port Configuration 2-3
Figure 2-4. Switch Setting for AUI Ethernet Port Configuration 2-3

Figure C-1. GPIB-ENET Back Panel Switches C-1

Tables

Table 1-1. LED Descriptions 1-3

Table 3-1. Interface Status Word Bits 3-2
Table 3-2. iberr Descriptions 3-6
Table 3-3. List of Device-Level Functions 3-11
Table 3-4. List of Board-Level Functions 3-11

Table 4-1. ibconfig Board Configuration Parameter Options 4-6
Table 4-2. ibconfig Device Configuration Parameter Options 4-10
Table 4-3. Wait Mask Layout 4-25

Table 5-1. ibic Commands 5-4

Table B-1. Electrical Characteristics B-1
Table B-2. Environmental Characteristics B-1
Table B-3. Physical Characteristics B-1

Table C-1. DIP Switch Settings for Modes of Operation C-2

Table E-1. Sample READY LED Signals and the Corresponding Status Code Numbers E-2

About This Manual

This manual describes the ESP-488 software in C for the GPIB-ENET and advanced GPIB-ENET programming information.

This manual assumes that you are familiar with the IEEE 488 (GPIB) standard and the IEEE 488.2 messaging protocols. You should use the IEEE 488 (GPIB) standard and the IEEE 488.2 messaging protocols when you develop your software.

Organization of This Manual

This manual is organized as follows:

- Chapter 1, *Introduction*, lists your kit contents and optional equipment, and briefly describes the GPIB-ENET hardware and ESP-488 software.
- Chapter 2, *Configuring the Hardware*, explains how to configure your GPIB-ENET hardware.
- Chapter 3, *The ESP-488 C Language Library*, contains a general description of the C language programming interface used in the ESP-488 software.
- Chapter 4, *ESP-488 Functions*, describes the purpose, format, input and output parameters, and possible errors for each function available with the ESP-488 software.
- Chapter 5, *Using Your ESP-488 Software*, describes ESP-488 coding conventions, explains other programming considerations, and discusses the sample code included on your ESP-488 distribution diskette.
- Chapter 6, *Verification and Troubleshooting*, describes how to verify the hardware and installation and troubleshoot problems.
- Appendix A, *Multiline Interface Messages*, contains a multiline interface message reference list, which describes the mnemonics and messages that correspond to the interface functions.
- Appendix B, *Hardware Specifications*, lists the electrical, environmental, and physical characteristics of the GPIB-ENET and the recommended operating conditions.
- Appendix C, *GPIB-ENET 8-Bit DIP Switch*, describes how the DIP switch on the back panel affects the operation of the GPIB-ENET.

- Appendix D, *GPIB-ENET Configuration Utilities,* contains information on the GPIB-ENET `IPassign, Update,` and `IPsetup` utilities.
- Appendix E, *READY LED Signaling,* describes how to interpret the **READY** LED error codes.
- Appendix F, *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 a description of the terms, such as abbreviations, acronyms, metric prefixes, mnemonics, and symbols, that this manual uses.
- The *Index* contains an alphabetical list of the key terms and topics that this manual uses, and it includes the page number where you can locate each term and topic.

Conventions Used in This Manual

This manual uses the following conventions.

italic	Italic text denotes emphasis, a cross reference, or an introduction to a key concept.
bold italic	Bold italic text denotes a note, caution, or warning.
`monospace`	Lowercase text in this font denotes text or characters that you literally input from the keyboard. It also denotes sections of code, programming examples, syntax examples, the proper names of disk drives, paths, directories, programs, subprograms, subroutines, device names, functions, variables, filenames, extensions, parameters, and statements and comments taken from program code.
`bold monospace`	Bold lowercase text in this font denotes the messages and responses that the computer automatically prints to the screen.
`italic monospace`	Italic lowercase text in this font denotes that you must supply the appropriate words or values in the place of these items.
IEEE 488 and IEEE 488.2	IEEE 488 and IEEE 488.2 refer to the ANSI/IEEE Standard 488.1-1987 and the ANSI/IEEE Standard 488.2-1987, respectively, which define the GPIB.

The *Glossary* lists abbreviations, acronyms, metric prefixes, mnemonics, symbols, and terms.

Related Documentation

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

- ANSI/IEEE Standard 488.1-1987, *IEEE Standard Digital Interface for Programmable Instrumentation*
- ANSI/IEEE Standard 488.2-1987, *IEEE Standard Codes, Formats, Protocols, and Common Commands*
- ANSI/IEEE Standard 802.3-1988, *Information Processing Systems, Local Area Networks, Part 3*

In addition, you might need to refer to the documentation that came with your system for network programming information.

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 F, *Customer Communication*, at the end of this manual.

Chapter 1
Introduction

This chapter lists your kit contents and optional equipment, and briefly describes the GPIB-ENET hardware and ESP-488 (Engineering Software Package) software.

What Your Kit Should Contain

Your kit should contain the following components:

- DOS-formatted, high-density diskette containing ESP-488 in C for the GPIB-ENET
- *ESP-488 Software Reference Manual for the GPIB-ENET*

Optional Equipment

You can call National Instruments to order the following optional equipment to use with your ESP-488 in C for the GPIB-ENET kit.

GPIB-ENET Hardware Kits

The following GPIB-ENET hardware kits include a GPIB-ENET with the specified connectors, voltage requirement, and power cord for use in the stated country.

GPIB-ENET Twisted Pair/AUI, United States 120 VAC
GPIB-ENET Twisted Pair/AUI, Swiss 220 VAC
GPIB-ENET Twisted Pair/AUI, Australian 240 VAC
GPIB-ENET Twisted Pair/AUI, Universal European 240 VAC
GPIB-ENET Twisted Pair/AUI, North American 240 VAC
GPIB-ENET Twisted Pair/AUI, UK 240 VAC

GPIB-ENET Coaxial/AUI, United States 120 VAC
GPIB-ENET Coaxial/AUI, Swiss 220 VAC
GPIB-ENET Coaxial/AUI, Australian 240 VAC
GPIB-ENET Coaxial/AUI, Universal European 240 VAC
GPIB-ENET Coaxial/AUI, North American 240 VAC
GPIB-ENET Coaxial/AUI, UK 240 VAC

Ethernet Cables

Twisted pair (10Base-T) cables (1 m, 5 m, or 10 m)
Coax (10Base-2) cables (1 m, 5 m, or 10 m)
AUI (10Base-5) cables (1 m, 5 m, or 10 m)

Shielded GPIB Cables*

Type X2 double-shielded cables (1 m, 2 m, or 4 m)

* To meet FCC emission limits for this device, you must use a shielded (Type X2) GPIB cable. Operating this equipment with a non-shielded cable may cause interference to radio and television reception.

Hardware Description

The GPIB-ENET transparently handles data transfers between an Ethernet-based TCP/IP host and the GPIB. With the GPIB-ENET, multiple computers can share a set of GPIB instruments or a single computer can control several GPIB systems.

The GPIB-ENET converts a computer equipped with an Ethernet port into a GPIB Talker/Listener/Controller. It is powered by an internal 100-120 VAC or 220-240 VAC supply.

The GPIB-ENET has all the firmware and logic required to implement the physical and electrical characteristics of all versions of the ANSI/IEEE 488 standard, including ANSI/IEEE Standard 488.2-1987, and ANSI/IEEE Standard 802.3. The GPIB-ENET interprets and executes commands that you send to it over an Ethernet link and performs all necessary Ethernet-to-GPIB protocol conversions. For detailed hardware specifications, refer to Appendix B, *Hardware Specifications*.

Figure 1-1 shows the seven light-emitting diodes (LEDs) on the GPIB-ENET top panel. The LEDs show the current status of the GPIB-ENET. Table 1-1 describes each LED.

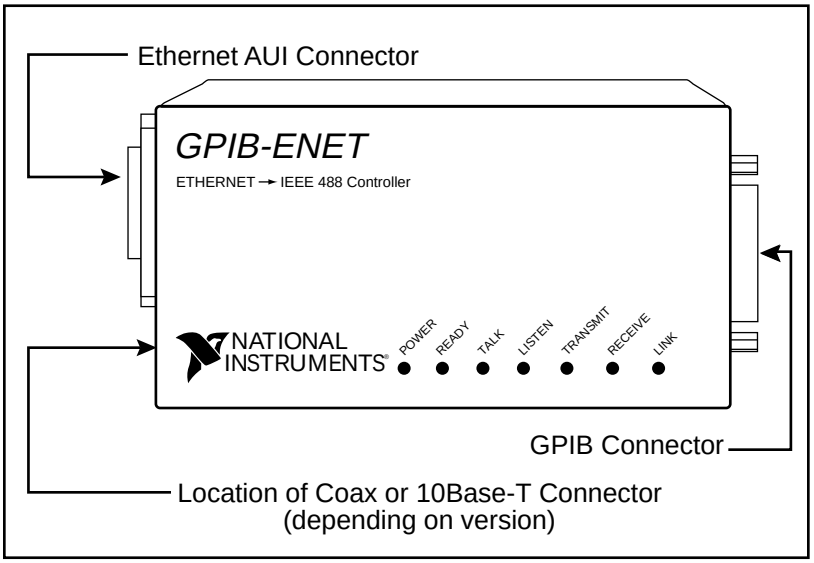


Figure 1-1. GPIB-ENET Top Panel and LEDs

Table 1-1. LED Descriptions

LED	Indication
POWER	Indicates that power has been supplied to the unit and the ON/OFF switch is in the ON position.
READY	Indicates the functional state of the GPIB-ENET. When the GPIB-ENET is powered on, this LED flashes and then becomes steady when the box is ready for operation. Continuous flashing indicates an error has occurred.
TALK	Indicates that GPIB-ENET is configured as a GPIB Talker.
LISTEN	Indicates that GPIB-ENET is configured as a GPIB Listener.
TRANSMIT	Indicates that the GPIB-ENET is transmitting onto the Ethernet network.
RECEIVE	Indicates that the GPIB-ENET is receiving Ethernet network traffic.
LINK	Indicates that the GPIB-ENET has detected a twisted pair (10Base-T) link. For GPIB-ENET coax and AUI options, this LED is not used and remains OFF.

ESP-488 Software Description

The ESP-488 development software includes source code that performs basic IEEE 488 functions. The code contains functions for synchronous communication, device configuration, and status reporting. You can customize the code for your specific applications. Refer to Chapter 5, *Using Your ESP-488 Software*, for more information about modifying the code.

You should use the ESP-488 software in systems that do not have a corresponding NI-488.2 software package. The NI-488.2 driver software packages provide complete NI-488.2 compatibility and development utilities. NI-488.2 software for the GPIB-ENET is available for the Sun Solaris, HP-UX, OSF/1, Mac OS, and Microsoft Windows platforms.

File Descriptions

Check your ESP-488 distribution diskette for the following files:

`\esp.c`	ESP-488 Source Code
`\esp.h`	Header File for the ESP-488 Code
`\eppproto.h`	Prototypes for the ESP-488 Code
`\gpibcnst.h`	ESP-488 Constants
`\ibic.c`	Sample Application Using the ESP-488 Software
`\read.me`	Extra Information Not Found in the Documentation
`\enet_xx.bin`	Latest GPIB-ENET Firmware Revision at Release
`\solaris1\IPassign`	IPassign Executable for SPARC Platforms using Solaris 1
`\solaris1\IPsetup`	IPsetup Executable for SPARC Platforms using Solaris 1
`\solaris1\Update`	Update Executable for SPARC Platforms using Solaris 1
`\solaris2\IPassign`	IPassign Executable for SPARC Platforms using Solaris 2
`\solaris2\IPsetup`	IPsetup Executable for SPARC Platforms using Solaris 2
`\solaris2\Update`	Update Executable for SPARC Platforms using Solaris 2
`\HP-UX\IPassign`	IPassign Executable for HP-UX Series 700 Platforms
`\HP-UX\IPsetup`	IPsetup Executable for HP-UX Series 700 Platforms
`\HP-UX\Update`	Update Executable for HP-UX Series 700 Platforms

Chapter 2
Configuring the Hardware

This chapter explains how to configure your GPIB-ENET hardware.

This chapter contains references to two utilities, `IPassign` and `IPsetup`, that are not available in source code. The ESP-488 distribution diskette includes executable versions of these utilities for the Solaris 1 and Solaris 2 SPARC platforms, and the HP-UX Series 700 platforms. Executables for other common platforms, such as Microsoft Windows and Mac OS, are also available from National Instruments. If you cannot use any of the included executables, contact National Instruments for assistance.

Step 1. Determine IP and Ethernet Addresses

The Internet Protocol (IP) address is the address that TCP/IP-based networks use to route information to the appropriate network and host. When you first install a GPIB-ENET in a network, or if the IP address for the GPIB-ENET changes, the GPIB-ENET IP address must be configured.

1. Note the Ethernet address from the bottom panel label of the GPIB-ENET.

   The Ethernet address is not the same thing as the IP address. All devices on an Ethernet network are assigned a physical address, the Ethernet address, so that they can communicate with each other. The identification label on the bottom panel of the GPIB-ENET contains various information, including the Ethernet address. See Figure 2-1 for a picture of the GPIB-ENET identification label.

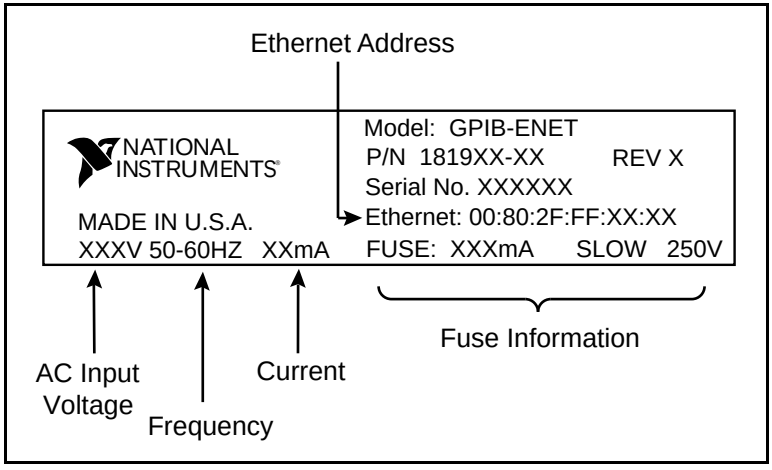


Figure 2-1. GPIB-ENET Bottom Panel Identification Label

2. Contact your network administrator to use the GPIB-ENET on your network.

   Your network administrator assigns a unique, valid IP address to your GPIB-ENET. You should also obtain a fully qualified name for the GPIB-ENET. Your network administrator can also tell you whether the GPIB-ENET should have its IP address configured automatically or manually. A Reverse Address Resolution Protocol (RARP) server on the network automatically configures the GPIB-ENET with an IP address when you power on the GPIB-ENET; once your network administrator has configured your network RARP server, you can use automatic configuration. You can configure the IP address manually using the `IPassign` utility. Once you configure the IP address, it is stored in the GPIB-ENET so that you do not need to reassign it every time you power on the box.

3. Make a note of the IP address and name mapping assigned to the GPIB-ENET hardware. You use the IP address when configuring the hardware and the name mapping when you using the ESP-488 software.

Step 2. Configure the GPIB-ENET Operating Characteristics

The GPIB-ENET has a two-position slide switch that you use to configure the network interface connector of the box. The GPIB-ENET also has an 8-bit DIP switch used to configure the operating characteristics. The switches are located on the back panel of the GPIB-ENET, as shown in Figure 2-2.

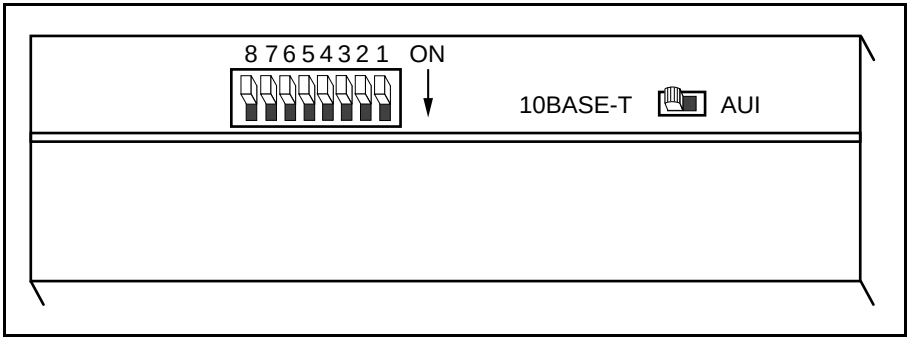


Figure 2-2. GPIB-ENET Back Panel Switches

Configure the Slide Switch

Your GPIB-ENET has two network interface connectors: a 15-pin AUI connector and either a 10Base-T Ethernet connector or a coax Ethernet connector. Use the slide switch to select which of the two connectors the GPIB-ENET uses. The default switch setting is either 10Base-T or coax. Figure 2-3 depicts the default setting for the different types of GPIB-ENET boxes.

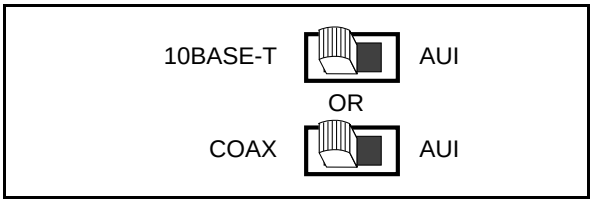


Figure 2-3. Default Switch Setting for Ethernet Port Configuration

If you want to use the AUI connector, change the switch setting to match the appropriate setting in Figure 2-4.

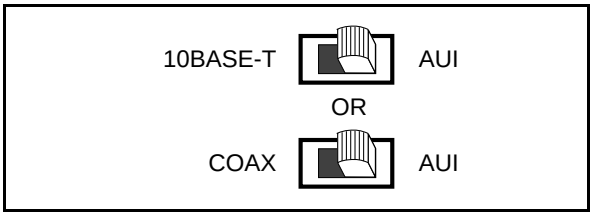


Figure 2-4. Switch Setting for AUI Ethernet Port Configuration

Configure the 8-Bit DIP Switch

The 8-bit DIP switch selects the operating mode of the GPIB-ENET. The GPIB-ENET is shipped to you with the DIP switches set for normal operation mode. In normal operation mode, all the switches are in the OFF position. Most users use the GPIB-ENET in normal operation mode. Do not change the DIP switch settings unless you run a GPIB-ENET utility that requires you to change them. For information about using the 8-bit DIP switch, refer to Appendix C, *GPIB-ENET 8-Bit DIP Switch*.

Step 3. Connect the Cables

If you do not have both a GPIB cable and Ethernet cable, refer to the *Optional Equipment* section in Chapter 1 for ordering information.

1. Connect one end of the Ethernet cable to the GPIB-ENET and securely fasten it. Connect the other end of the Ethernet cable to your Ethernet network. Be sure to follow all IEEE 802.3 cabling restrictions.

2. Connect one end of the GPIB cable to the GPIB-ENET and tighten the thumb screws on the connector. Connect the other end of the GPIB cable to your GPIB system. Be sure to follow all IEEE 488.1 cabling restrictions and use only shielded GPIB cables.

3. Plug the power cord into an AC outlet of the correct voltage.

Step 4. Switch on Your GPIB-ENET

Warning: ***Operating the GPIB-ENET at any voltage other than the one specified could damage the unit.***

The GPIB-ENET is shipped from the factory with either a 100-120 VAC or 220-240 VAC power supply. Before you configure your GPIB-ENET, verify that the voltage rating listed on the bottom of the box matches the voltage that is supplied in your area.

To assign the IP address manually, refer to Appendix D, *GPIB-ENET Configuration Utilities*, and run the `IPassign` utility.

For automatic IP address assignment using a RARP server, power on the GPIB-ENET. The **POWER** LED comes on immediately. The **READY** LED flashes while the GPIB-ENET completes its power-on self tests and the IP address is assigned.

The power-on self tests take about 10 seconds to complete, then the box obtains its IP address. The time required for the IP address assignment is highly dependent on your network and the configuration of your GPIB-ENET. If the **READY** LED does not become steady after 1 minute, refer to Chapter 6, the section *Troubleshooting Hardware Problems*.

Once the **READY** LED remains steady, the unit is ready to operate and you can configure the software.

Step 5. Configure Subnet Information

Before the GPIB-ENET is fully functional, you should configure it with proper subnet information. Your GPIB-ENET may work properly even if you skip this step, but National Instruments recommends that you configure information about the subnet. This information includes the broadcast IP address and netmask to use on the subnet and up to four router IP addresses. Refer to Appendix D, *GPIB-ENET Configuration Utilities*, and run the `IPsetup` utility.

Chapter 3
The ESP-488 C Language Library

This chapter contains a general description of the C language programming interface used in the ESP-488 software.

Global Variables

Upon completion of a command, the following global variables describe the GPIB status, the error conditions, and, in the case of I/O operations, the data transfer count.

`ibsta`	Describes the current GPIB status.
`iberr`	Gives the error code but is not meaningful unless the ERR bit is set in `ibsta`.
`ibcntl`	Gives the number of bytes read or written over the GPIB. This number can be an unsigned integer value in the range of 0 to 4 GB.

Status Variable: ibsta

ESP-488 functions automatically update the `ibsta` status word, which contains 16 status bits. Some bits in `ibsta` can be set for both device-level calls (dev) and board-level calls (brd). Other bits are valid for board-level calls only.

Table 3-1 lists the `ibsta` bits.

Table 3-1. Interface Status Word Bits

Mnemonic	Bit Pos.	Hex Value	Type	Description
ERR	15	8000	dev, brd	GPIB error
TIMO	14	4000	dev, brd	Time limit exceeded
END	13	2000	dev, brd	END or EOS detected
SRQI	12	1000	brd	SRQ is asserted
CMPL	8	100	dev, brd	I/O completed
LOK	7	80	brd	Lockout State
REM	6	40	brd	Remote State
CIC	5	20	brd	Controller-In-Charge
ATN	4	10	brd	Attention is asserted
TACS	3	8	brd	Talker
LACS	2	4	brd	Listener
DTAS	1	2	brd	Device Trigger State
DCAS	0	1	brd	Device Clear State

ERR (dev, brd)

ERR is set in the status word following any call that results in an error. You can determine the particular error by examining the error variable `iberr`. Later in this chapter, a section describes error codes that are recorded in `iberr` along with possible solutions. ERR is cleared following any call that does not result in an error.

TIMO (dev, brd)

TIMO indicates that the timeout period has been exceeded. TIMO is set in the status word following an `ibwait` call if the TIMO bit of the `ibwait` mask parameter is set and the time limit expires. TIMO is also set following any synchronous I/O functions (for example, `ibcmd`, `ibrd`, and `ibwrt`) if a timeout occurs during one of these calls. TIMO is cleared in all other circumstances.

END (dev, brd)

END indicates that either the GPIB EOI line has been asserted or that the EOS byte has been received, if the software is configured to terminate a read on an EOS byte. END is cleared when any I/O operation is initiated.

Some applications might need to know the exact I/O read termination mode of a read operation–EOI by itself, the EOS character by itself, or EOI plus the EOS character. You can use the `ibconfig` function (option `IbcEndBitIsNormal`) to enable a mode in which the END bit is set only when EOI is asserted. In this mode if the I/O operation completes because of the EOS character by itself, END is not set. The application should check the last byte of the received buffer to see if it is the EOS character.

SRQI (brd)

SRQI indicates that a GPIB device is requesting service. SRQI is set whenever the GPIB-ENET is CIC and the GPIB SRQ line is asserted. SRQI is cleared either when the GPIB board ceases to be the CIC or when the GPIB SRQ line is unasserted.

CMPL (dev, brd)

CMPL indicates the condition of I/O operations. It is set whenever an I/O operation is complete. CMPL is cleared while an I/O operation is in progress.

LOK (brd)

LOK indicates whether the GPIB-ENET is in a lockout state. LOK is set whenever the GPIB-ENET detects that the Local Lockout (LLO) message has been sent either by the GPIB-ENET or by another Controller. LOK is cleared when the System Controller unasserts the Remote Enable (REN) GPIB line.

REM (brd)

REM indicates whether the GPIB-ENET is in the remote state. REM is set whenever the Remote Enable (REN) GPIB line is asserted and the GPIB-ENET detects that its listen address has been sent either by the GPIB-ENET or by another Controller. REM is cleared in the following situations:

- When REN becomes unasserted
- When the GPIB-ENET as a Listener detects that the Go to Local (GTL) command has been sent either by the GPIB-ENET or by another Controller

CIC (brd)

CIC indicates whether the GPIB-ENET is the Controller-In-Charge. CIC is set when the `ibsic` function is executed while the GPIB-ENET is System Controller or when another Controller passes control to the GPIB-ENET. CIC is cleared whenever the GPIB-ENET detects Interface Clear (IFC) from the System Controller, or when the GPIB-ENET passes control to another device.

ATN (brd)

ATN indicates the state of the GPIB Attention (ATN) line. ATN is set whenever the GPIB ATN line is asserted, and it is cleared when the ATN line is unasserted.

TACS (brd)

TACS indicates whether the GPIB-ENET is addressed as a Talker. TACS is set whenever the GPIB-ENET detects that its talk address (and secondary address, if enabled) has been sent either by the GPIB-ENET itself or by another Controller. TACS is cleared whenever the GPIB-ENET detects the Untalk (UNT) command, its own listen address, a talk address other than its own talk address, or Interface Clear (IFC).

LACS (brd)

LACS indicates whether the GPIB-ENET is addressed as a Listener. LACS is set whenever the GPIB-ENET detects that its listen address (and secondary address, if enabled) has been sent either by the GPIB-ENET itself or by another Controller. LACS is cleared whenever the GPIB-ENET detects the Unlisten (UNL) command, its own talk address, or Interface Clear (IFC).

DTAS (brd)

DTAS indicates whether the GPIB-ENET has detected a device trigger command. DTAS is set whenever the GPIB-ENET, as a Listener, detects that the Group Execute Trigger (GET) command has been sent by another Controller. DTAS is cleared on any call immediately following an `ibwait` call, if the DTAS bit is set in the `ibwait` mask parameter. See the *Note* below.

DCAS (brd)

DCAS indicates whether the GPIB-ENET has detected a device clear command. DCAS is set whenever the GPIB-ENET detects that the Device Clear (DCL) command has been sent by another Controller, or whenever the GPIB-ENET as a Listener detects that the Selected Device Clear (SDC) command has been sent by another Controller. DCAS is cleared on any call immediately following an `ibwait` call, if the DCAS bit was set in the `ibwait` mask parameter. It also clears on any call immediately following a read or write. See the *Note* below.

Note: ***The ESP-488 package does not contain complete functions for handling DCAS and DTAS. (The actions that you should take are described by the IEEE 488.2 standard and are device specific.) Therefore, you must clear these bits programmatically when you develop routines to handle these events.***

Error Variable: iberr

When the ERR bit in the `ibsta` status word is set, the error code in `iberr` is valid. Table 3-2 lists possible error indications.

Table 3-2. iberr Descriptions

Error Mnemonic	iberr Value	Meaning
EDVR	0	Operating system error
ECIC	1	Function requires GPIB board to be CIC
ENOL	2	No Listeners on the GPIB
EADR	3	GPIB board not addressed correctly
EARG	4	Invalid argument to function call
ESAC	5	GPIB board not System Controller as required
EABO	6	I/O operation aborted
ECAP	11	No capability for operation
EBUS	14	GPIB bus error
ELCK	21	Lock error
ECFG	24	Board is configured differently than requested

EDVR (0)

EDVR is an operating system error. When this error is reported, additional information is recorded in the `ibcntl` global variable. If `ibcntl` is 0xe0 or greater, the error is returned from the ESP-488 software. Otherwise, the error is the value in the global error variable `errno`.

ECIC (1)

ECIC is returned when one of the following functions is performed while the board is not CIC:

- Any device-level function that affects the GPIB
- `ibcmd`
- `ibrpp`

Solutions

- Use `ibsic` to make the GPIB-ENET become Controller-In-Charge on the GPIB.
- Make sure your GPIB-ENET is configured as System Controller.
- In multiple CIC situations, always be certain that the CIC bit appears in the status word `ibsta` before attempting these calls. If it does not appear, you can perform an `ibwait` (for CIC) call to delay further processing until control is passed to the board.

ENOL (2)

ENOL usually occurs when a write operation is attempted with no Listeners addressed. ENOL can also occur in situations in which the GPIB-ENET is not the CIC and the Controller asserts ATN before the write call in progress has ended. When the GPIB-ENET is CIC, this error can also occur when an `ibcmd` is performed with no other devices attached to the GPIB.

Solutions

- Make sure that the GPIB address of your device configuration matches the GPIB address of the device to which you want to write data.
- If you are not using device-level calls, make sure that your device is properly addressed to listen before writing to it by using `ibcmd`.

- Use the appropriate hex code in `ibcmd` to address your device.
- Check your cable connections and make sure at least two-thirds of your devices are powered on.
- Reduce the write byte count to that which is expected by the Controller.

EADR (3)

EADR occurs when the GPIB-ENET is CIC and is not properly addressing itself before read and write functions. This error is associated with board-level functions.

Solutions

Make sure that the GPIB-ENET is addressed correctly before calling board-level `ibrd` or `ibwrt`.

EARG (4)

EARG results when an invalid argument is passed to a function call. The following are some examples:

- A board-level call made with a valid device descriptor, or a device-level call made with a valid board descriptor
- An invalid `Value` is given for the `Option` requested with `ibconfig`

Solutions

- Make sure that the parameters passed to the functions are valid.
- Do not use a device descriptor in a board function or vice-versa.

ESAC (5)

ESAC results when a function requiring System Controller functionality is called when the GPIB-ENET does not have System Controller capability.

Solutions

Give the GPIB board System Controller capability by calling `ibconfig` with the `IbcSC` option.

EABO (6)

EABO indicates that an I/O operation has been canceled, usually due to a timeout condition. This error also occurs upon receipt of the Device Clear message from the CIC while performing an I/O operation.

Frequently, the I/O is not progressing (the Listener is not continuing to handshake or the Talker has stopped talking), or the byte count in the call which timed out was more than the other device was expecting.

Solutions

- Use the correct byte count in input functions or have the Talker use the END message to signify the end of the transfer.
- Lengthen the timeout period for the I/O operation using `ibconfig` with the `IbcTMO` option.
- Make sure that you have configured your device to send data before you request data.

ECAP (11)

ECAP results when the GPIB-ENET lacks the ability to carry out an operation or when a particular capability has been disabled in the software and a call is made that requires the capability.

Solutions

Check the validity of the call, or make sure the GPIB-ENET has the necessary capability.

EBUS (14)

EBUS results when certain GPIB bus errors occur during device-level functions. All device functions send command bytes to perform addressing and other bus management. Devices are expected to accept these command bytes within the time limit specified by the default configuration or the `ibconfig` function using the `IbcTMO` option. EBUS results if a timeout occurred while sending these command bytes.

Solutions

- Verify that the instrument is operating correctly.
- Check for loose or faulty cabling or several powered off instruments on the GPIB.
- If the timeout period is too short to send command bytes, increase the timeout period.

ELCK (21)

ELCK occurs when a request cannot be completed because the GPIB-ENET is locked or being used by another user. Locking of descriptors is not supported by the ESP-488 software. However, this error is possible when another user is using your GPIB-ENET with a National Instruments driver that supports locking.

Solutions

Make sure that no one else is using your GPIB-ENET.

ECFG (24)

ECFG occurs only when opening a descriptor. It indicates that the GPIB-ENET is already online and configured (PAD, SAD, TIMEOUT, and so on) by another user. The descriptors returned are valid. ECFG is just a warning that the GPIB-ENET is configured differently than requested. After you have a valid descriptor, you can change the configuration using `ibconfig` (as long as the descriptor is not locked).

Note: ***You should alert other active GPIB-ENET users before changing the configuration of a descriptor, because the changes may have unpredictable effects on other applications.***

Solutions

Make sure that no one else is using your GPIB-ENET.

Count Variable: ibcntl

The `ibcntl` variable is updated with the actual byte count that is transferred over the GPIB after every I/O function has completed.

ESP-488 Function Descriptions

The remainder of this chapter lists the functions in the ESP-488 software package. Table 3-3 lists the device-level functions. Table 3-4 lists the board-level functions.

Table 3-3. List of Device-Level Functions

Function	Purpose
`ibconfig`	Change the software configuration parameters
`ibfind`	Open and initialize a device descriptor
`ibonl`	Place the device online or offline
`ibrd`	Read data from a device into a user buffer
`ibrpp`	Conduct a parallel poll
`ibrsp`	Conduct a serial poll
`ibwait`	Wait for GPIB events
`ibwrt`	Write data to a device from a user buffer

Table 3-4. List of Board-Level Functions

Function	Purpose
`ibcmd`	Send GPIB commands
`ibconfig`	Change the software configuration parameters
`ibfind`	Open and initialize a board descriptor
`ibonl`	Place the interface board online or offline
`ibrd`	Read data from a device into a user buffer
`ibrpp`	Conduct a parallel poll
`ibsic`	Assert interface clear
`ibwait`	Wait for GPIB events
`ibwrt`	Write data to a device from a user buffer

Chapter 4
ESP-488 Functions

This chapter describes the purpose, format, input and output parameters, and possible errors for each function available with the ESP-488 software.

For general programming information, refer to Chapter 5, *Using Your ESP-488 Software*. In addition, the ESP-488 distribution diskette contains sample source code.

Function Names

The functions in this chapter are listed alphabetically. Each function is designated as board level, device level, or both.

Purpose

Each function description includes a brief statement of the purpose of the function.

Input and Output

The input and output parameters for each function are listed. Function Return describes the return value of the function. The return value of the ESP-488 functions is usually the value of `ibsta`.

Description

The description section gives details about the purpose and effect of each function.

Possible Errors

Each function description includes a list of errors that could occur when the function is invoked.

Examples

Each function description includes sample code showing how to use the function. Unlike an actual program, these examples do not check for or handle errors. For a more detailed and complete example, refer to the sample application program supplied on the distribution diskette.

IBCMD Board Level IBCMD

Purpose

Send GPIB commands.

Format

```
int16 ibcmd (GPIBDESCRIPTOR Descriptor, int8 *Cmdbuf,
             uint32 Cnt)
```

Input

`Descriptor`	Board descriptor returned from `ibfind`
`Cmdbuf`	Buffer of command bytes to send
`Cnt`	Number of command bytes to send

Output

Function Return	The value of `ibsta`

Description

`ibcmd` sends `Cnt` bytes from `Cmdbuf` over the GPIB as command bytes (interface messages). The number of command bytes transferred is returned in the global variable `ibcntl`. Refer to Appendix A, *Multiline Interface Messages*, for a table of the defined interface messages.

Command bytes are used to configure the state of the GPIB. They are not used to send instructions to GPIB devices. Use `ibwrt` to send device-specific instructions.

Possible Errors

EABO	The timeout period expired before all of the command bytes were sent.
EARG	`Descriptor` does not refer to a board.
ECIC	The GPIB-ENET is not Controller-In-Charge.
EDVR	A system or other unrecoverable error has occurred.
ENOL	No devices are on the GPIB.

IBCMD **Board Level** **IBCMD (Continued)**

Example

Using a board-level descriptor previously opened with `ibfind()`, issue GPIB command bytes.

```
#include "esp.h"

GPIBDESCRIPTOR Board;

ibcmd(Board, "?_@!", 4); /*Issue GPIB command messages UNL and UNT
                           followed by Talk address 0 and Listen
                           address 1*/
```

IBCONFIG

Board Level
Device Level

IBCONFIG

Purpose

Change the software configuration parameters.

Format

```
int16 ibconfig (GPIBDESCRIPTOR Descriptor, int8 Option,
                int8 Value
```

Input

`Descriptor`	Board or device unit descriptor returned from `ibfind`
`Option`	A parameter that selects the configuration item
`Value`	The value to which the selected configuration item is to be changed

Output

Function Return	The value of `ibsta`

Description

`ibconfig` changes the configuration item to the specified value for the selected board or device. `Option` may be any of the defined constants in Table 4-1 and `Value` must be valid for the parameter that you are configuring. The previous setting of the configured item is return in `iberr`.

Possible Errors

EARG	Either `Option` or `Value` is not valid. See Tables 4-1 and 4-2.
ECAP	The driver is not able to make the requested change.
EDVR	A system or other unrecoverable error has occurred.

IBCONFIG

Board Level
Device Level

IBCONFIG
(Continued)

Table 4-1 lists the options you can use with `ibconfig` when `Descriptor` is a board descriptor. If the table does not list the default value for a particular option, the default value is determined by the ESP-488 software.

The following is an alphabetical list of the `Option` constants included in Table 4-1.

Constants	Definitions	Constants	Definitions
• IbcEndBitIsNormal	0x001A	• IbcPPollTime	0x0019
• IbcEOSchar	0x000F	• IbcReadAdjust	0x0013
• IbcEOScmp	0x000E	• IbcRsv	0x0021
• IbcEOSrd	0x000C	• IbcSAD	0x0002
• IbcEOSwrt	0x000D	• IbcSC	0x000A
• IbcEOT	0x0004	• IbcSRE	0x000B
• IbcIst	0x0020	• IbcTIMING	0x0011
• IbcPAD	0x0001	• IbcTMO	0x0003
• IbcPP2	0x0010	• IbcWriteAdjust	0x0014
• IbcPPC	0x0005		

IBCONFIG

Board Level
Device Level

IBCONFIG
(Continued)

Table 4-1. ibconfig Board Configuration Parameter Options

Options (Constants)	Options (Definitions)	Description
IbcPAD	0x0001	Changes the primary address of the GPIB-ENET. Valid values are 0-30.
IbcSAD	0x0002	Changes the secondary address of the GPIB-ENET. Valid values are 0 to disable secondary addressing; 96-126 to change the secondary address.
IbcTMO	0x0003	Changes the I/O timeout limit of the GPIB-ENET. TNONE 0 no timeout T10us 1 10 µs T30us 2 30 µs T100us 3 100 µs T300us 4 300 µs T1ms 5 1 ms T3ms 6 3 ms T10ms 7 10 ms T30ms 8 30 ms T100ms 9 100 ms T300ms 10 300 ms T1s 11 1 s T3s 12 3 s T10s 13 10 s T30s 14 30 s T100s 15 100 s T300s 16 300 s T1000s 17 1000 s
IbcEOT	0x0004	zero = Disable EOI termination. non-zero = Enable EOI termination.
IbcPPC	0x0005	Configures the GPIB-ENET for parallel polls. Valid values are 0 to unconfigure the parallel poll configuration; 96-111 to configure parallel polls. Default: zero.
IbcSC	0x000A	zero = Release system control. non-zero = Request system control.

(continues)

Table 4-1. ibconfig Board Configuration Parameter Options (Continued)

Options (Constants)	Options (Definitions)	Description
IbcSRE	0x000B	zero = Release the REN line. non-zero = Assert the REN line. Default: zero.
IbcEOSrd	0x000C	zero = Ignore EOS character during read operations. non-zero = Terminate reads when the EOS character is read.
IbcEOSwrt	0x000D	zero = Do not assert EOI with the EOS character during write operations. non-zero = Assert EOI with the EOS character during write operations.
IbcEOScmp	0x000E	zero = Use seven bits for the EOS character comparison. non-zero = Use eight bits for the EOS character comparison.
IbcEOSchar	0x000F	Any 8-bit value. This byte becomes the new EOS character.
IbcPP2	0x0010	zero = PP1 mode-remote parallel poll configuration. non-zero = PP2 mode-local parallel poll configuration. Default: zero.
IbcTIMING	0x0011	1 = Normal timing (T1 delay of 2 μs). 2 = High-speed timing (T1 delay of 500 ns). 3 = Very high-speed timing (T1 delay of 350 ns). The T1 delay is the GPIB source handshake timing.
IbcReadAdjust	0x0013	0 = No byte swapping. 1 = Swap pairs of bytes during a read. Default: zero.

(continues)

Table 4-1. ibconfig Board Configuration Parameter Options (Continued)

Options (Constants)	Options (Definitions)	Description
IbcWriteAdjust	0x0014	0 = No byte swapping. 1 = Swap pairs of bytes during a write. Default: zero.
IbcPPollTime	0x0019	0 = Use the standard duration (2 µs) when conducting a parallel poll. 1 to 17 = Use a variable length duration when conducting a parallel poll. The duration represented by 1 to 17 corresponds to the `IbcTMO` values. Default: zero.
IbcEndBitIsNormal	0x001A	zero = Do not set the END bit of `ibsta` when an EOS match occurs during a read. non-zero = Set the END bit of `ibsta` when an EOS match occurs during a read. Default: non-zero.
IbcIst	0x0020	zero = Clear the individual status (`ist`) bit of the GPIB-ENET. non-zero = Set the individual status (`ist`) bit of the GPIB-ENET.
IbcRsv	0x0021	Changes the serial poll status byte of the GPIB-ENET. Default: zero.

IBCONFIG

Board Level
Device Level

IBCONFIG
(Continued)

Table 4-2 lists the options you can use with `ibconfig` when `Descriptor` is a device descriptor. If the table does not list the default value for a particular option, the default value is determined by the ESP-488 software.

The following is an alphabetical list of the `Option` constants included in Table 4-2.

Constants	Definitions	Constants	Definitions
• IbcEndBitIsNormal	0x001A	• IbcPAD	0x0001
• IbcEOSchar	0x000F	• IbcReadAdjust	0x0013
• IbcEOScmp	0x000E	• IbcSAD	0x0002
• IbcEOSrd	0x000C	• IbcSPollTime	0x0018
• IbcEOSwrt	0x000D	• IbcTMO	0x0003
• IbcEOT	0x0004	• IbcWriteAdjust	0x0014

IBCONFIG

Board Level
Device Level

IBCONFIG
(Continued)

Table 4-2. ibconfig Device Configuration Parameter Options

Options (Constants)	Options (Definitions)	Description
IbcPAD	0x0001	Changes the primary address of the device. Valid values are 0-30.
IbcSAD	0x0002	Changes the secondary address of the device. Valid values are 0 to disable secondary addressing; 96-126 to change the secondary address.
IbcTMO	0x0003	Changes the device I/O timeout limit. TNONE 0 no timeout T10us 1 10 µs T30us 2 30 µs T100us 3 100 µs T300us 4 300 µs T1ms 5 1 ms T3ms 6 3 ms T10ms 7 10 ms T30ms 8 30 ms T100ms 9 100 ms T300ms 10 300 ms T1s 11 1 s T3s 12 3 s T10s 13 10 s T30s 14 30 s T100s 15 100 s T300s 16 300 s T1000s 17 1000 s
IbcEOT	0x0004	zero = Disable EOI termination. non-zero = Enable EOI termination.
IbcEOSrd	0x000C	zero = Ignore EOS character during read operations. non-zero = Terminate reads when the EOS character is read.

(continues)

Table 4-2. ibconfig Device Configuration Parameter Options (Continued)

Options (Constants)	Options (Definitions)	Description
IbcEOSwrt	0x000D	zero = Do not send EOI with the EOS character during write operations. non-zero = Send EOI with the EOS character during writes.
IbcEOScmp	0x000E	zero = Use seven bits for the EOS character comparison. non-zero = Use eight bits for the EOS character comparison.
IbcEOSchar	0x000F	Any 8-bit value. This byte becomes the new EOS character.
IbcReadAdjust	0x0013	0 = No byte swapping. 1 = Swap pairs of bytes during a read. Default: zero.
IbcWriteAdjust	0x0014	0 = No byte swapping. 1 = Swap pairs of bytes during a write. Default: zero.
IbcSPollTime	0x0018	0 to 17 = Sets the length of time the driver waits for a serial poll response byte when polling the given device. The length of time represented by 0 to 17 corresponds to the `IbcTMO` values. Default: 11.
IbcEndBitIsNormal	0x001A	zero = Do not set the END bit of `ibsta` when an EOS match occurs during a read. non-zero = Set the END bit of `ibsta` when an EOS match occurs during a read. Default: non-zero.

IBCONFIG

Board Level
Device Level

IBCONFIG
(Continued)

Examples

1. Configure the System Controller capability of a board-level descriptor previously opened with `ibfind()`.

```
#include "esp.h"

GPIBDESCRIPTOR Board;

ibconfig(Board, IbcSC, 1);       /*Become System Controller*/
ibconfig(Board, IbcSRE, 1);      /*Assert the REN GPIB line*/
```

2. Configure the primary and secondary address of a device-level descriptor previously opened with `ibfind()`.

```
#include "esp.h"

GPIBDESCRIPTOR Device;

ibconfig(Device, IbcPAD, 0x5);   /*Set Primary GPIB address
                                   to 5*/
ibconfig(Device, IbcSAD, 0x61);  /*Set Secondary GPIB address
                                   to 1*/
```

IBFIND

Board Level
Device Level

IBFIND

Purpose

Open and initialize a GPIB board or a user-configurable device.

Format

```
GPIBDESCRIPTOR ibfind(int8* Hostname, TYPE Role)
```

Input

`Hostname`	The network name of a GPIB-ENET
`Role`	Indicates whether the returned descriptor represents a board or device

Output

Function Return	The board or device descriptor

Description

`ibfind` is used to acquire a descriptor for a board or device; this board or device descriptor can be used in subsequent ESP-488 calls.

`ibfind` performs the equivalent of an `ibonl 1` to initialize the board or device descriptor. The descriptor returned by `ibfind` remains valid until the board or device is put offline using `ibonl 0`.

If `ibfind` is unable to get a valid descriptor, a -1 is returned; the ERR bit is set in `ibsta` and `iberr` contains EDVR.

Possible Errors

ECIC	Device level: The access board is not CIC.
EDVR	A system or other unrecoverable error has occurred.

IBFIND **Board Level** **Device Level** **IBFIND (Continued)**

Examples

1. Open a board-level connection to a GPIB-ENET whose fully qualified hostname is `enet1.natinst.com`.

```
#include "esp.h"

GPIBDESCRIPTOR Board;

Board = ibfind("enet1.natinst.com", BOARD);
```

2. Open a device-level connection to a GPIB-ENET whose local hostname is `enetgold`.

```
#include "esp.h"

GPIBDESCRIPTOR Device;

Device = ibfind("enetgold", DEVICE);
```

IBONL

Board Level
Device Level

IBONL

Purpose

Place the device or board online or offline.

Format

```
int16 ibonl (GPIBDESCRIPTOR Descriptor, int8 V)
```

Input

`Descriptor`	Board or device descriptor returned from `ibfind`
`V`	Indicates whether the board or device is to be put online or taken offline

Output

Function Return	The value of `ibsta`

Description

`ibonl` resets the board or device and places all its software configuration parameters in their pre-configured state. In addition, if `V` is zero, the device or board is taken offline. If `V` is non-zero, the device or board is left operational, or online.

If a device or board is taken offline, the board or device descriptor (`Descriptor`) is no longer valid. You must execute an `ibfind` to access the board or device again.

Possible Errors

EDVR	A system or other unrecoverable error has occurred.

Examples

1. Reset the characteristics of a device-level descriptor previously opened with `ibfind()` to the initial (default) state.

```
#include "esp.h"

GPIBDESCRIPTOR Device;

ibonl(Device, 1);
```

IBONL — Board Level / Device Level — IBONL (Continued)

2. Reset the characteristics of a board-level descriptor previously opened with `ibfind()` to the initial (default) state and take the board offline (similar to disconnecting the GPIB cable).

```
#include "esp.h"

GPIBDESCRIPTOR Board;

ibonl(Board, 0);
```

IBRD

Board Level
Device Level

IBRD

Purpose

Read data from a device into a user buffer.

Format

```
int16 ibrd (GPIBDESCRIPTOR Descriptor, int8* Rdbuf, uint32 Cnt)
```

Input

`Descriptor`	Board or device descriptor returned from `ibfind`
`Cnt`	Number of bytes to be read from the GPIB

Output

`Rdbuf`	Address of buffer into which data is read
Function Return	The value of `ibsta`

Description

Device Level

If `Descriptor` is a device descriptor, `ibrd` addresses the GPIB, reads up to `Cnt` bytes of data, and places the data into the buffer specified by `Rdbuf`. The operation terminates normally when `Cnt` bytes have been received or END is received. The operation terminates with an error if the transfer could not complete within the timeout period. The actual number of bytes transferred is returned in the global variable `ibcntl`.

Board Level

If `Descriptor` is a board descriptor, `ibrd` reads up to `Cnt` bytes of data from a GPIB device and places it into the buffer specified by `Rdbuf`. A board-level `ibrd` assumes that the GPIB is already properly addressed. The operation terminates normally when `Cnt` bytes have been received or END is received. The operation terminates with an error if the transfer could not complete within the timeout period or, if the board is not the CIC, the CIC sends a Device Clear message on the GPIB. The actual number of bytes transferred is returned in the global variable `ibcntl`.

IBRD **Board Level** **IBRD**
Device Level **(Continued)**

Possible Errors

EABO	Either `Cnt` bytes or END was not received within the timeout period or, for board-level reads, a Device Clear message was received after the read operation began.
EADR	Board level: The GPIB is not correctly addressed. Use `ibcmd` to address the GPIB.
EBUS	Device level: No devices are connected to the GPIB.
ECIC	Device level: The access board is not CIC.
EDVR	A system or other unrecoverable error has occurred.

Examples

1. Using a board-level descriptor previously opened with `ibfind()`, read data from a GPIB device. When using a board-level read, all GPIB addressing must be performed before the `ibrd()`.

```
#include "esp.h"

GPIBDESCRIPTOR Board;
int8 Buffer[255];

ibrd(Board, Buffer, 255);    /*Read up to 255 bytes from a Talk-
                               addressed GPIB device*/
```

2. Using a device-level descriptor previously opened with `ibfind()`, read data from a GPIB device. When using a device-level read, all GPIB addressing is performed by the GPIB-ENET.

```
#include "esp.h"

GPIBDESCRIPTOR Device;
int8 Buffer[255];

ibrd(Device, Buffer, 255);   /*Read up to 255 bytes from a GPIB
                               device associated with the
                               device-level descriptor*/
```

IBRPP — Board Level / Device Level — IBRPP

Purpose

Conduct a parallel poll.

Format

```
int16 ibrpp (GPIBDESCRIPTOR Descriptor, int8* Ppr)
```

Input

`Descriptor`	Board or device descriptor returned from `ibfind`

Output

`Ppr`	Parallel poll response byte
Function Return	The value of `ibsta`

Description

`ibrpp` parallel polls all the devices on the GPIB. The result of this poll is returned in `Ppr`.

Possible Errors

ECIC	The GPIB-ENET is not CIC.
EDVR	A system or other unrecoverable error has occurred.

Example

Using a board-level descriptor previously opened with `ibfind()`, perform a parallel poll on the GPIB.

```
#include "esp.h"

GPIBDESCRIPTOR Board;
int8 ParallelPollResponse;

ibrpp(Board, &ParallelPollResponse);
if(ParallelPollResponse != 0)
{  /*At least one configured device is responding to the
     parallel poll*/
   ProcessParallelPoll(ParallelPollResponse);
}
```

IBRSP Device Level IBRSP

Purpose

Conduct a serial poll.

Format

```
int16 ibrsp (GPIBDESCRIPTOR Descriptor, int8* Spr)
```

Input

`Descriptor`	Device descriptor returned from `ibfind`

Output

`Spr`	Serial poll response byte
Function Return	The value of `ibsta`

Description

The `ibrsp` function is used to serial poll the device represented by `Descriptor`. The serial poll response byte is returned in `Spr`. If bit 6 (hex 40) of the response byte is set, the device is requesting service.

Possible Errors

EABO	The serial poll response could not be read within the serial poll timeout period.
EARG	`Descriptor` does not refer to a device.
EBUS	No devices are connected to the GPIB.
ECIC	The access board is not CIC.
EDVR	A system or other unrecoverable error has occurred.

IBRSP **Device Level** **IBRSP (Continued)**

Example

Using a device-level descriptor previously opened with `ibfind()`, perform a serial poll to get the status byte from the device represented by the descriptor.

```
#include "esp.h"

GPIBDESCRIPTOR Device;
int8 SerialPollResponse;

ibrsp(Device, &SerialPollResponse);
if(SerialPollResponse & 0x40)
{  /*The device is requesting service*/
   ProcessDeviceStatus(SerialPollResponse);
}
```

IBSIC **Board Level** IBSIC

Purpose

Assert interface clear.

Format

```
int16 ibsic (GPIBDESCRIPTOR Descriptor)
```

Input

`Descriptor`	Board descriptor returned from `ibfind`

Output

Function Return	The value of `ibsta`

Description

`ibsic` asserts the GPIB interface clear (IFC) line for at least 100 µs if the GPIB-ENET is System Controller. This initializes the GPIB and makes the GPIB-ENET CIC and Active Controller with ATN asserted.

The IFC signal resets only the GPIB interface functions of bus devices and not the internal device functions. Consult your device documentation to determine how to reset the internal functions of your device.

Possible Errors

EARG	`Descriptor` does not refer to a board.
EDVR	A system or other unrecoverable error has occurred.
ESAC	GPIB-ENET does not have System Controller capability.

Example

Using a board-level descriptor previously opened with `ibfind()`, assert the GPIB line IFC to become CIC.

```
#include "esp.h"
GPIBDESCRIPTOR Board;
ibsic(Board);
```

IBWAIT

Board Level
Device Level

IBWAIT

Purpose

Wait for GPIB events.

Format

```
int16 ibwait (GPIBDESCRIPTOR Descriptor, int16 Mask)
```

Input

`Descriptor`	Board or device descriptor returned from `ibfind`
`Mask`	Bit mask of desired GPIB events

Output

Function Return	The value of `ibsta`

Description

`ibwait` monitors the events specified by `Mask` and delays processing until one or more of the events occurs. If the wait mask is zero, `ibwait` returns immediately with an updated `ibsta`. If TIMO is set in the wait mask, `ibwait` returns when the timeout period has elapsed (if one or more of the other specified events have not already occurred). If TIMO is not set in the wait mask, then the function waits indefinitely for one or more of the specified events to occur. The `ibwait` mask bits are identical to the `ibsta` bits and are described in Table 4-3. If `Descriptor` is a device descriptor, the only valid wait mask bits are TIMO, END, and CMPL. If `Descriptor` is a board descriptor, all wait mask bits are valid. You can configure the timeout period using `ibconfig`.

Possible Errors

EARG	`Mask` contains invalid bits.
EDVR	A system or other unrecoverable error has occurred.

IBWAIT

Board Level
Device Level

IBWAIT
(Continued)

Examples

1. Wait for the GPIB-ENET associated with a board-level descriptor previously opened with `ibfind()` to become CIC by being passed control or a timeout.

```
#include "esp.h"

GPIBDESCRIPTOR Board;

ibwait(Board, TIMO|CIC);

if(ibsta & TIMO)
{  /*Did not get CIC within timout period. Process the error*/
   CICError();
}
else
{  /*Received CIC. Proceed with the program*/
   ProcessCIC();
}
```

2. While looping, wait indefinitely for the GPIB-ENET associated with a board-level descriptor previously opened with `ibfind()` to become either a Talker or a Listener.

```
#include "esp.h"

GPIBDESCRIPTOR Board;

for(;;)
{
   ibwait(Board, TACS|LACS);
   if(ibsta & TACS)
   {  /*GPIB-ENET is now a Talker*/
      ProcessTalker();
   }
   else
   {  /*GPIB-ENET is now a Listener*/
      ProcessListener();
   }
}
```

Table 4-3. Wait Mask Layout

Mnemonic	Bit Pos.	Hex Value	Description
TIMO	14	4000	Time limit exceeded
END	13	2000	GPIB board detected END or EOS
SRQI	12	1000	SRQ asserted (board only)
CMPL	8	100	I/O completed
LOK	7	80	GPIB board is in Lockout State
REM	6	40	GPIB board is in Remote State
CIC	5	20	GPIB board is CIC
TACS	3	8	GPIB board is Talker
LACS	2	4	GPIB board is Listener
DTAS	1	2	GPIB board is in Device Trigger State
DCAS	0	1	GPIB board is in Device Clear State

IBWRT

Board Level
Device Level

IBWRT

Purpose

Write data to a device from a user buffer.

Format

```
int16 ibwrt (GPIBDESCRIPTOR Descriptor, int8* Wrtbuf,
             uint32 Cnt)
```

Input

`Descriptor`	Board or device descriptor returned from `ibfind`
`Wrtbuf`	Address of the buffer containing the bytes to write
`Cnt`	Number of bytes to be written

Output

Function Return	The value of `ibsta`

Description

Device Level

If `Descriptor` is a device descriptor, `ibwrt` addresses the GPIB and writes `Cnt` bytes from the memory location specified by `Wrtbuf` to a GPIB device. The operation terminates normally when `Cnt` bytes have been sent. The operation terminates with an error if `Cnt` bytes could not be sent within the timeout period or if there is no listener present on the GPIB. The actual number of bytes transferred is returned in the global variable `ibcntl`.

Board Level

If `Descriptor` is a board descriptor, `ibwrt` writes `Cnt` bytes of data from the buffer specified by `Wrtbuf` to a GPIB device; a board-level `ibwrt` assumes that the GPIB is already properly addressed. The operation terminates normally when `Cnt` bytes have been sent. The operation terminates with an error if `Cnt` bytes could not be sent within the timeout period, if there is no listener present on the GPIB, or, if the board is not CIC, the CIC sends the Device Clear message on the GPIB. The actual number of bytes transferred is returned in the global variable `ibcntl`.

IBWRT **Board Level** **IBWRT**
Device Level

Possible Errors

EABO	Either `Cnt` bytes were not sent within the timeout period or, for board-level writes, a Device Clear message was received after the write operation began.
EADR	Board level: The GPIB is not correctly addressed. Use `ibcmd` to address the GPIB.
EBUS	Device level: No devices are connected to the GPIB.
ECIC	Device level: The access board is not CIC.
EDVR	A system or other unrecoverable error has occurred.
ENOL	No Listeners were detected on the bus.

Examples

1. Using a board-level descriptor previously opened with `ibfind()`, write data to a GPIB device. When using a board-level write, all GPIB addressing must be performed before the `ibwrt()`.

```
#include "esp.h"

GPIBDESCRIPTOR Board;

ibwrt(Board, "OHMS?", 5);      /*Write the command to report the
                                 preset Ohms reading to a
                                 Listen-addressed GPIB device*/
```

2. Using a device-level descriptor previously opened with `ibfind()`, write data to a GPIB device. When using a device-level write, all GPIB addressing is performed by the GPIB-ENET.

```
#include "esp.h"

GPIBDESCRIPTOR Device;

ibwrt(Device, "VOLTS?", 6);    /*Write the command to report the
                                 preset Volts reading to a GPIB
                                 device associated with the
                                 device-level descriptor*/
```

Chapter 5
Using Your ESP-488 Software

This chapter describes ESP-488 coding conventions, explains other programming considerations, and discusses the sample code included on your ESP-488 distribution diskette.

Helpful Source Code Rules

The following is a description of the function and variable naming conventions that have been used to write the ESP-488 code. You may find this information helpful when you review the source code.

- All constants, locals, and global values have meaningful names, with the possible exception of loop counters.
- All constants are capitalized (for example, `DCAS`, `TACS`, `EOI`).
- All status globals are lowercase (`ibsta`, `iberr`, and `ibcntl`).
- All support function names capitalize the first letter in every word (for example, `UpdateStatus()`, `CmdIoctl()`).
- The GPIB support functions are all lowercase (for example, `ibfind()`, `ibrd()`, `ibwait()`).
- There are four network support functions that you may need to modify to support the networking software interface for your system. These functions begin with TCP (`TCPOpen()`, `TCPClose()`, `TCPRead()`, and `TCPWrite()`).
- Local variables capitalize the first letter in every word (for example, `HostName`, `ConnectionIndex`).
- The common data types (`char`, `int`, and `long`) have been replaced with meaningful typedefs (`int8`, `uint8`, `int16`, `uint16`, `int32`, and `uint32`). Change these typedefs to match the correct types for your compiler.

Programming Considerations

- You must include the appropriate header files to describe the prototypes for functions called in the program.
- Programming the ESP-488 software is very similar to using National Instruments NI-488.2 software.
- A GPIB Controller must appropriately address the device program to talk or listen before board-level I/O functions can be implemented.
- Three status variables—`ibsta`, `iberr`, and `ibcntl`—contain important information about the status of the function call last completed.
- The ESP-488 code was developed on a SPARC platform under Solaris 1. The code should be easily portable to other C compilers. Programming optimizations have not been made, so the code is easier to read.
- The code is written to easily work with network software based on the Berkeley Sockets interface. If you do not have access to such an interface, you can adapt the ESP-488 code to work with your network software by changing the functions `TCPOpen`, `TCPClose`, `TCPRead`, and `TCPWrite`.
- The `esp.h` file contains a section that you can configure easily. The section is marked by the words **`BEGIN USER CONFIGURATION SECTION`** and **`END USER CONFIGURATION SECTION`**. Although you can make other changes in this file, changes made in other sections of the code might cause problems.

Compiling a C Program with the ESP-488 Package

In addition to any other necessary header files for the program, you should include the following header file for the ESP-488 package:

```
#include "esp.h"
```

You should enable ANSI C compilation if your compiler has this option. When you enable this feature and include the prototypes, you ensure that your parameters match what is required by ESP-488.

ESP-488 Example

The ESP-488 distribution diskette contains a sample application, `ibic.c`, for use with ESP-488.

`ibic` is an interactive control utility for entering ESP-488 functions and displaying the results of each call.

After you have compiled `ibic.c` with the ESP-488 code, you can use the utility to do the following:

- See the ESP-488 software work in a real application
- Learn the ESP-488 functions before writing an application
- Verify network communications between your computer and a GPIB-ENET
- Verify GPIB communications with your device quickly and easily
- Become familiar with the commands of your GPIB device
- Receive data from your GPIB device
- Troubleshoot problems with your application

Table 5-1 lists the necessary parameters, the type, and the purpose for each of the commands that you can use with `ibic`. The *Example* command type demonstrates ESP-488 commands, and the *Support* command type makes `ibic` more useful.

Note: ***You can add other commands to*** `ibic` ***by modifying the input command array*** `cmds[ ]`.

Table 5-1. ibic Commands

Parameters	Type	Purpose
`$ <filename>`	Support	Executes a text file, represented by `<filename>`, of `ibic` commands.
`buffer <number>`	Support	Sets the display options for data buffers returned from GPIB reads. The valid parameters are: 0 = Buffer printing off. 1 = Hex display of buffer data. 2 = Display actual characters in buffer. 3 = Hex display of buffer data followed by a display of the actual characters in the buffer.
`cmd <command bytes>`	Example	Demonstrates the ESP-488 `ibcmd` function. `command bytes` are issued as GPIB command data (ATN asserted). In `ibic`, the command bytes cannot start with any whitespace (tab, space, line feed, and so on).
`config <option> <value>`	Example	Demonstrates the ESP-488 `ibconfig` function. It is used to configure GPIB characteristics.
`find <GPIB-ENET network name> <type>`	Example	Demonstrates the ESP-488 `ibfind` function. It is used to open a GPIB descriptor of the requested type in the specified GPIB-ENET. `type` is the string “board” or the string “device.”
`q, quit`	Support	Both of these commands exit `ibic`.
`onl <v>`	Example	Demonstrates ESP-488 `ibonl` function. It is used to place the GPIB descriptor ONLINE(v = non-zero) or OFFLINE (v = zero).
`pad <primary address>`	Example	Demonstrates the ESP-488 `ibconfig` function being used to implement other functions. It is used to configure the primary GPIB address of the associated descriptor.
`rd <count>`	Example	Demonstrates the ESP-488 `ibrd` function. It is used to read up to `count` bytes of data from the GPIB.
`rpp`	Example	Demonstrates the ESP-488 `ibrpp` function. It is used to perform a parallel poll.
`rsp`	Example	Demonstrates the ESP-488 `ibrsp` function. It is used to serial poll a device to obtain the device’s status byte.

(continues)

Table 5-1. ibic Commands (Continued)

Parameters	Type	Purpose
`sad <secondary address>`	Example	Demonstrates the ESP-488 `ibconfig` function being used to implement other functions. It is used to configure the secondary GPIB address of the associated descriptor.
`sic`	Example	Demonstrates the ESP-488 `ibsic` function. It is used to make the GPIB-ENET CIC by asserting the GPIB IFC signal.
`tmo <I/O timeout>`	Example	Demonstrates the ESP-488 `ibconfig` function being used to implement other functions. It is used to configure the I/O timeout in effect for an associated descriptor.
`Wait <Wait Mask>`	Example	Demonstrates the ESP-488 `ibwait` function. It is used to wait for one or more of the specified events to occur.
`wrt <data bytes>`	Example	Demonstrates the ESP-488 `ibwrt` function. It is used to send `data bytes` across the GPIB. In `ibic`, the data bytes cannot start with any whitespace (tab, space, line feed, and so on).

Chapter 6
Verification and Troubleshooting

This chapter describes how to verify the hardware installation and troubleshoot problems.

Verify the Hardware Installation

When you power on your GPIB-ENET, the **POWER** LED comes on immediately. The **READY** LED flashes while it completes its power-on self-tests. When the tests complete successfully and the IP address is assigned, the **READY** LED remains steady, indicating that the unit is ready to operate.

The power-on self tests take about 10 seconds to complete. The time required for the IP address assignment is highly dependent on your network and the configuration of your GPIB-ENET. If the **POWER** LED does not come on immediately or the **READY** LED continues to flash for more than 1 minute, refer to the following section, *Troubleshooting Hardware Problems*.

Troubleshooting Hardware Problems

Warning: ***The GPIB-ENET contains circuitry that operates with hazardous voltages. Do not attempt to open and service the box. Refer servicing to qualified personnel.***

- Verify that all cables are securely connected to the GPIB-ENET.
- Verify that the GPIB-ENET is powered on. If the unit is powered on and plugged into an outlet of the proper voltage, yet you see no LED activity, check the condition of the fuse.

Warning: ***For continued protection against fire, replace the fuse with only the same type and rating of fuse. See Appendix B,*** *Hardware Specifications****, for fuse specifications.***

- Verify that the Ethernet port configuration slide switch is set to the correct Ethernet port. Refer to the *Configure the Slide Switch* section of Chapter 2, *Configuring the Hardware*, for more information.
- Verify that the DIP switch settings are configured properly for your setup. Refer to the *Configure the 8-Bit DIP Switch* section of Chapter 2, and also to Appendix C, *GPIB-ENET 8-Bit DIP Switch*, for more information.
- Verify with the system administrator that your IP address is valid and that the network is set up to recognize the IP address properly.

- If you configured the IP address manually using the `IPassign` utility, verify that you have followed the steps in Appendix D, *GPIB-ENET Configuration Utilities*, correctly.

- If the **READY** LED continues to flash for more than 1 minute, you have a problem with your setup. If the **READY** LED is flashing quickly, the GPIB-ENET is unable to obtain its IP address from the network. In this case, check with your network administrator to verify your IP address and make sure that you recorded the Ethernet address correctly. If the **READY** LED is blinking slowly, the GPIB-ENET has an internal error. In this case, refer to Appendix E, *READY LED Signaling*, to determine which error is being reported and contact National Instruments.

Common Questions

My software is not communicating with the GPIB-ENET. What should I do?

Make sure that the GPIB-ENET is configured and installed properly. To isolate the problem, try to *ping* your GPIB-ENET to make sure it is recognizable on the network. `ping` is a network utility that sends a packet to a device/host on the network, then waits for it to be echoed back, which indicates an active device/host. Typically, you can use one of the following commands:

```
ping <IPaddress>
ping <Network_name>
```

where *IPaddress* is the IP address assigned to your GPIB-ENET, and *Network_name* is the name assigned to your GPIB-ENET. If `ping` is unable to communicate with your GPIB-ENET, make sure that your network is properly configured for the GPIB-ENET and that the GPIB-ENET has an IP address. Refer to Chapter 2, *Configuring the Hardware*, for more information.

The `Update` utility returns a message about a checksum failure. What should I do?

Run the `Update` utility again. Your imagefile might be corrupted. If the same message appears when you run the utility again, try reinstalling your software and then running the `Update` utility again.

I powered off my GPIB-ENET before recording the flashing READY LED pattern. What should I do?

Try to duplicate the error before calling National Instruments for assistance. Recording the **READY** LED pattern is not critical to the operation of your GPIB-ENET, but it saves you time and helps Product Support diagnose the problem quickly and accurately.

Where can I obtain the utilities referenced in this manual?

The GPIB-ENET requires several utilities to run correctly. These utilities are not available in source code form. However, the ESP-488 distribution diskette includes executable versions of these utilities for the Solaris 1 and Solaris 2 SPARC platforms, and the HP-UX Series 700 platform. Executables for other common platforms are also available from National Instruments. If you cannot use any of the included executables, contact National Instruments for assistance.

What information should I have before I contact National Instruments?

When you contact National Instruments, make sure you have filled out the configuration form in Appendix F, *Customer Communication.*

Appendix A
Multiline Interface Messages

This appendix contains a multiline interface message reference list, which describes the mnemonics and messages that correspond to the interface functions. These multiline interface messages are sent and received with ATN TRUE.

For more information on these messages, refer to the ANSI/IEEE Standard 488.1-1987, *IEEE Standard Digital Interface for Programmable Instrumentation*.

Multiline Interface Messages

Hex	Oct	Dec	ASCII	Msg	Hex	Oct	Dec	ASCII	Msg
00	000	0	NUL		20	040	32	SP	MLA0
01	001	1	SOH	GTL	21	041	33	!	MLA1
02	002	2	STX		22	042	34	"	MLA2
03	003	3	ETX		23	043	35	#	MLA3
04	004	4	EOT	SDC	24	044	36	$	MLA4
05	005	5	ENQ	PPC	25	045	37	%	MLA5
06	006	6	ACK		26	046	38	&	MLA6
07	007	7	BEL		27	047	39	'	MLA7
08	010	8	BS	GET	28	050	40	(	MLA8
09	011	9	HT	TCT	29	051	41	)	MLA9
0A	012	10	LF		2A	052	42	*	MLA10
0B	013	11	VT		2B	053	43	+	MLA11
0C	014	12	FF		2C	054	44	,	MLA12
0D	015	13	CR		2D	055	45	-	MLA13
0E	016	14	SO		2E	056	46	.	MLA14
0F	017	15	SI		2F	057	47	/	MLA15
10	020	16	DLE		30	060	48	0	MLA16
11	021	17	DC1	LLO	31	061	49	1	MLA17
12	022	18	DC2		32	062	50	2	MLA18
13	023	19	DC3		33	063	51	3	MLA19
14	024	20	DC4	DCL	34	064	52	4	MLA20
15	025	21	NAK	PPU	35	065	53	5	MLA21
16	026	22	SYN		36	066	54	6	MLA22
17	027	23	ETB		37	067	55	7	MLA23
18	030	24	CAN	SPE	38	070	56	8	MLA24
19	031	25	EM	SPD	39	071	57	9	MLA25
1A	032	26	SUB		3A	072	58	:	MLA26
1B	033	27	ESC		3B	073	59	;	MLA27
1C	034	28	FS		3C	074	60	<	MLA28
1D	035	29	GS		3D	075	61	=	MLA29
1E	036	30	RS		3E	076	62	>	MLA30
1F	037	31	US		3F	077	63	?	UNL

Message Definitions

DCL	Device Clear
GET	Group Execute Trigger
GTL	Go To Local
LLO	Local Lockout
MLA	My Listen Address
MSA	My Secondary Address
MTA	My Talk Address
PPC	Parallel Poll Configure
PPD	Parallel Poll Disable

Multiline Interface Messages

Hex	Oct	Dec	ASCII	Msg	Hex	Oct	Dec	ASCII	Msg
40	100	64	@	MTA0	60	140	96	`	MSA0,PPE
41	101	65	A	MTA1	61	141	97	a	MSA1,PPE
42	102	66	B	MTA2	62	142	98	b	MSA2,PPE
43	103	67	C	MTA3	63	143	99	c	MSA3,PPE
44	104	68	D	MTA4	64	144	100	d	MSA4,PPE
45	105	69	E	MTA5	65	145	101	e	MSA5,PPE
46	106	70	F	MTA6	66	146	102	f	MSA6,PPE
47	107	71	G	MTA7	67	147	103	g	MSA7,PPE
48	110	72	H	MTA8	68	150	104	h	MSA8,PPE
49	111	73	I	MTA9	69	151	105	i	MSA9,PPE
4A	112	74	J	MTA10	6A	152	106	j	MSA10,PPE
4B	113	75	K	MTA11	6B	153	107	k	MSA11,PPE
4C	114	76	L	MTA12	6C	154	108	l	MSA12,PPE
4D	115	77	M	MTA13	6D	155	109	m	MSA13,PPE
4E	116	78	N	MTA14	6E	156	110	n	MSA14,PPE
4F	117	79	O	MTA15	6F	157	111	o	MSA15,PPE
50	120	80	P	MTA16	70	160	112	p	MSA16,PPD
51	121	81	Q	MTA17	71	161	113	q	MSA17,PPD
52	122	82	R	MTA18	72	162	114	r	MSA18,PPD
53	123	83	S	MTA19	73	163	115	s	MSA19,PPD
54	124	84	T	MTA20	74	164	116	t	MSA20,PPD
55	125	85	U	MTA21	75	165	117	u	MSA21,PPD
56	126	86	V	MTA22	76	166	118	v	MSA22,PPD
57	127	87	W	MTA23	77	167	119	w	MSA23,PPD
58	130	88	X	MTA24	78	170	120	x	MSA24,PPD
59	131	89	Y	MTA25	79	171	121	y	MSA25,PPD
5A	132	90	Z	MTA26	7A	172	122	z	MSA26,PPD
5B	133	91	[	MTA27	7B	173	123	{	MSA27,PPD
5C	134	92	\	MTA28	7C	174	124	|	MSA28,PPD
5D	135	93	]	MTA29	7D	175	125	}	MSA29,PPD
5E	136	94	^	MTA30	7E	176	126	~	MSA30,PPD
5F	137	95	_	UNT	7F	177	127	DEL	

PPE	Parallel Poll Enable	SPE	Serial Poll Enable
PPU	Parallel Poll Unconfigure	TCT	Take Control
SDC	Selected Device Clear	UNL	Unlisten
SPD	Serial Poll Disable	UNT	Untalk

Appendix B
Hardware Specifications

This appendix lists the electrical, environmental, and physical characteristics of the GPIB-ENET and the recommended operating conditions.

Table B-1. Electrical Characteristics

Characteristic	Specification
Power Supply Unit	100-120 VAC ± 10%, 50-60 Hz or 220-240 VAC ± 10%, 50-60 Hz
Maximum Current Requirement	100-120 VAC 110 mA or 220-240 VAC 55 mA
Fuse Rating and Type	100-120 VAC 300 mA, UL/CSA approved or 220-240 VAC 500 mA, IEC approved

Table B-2. Environmental Characteristics

Characteristic	Specification
Operating Temperature	0° to 40° C
Storage Temperature	-20° to 70° C
Relative Humidity	10% to 90% noncondensing conditions
EMI	FCC Class A Verified

Table B-3. Physical Characteristics

Characteristic	Specification
Overall Case Size (Dimensions)	8.89 cm by 14.35 cm by 4.11 cm (3.5 in by 5.65 in. by 1.62 in.)
Case Material	All metal enclosure
Weight	0.41 kg (0.9 lb)

Appendix C
GPIB-ENET 8-Bit DIP Switch

This appendix describes how the DIP switch on the back panel affects the operation of the GPIB-ENET.

The 8-bit DIP switch is located on the back panel of the GPIB-ENET, as shown in Figure C-1. The DIP switches are used to set the operation mode of the GPIB-ENET. The GPIB-ENET is shipped to you with the DIP switches set for normal operation mode; all the switches are in the OFF position.

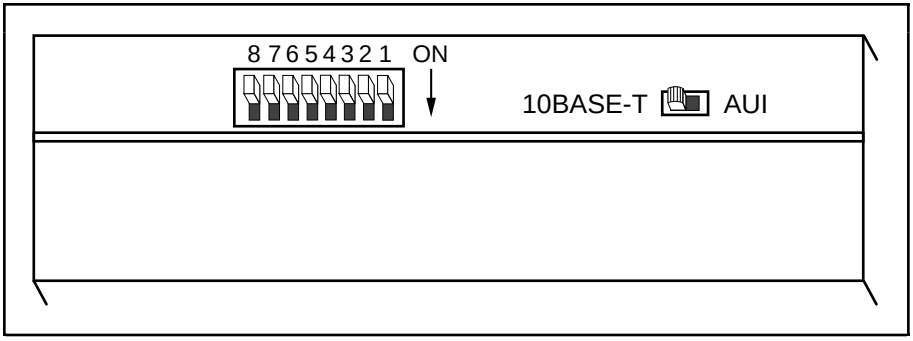


Figure C-1. GPIB-ENET Back Panel Switches

8-Bit DIP Switch Descriptions

Table C-1 shows the possible configurations of the GPIB-ENET DIP switches and what each configuration indicates. The entries in ***bold italic*** text represent the default settings.

Table C-1. DIP Switch Settings for Modes of Operation

Switch	Position	Indication
8	ON	Extended Factory Test
	OFF	***Normal Operation***
7	ON	Modify Mode
	OFF	***Normal Operation***
6	ON	Use Stored IP Address
	OFF	***Obtain IP Address***
5	ON	Manual IP Assignment
	OFF	***Automatic IP Assignment***
1-4	***OFF***	***These switches are reserved and should remain OFF.***

Switch 8 selects the Extended Factory Test mode and should remain OFF for normal operation mode. When this switch is ON, the GPIB-ENET performs extended tests which require special connections to the GPIB and Ethernet ports. Without these connections, the GPIB-ENET cannot pass the extended factory tests.

Switch 7 selects the Modify Mode and should remain OFF for normal operation mode. The firmware controlling the GPIB-ENET is stored in an electrically erasable memory. When this switch is ON, you can reprogram the firmware memory from the Ethernet network using the `Update` utility, or you can modify the subnet information using the `IPsetup` utility. Refer to Appendix D, *GPIB-ENET Configuration Utilities*, for more information.

Switch 6 indicates how the GPIB-ENET IP address is determined when it is powered on. If switch 6 is OFF, the GPIB-ENET expects to receive its IP address from the network, and then the GPIB-ENET stores the address in nonvolatile memory. If switch 6 is ON, the GPIB-ENET retrieves the IP address that is stored in nonvolatile memory.

Switch 5 selects the method by which IP addresses are configured in the GPIB-ENET. When switch 5 is OFF, the GPIB-ENET issues Reverse Address Resolution Protocol (RARP) requests each time it is powered on. The GPIB-ENET continues to issue RARP requests until a valid RARP reply is received. If switch 5 is ON, the GPIB-ENET does not issue RARP requests. In most networks, the GPIB-ENET can use RARP to have its address automatically configured; however, there are some networks which, for security or other reasons, do not use RARP. In this case, you must configure the IP address of the GPIB-ENET using `IPassign`. Refer to Appendix D, *GPIB-ENET Configuration Utilities*, for more information on `IPassign`. Notice that if switch 6 is ON, the setting of Switch 5 is ignored.

Switches 1-4 are reserved for future expansion and should remain OFF.

Appendix D
GPIB-ENET Configuration Utilities

This appendix contains information on the `IPassign`, `Update`, and `IPsetup` utilities for the GPIB-ENET.

The `IPassign`, `Update`, and `IPsetup` utilities are not available in source code. However, the ESP-488 distribution diskette includes executable versions of these utilities for the Solaris 1 and Solaris 2 SPARC platforms, and the HP-UX Series 700 platform. Executables for other common platforms are also available from National Instruments. If you cannot use any of the included executables, contact National Instruments for assistance.

IPassign Utility

Use the `IPassign` utility to manually configure the IP address of your GPIB-ENET. Once you configure the IP address, it is stored in nonvolatile memory in the GPIB-ENET. You only need to run the `IPassign` utility when the box is new or when you decide to reconfigure your network in a way that changes the IP address of the GPIB-ENET.

The `IPassign` utility must be run on a machine that is connected to the same subnet as the GPIB-ENET you are configuring.

To run `IPassign`, follow these steps after you have connected the GPIB-ENET to the network.

1. Make sure you know the Ethernet address of the GPIB-ENET (listed on the bottom panel of the box) and the IP address of your GPIB-ENET (assigned by your network administrator). If you do not know the IP address, repeat Step 1 in Chapter 2, *Configuring the Hardware*.

2. Power off the GPIB-ENET. Make sure that all of the DIP switches are OFF except for switch 5, which should be ON. This indicates that you want to use `IPassign` to configure the IP address.

3. Power on the GPIB-ENET and wait 10 seconds for the power-on self tests to complete. The **READY** LED will continue to flash.

4. Run `IPassign`. This utility is always invoked with two parameters: an Ethernet address and an IP address. These addresses are entered in standard notation, which for the Ethernet address is six hexadecimal numbers separated by colons. Standard notation for the IP address is four decimal numbers separated by periods. For example, the following numbers program the IP address 130.164.1.1 into the GPIB-ENET located at Ethernet address 00:80:2F:FF:00:10.

```
IPassign  00:80:2F:FF:00:10   130.164.1.1
```

5. The utility constructs 10 packets and sends them across the local Ethernet. When the GPIB-ENET successfully receives its IP address, the **READY** LED remains steady. As long as the **READY** LED is not steady, the address has not been configured.

   Because the reply packets do not form a guaranteed delivery protocol, it is possible (though extremely unlikely) for all 10 packets to get lost in the network and for the GPIB-ENET not to receive any of them. In this case, repeat Step 4 several times until the GPIB-ENET successfully receives its new IP address, indicated by a steady **READY** LED. If after several attempts, the **READY** LED does not become steady, contact National Instruments for further assistance.

6. Power off the GPIB-ENET. Set DIP switch 5 to the OFF position and switch 6 to the ON position. With the switches in this position, the GPIB-ENET powers on with the IP address stored in nonvolatile memory. For more information on the DIP switch settings, refer to Appendix C, *GPIB-ENET 8-Bit DIP Switch*.

7. Power on the GPIB-ENET and wait for the **READY** LED to come on and remain steady. If the **READY** LED does not become steady, the slow blinking indicates an internal GPIB-ENET error. Refer to Appendix E, *READY LED Signaling*, to determine which error is being reported.

Update Utility

Electrically Erasable Programmable Read Only Memory (EEPROM) stores the firmware that controls the GPIB-ENET. You can use the `Update` utility to upgrade the firmware in the EEPROM. The latest firmware file is called `enet_xx.bin`, where *xx* stands for the latest firmware revision number.

Caution: ***Because of the potential loss of EEPROM functionality, it is extremely important to follow the instructions in this section. Please read this entire section before attempting to update your firmware.***

If you are directly linked to the Internet, and not just to an isolated network using IP protocols, you can contact the National Instruments GPIB Product Support Department to update your firmware. If you are not on the actual Internet or you want to update the firmware yourself, you can execute `Update` locally. In either case, pay close attention to the GPIB-ENET **READY** LED, which reports important status codes.

Although you should update your firmware as needed, the number of times you can do so is limited. Therefore, update the firmware only when necessary. Also, notice that there is a time period of about two seconds when the GPIB-ENET runs exclusively in volatile RAM memory with the EEPROMs erased. If the box loses power during this time, it is no longer usable and must be returned to National Instruments for repair. If you are not comfortable with this procedure, call National Instruments for assistance.

Caution: ***Because of the potential loss of EEPROM information, do not attempt to update the firmware when your electrical power is in danger; that is, during a weather storm or similar situation. Once you have started the update process, do not switch off power to your GPIB-ENET until you see and record the flashing* READY *LED pattern.***

The following steps describe how to update the firmware.

1. Power off the GPIB-ENET and set DIP switch 7 of the 8-bit DIP switch to the ON position.

2. Power on the GPIB-ENET and wait for the **READY** LED light to remain steady.

3. Run `Update`. This utility is always invoked with two parameters: an imagefile name and a hostname. For example, the following command copies the imagefile `enet_a3.bin` into the EEPROMs of `gpib0.natinst.com`.

   ```
   update enet_a3.bin gpib0.natinst.com
   ```

4. Wait for the **READY** LED to begin flashing, which signals that the update is complete. Do *not* power off your GPIB-ENET before you see and record the flashing **READY** LED pattern.

   If the programming is successful, the **READY** LED signals a status code of 00, signified by one long flash and one short flash. Other patterns are used to indicate that the EEPROMs did not program properly. If a pattern other than 00 is indicated, refer to Appendix E, *READY LED Signaling*, for more information on recording the flashing pattern.

5. Power off the GPIB-ENET and return DIP switch 7 to the OFF position.

6. Power on the GPIB-ENET and wait for the **READY** LED to remain steady. If the **READY** LED does not remain steady, repeat the update procedure.

IPsetup Utility

Use the `IPsetup` utility to configure the IP characteristics of the specified GPIB-ENET. The IP characteristics include the following:

- Broadcast IP address—used by an IP device when a packet must go to all devices on the local subnet.

- Netmask of the local subnet—used by an IP device to determine if the destination of an out-going packet is local (that is, on the same subnet). If the destination is local, no routing is required. If the destination is not local, routing is required.

- List of router IP addresses—used when an IP device determines that the destination of a packet is not on the local subnet. If routing is required, the best route is chosen dynamically. Up to four default routers can be specified.

Note: ***You should run this utility with assistance from your network administrator*.**

The following steps describe how to configure the IP characteristics:

1. Power off the GPIB-ENET, and set DIP switch 7 to the ON position.
2. Power on the GPIB-ENET and wait for the **READY** LED light to remain steady.
3. Run `IPsetup`. This utility takes no command line parameters. All necessary information is obtained during execution.
4. Enter the appropriate information as it is requested. For help information, enter a single question mark (?).

   To remove the current list of routers for the GPIB-ENET, run the `IPsetup` utility without any routers specified.
5. Power off the GPIB-ENET, and return DIP switch 7 to the OFF position.
6. Power on the GPIB-ENET and wait for the **READY** LED to remain steady.

Appendix E
READY LED Signaling

This appendix describes how to interpret the **READY** LED error codes.

READY LED Overview

The **READY** LED has several purposes on the GPIB-ENET. When you first power on the GPIB-ENET, the **READY** LED flashes quickly while it completes its power-on self-tests. When the tests complete successfully and the IP address is assigned from either nonvolatile memory or the network, the **READY** LED remains steady, indicating that the unit is ready to operate.

During operation, the **READY** LED might begin to blink slowly. This occurs after running the `Update` utility, when the GPIB-ENET reports status on the operation. Refer to Appendix D, *GPIB-ENET Configuration Utilities*, for more information. At other times, the **READY** LED blinks slowly to alert you of internal GPIB-ENET errors. For assistance in correcting this problem, use this appendix to interpret and record the pattern that the **READY** LED flashes, and then contact National Instruments.

Note: ***By recording the*** READY ***LED status messages before calling National Instruments, you can save yourself time, and the GPIB Product Support Department can answer your questions more accurately and efficiently. Do not switch off power to your GPIB-ENET before recording the flashing*** READY ***LED pattern.***

READY LED signaling can report up to 100 different errors. The errors are numbered from 0 to 99 and are reported through sequences of **READY** LED flashes.

Step 1. Count the Long Flashes

A 3-second interval, during which the **READY** LED is OFF, separates each repetition of the sequence. The sequence begins with a series of long 1-second flashes—that is, one second ON, one second OFF. These long flashes represent the digit in the tens column. There can be 1 to 10 long flashes, which represent digits 0 through 9. For example, one long flash represents the digit 0 in the tens column, two long flashes represent the digit 1 in the tens column, and 10 long flashes represent the digit 9 in the tens column.

Step 2. Count the Short Flashes

The long flashes are followed by shorter flashes; each short flash lasts about one-fifth of a second. These short flashes represent the digit in the ones column. Again, there can be 1 to 10 flashes, which represent the digits 0 through 9. For example, one short flash represents the digit 0 in the ones column, two short flashes represent the digit 1 in the ones column, and 10 short flashes represent a 9 in the ones column.

Using this method, the **READY** LED flashes the following sequence to represent status message 11.

<three seconds OFF> <two long flashes> <two short flashes> <three seconds OFF>. . .

The **READY** LED flashes the following sequence to represent status message 30.

<three seconds OFF> <four long flashes> <one short flash> <three seconds OFF>. . .

Step 3. Record Your Status Code Number

When you have computed your status code number, record it on the *Hardware and Software Configuration Form* in Appendix F, *Customer Communication*, before calling National Instruments.

Table E-1 lists some examples of the long and short flashes and the status codes they report.

Table E-1. Sample **READY** LED Signals and the Corresponding Status Code Numbers

Number of Long Flashes	Number of Short Flashes	Corresponding Status Code Number
1	1	00
2	1	10
5	3	42
1	8	07
3	4	23
10	10	99

Appendix F
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.

Corporate Headquarters
(512) 795-8248
Technical support fax: (800) 328-2203
(512) 794-5678

Branch Offices	**Phone Number**	**Fax Number**
Australia	(03) 879 9422	(03) 879 9179
Austria	(0662) 435986	(0662) 437010-19
Belgium	02/757.00.20	02/757.03.11
Denmark	45 76 26 00	45 76 71 11
Finland	(90) 527 2321	(90) 502 2930
France	(1) 48 14 24 00	(1) 48 14 24 14
Germany	089/741 31 30	089/714 60 35
Italy	02/48301892	02/48301915
Japan	(03) 3788-1921	(03) 3788-1923
Mexico	95 800 010 0793	95 800 010 0793
Netherlands	03480-33466	03480-30673
Norway	32-848400	32-848600
Singapore	2265886	2265887
Spain	(91) 640 0085	(91) 640 0533
Sweden	08-730 49 70	08-730 43 70
Switzerland	056/20 51 51	056/20 51 55
Taiwan	02 377 1200	02 737 4644
U.K.	0635 523545	0635 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. Use additional pages if necessary.

Name _____

Company _____

Address _____

_____

Fax (___)_____ Phone (___)_____

Computer brand _____

Model _____ RAM _____MB

Processor _____ Speed _____MHz

Operating system _____

Display adapter _____

Mouse _____ yes _____ no

Other adapters installed_____

Hard disk capacity _____ MB Brand _____

Instruments used _____

National Instruments hardware product model _____

Revision _____

Configuration _____

(continues)

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. Update this form each time you revise your software or hardware configuration, and use this form as a reference for your current configuration.

National Instruments Products

- GPIB-ENET Hardware Revision _____
- 10Base-T or Coax Version _____
- AC Input Voltage (100-120 V or 220-240 V) _____
- DIP Switch Settings _____
- Ethernet Port Configuration Slide Switch Setting _____
- IP Address _____
- **READY** LED Signaling Status Code _____
- ESP-488 Software in C for the GPIB-ENET Revision Number on Disk _____

Other Products

- Computer Make and Model _____
- Memory Capacity on Computer _____
- Operating System Version _____
- Application Programming Language _____
- Other GPIB Devices in System _____
- Type of Monitor _____

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: **ESP-488 Software Reference Manual for the GPIB-ENET**

Edition Date: **February 1995**

Part Number: **320910A-01**

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

(continues)

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, MS 53-02
Austin, TX 78730-5039

Fax to: Technical Publications
National Instruments Corporation
MS 53-02
(512) 794-5678

Glossary

Prefix	Meaning	Value
n-	nano-	10^{-9}
µ-	micro-	10^{-6}
m-	milli-	10^{-3}
c-	centi-	10^{2}
K-	kilo-	10^{3}
M-	mega-	10^{6}
G-	giga-	10^{9}

o	degrees
%	percent
A	amperes
AC	alternating current
ANSI	American National Standards Institute
AUI	attachment unit interface
C	Celsius
CIC	Controller-in-Charge
CSA	Canadian Standards Association
DIP	dual inline package
EEPROM	Electrically Erasable Programmable Read Only Memory
EOI	end or identify
EOS	End-of-String
EMI	electromagnetic interference
ESP	Engineering Software Package
FCC	Federal Communications Commission
g	grams
GB	gigabytes
GPIB	General Purpose Interface Bus
hex	hexadecimal
Hz	hertz
IEC	International Electrotechnical Commission
IEEE	Institute of Electrical and Electronic Engineers
in.	inches
I/O	input/output
IP	Internet Protocol
lb	pounds
LED	light-emitting diode
m	meters
MB	megabytes of memory
RAM	random-access memory
RARP	Reverse Address Resolution Protocol

s	seconds
TCP	Transmission Control Protocol
UL	Underwriters Laboratories
V	volts
VAC	volts alternating current

Index

A

addresses
    Ethernet address, 2-1
    IP address
        automatic assignment, 2-4
        configuring with IPassign, 2-2, D-1 to D-2
        determining, 2-1 to 2-2
ATN, 3-2, 3-4

C

C language library. *See* ESP-488 C language library.
Cables
    connecting the cables, 2-3
    Ethernet cables, 1-1
    shielded GPIB cables, 1-2
CIC, 3-2, 3-4
CMPL, 3-2, 3-3
common questions, 6-2 to 6-3
configuration utilities
    IPassign utility, 2-1, D-1 to D-2
    IPsetup, 2-4, D-3 to D-4
    Update utility, 6-2, D-2 to D-3
configuring the GPIB-ENET
    8-bit DIP switch, 2-3, C-1 to C-2
    slide switch, 2-2
        default setting for Ethernet port (picture), 2-3
        setting for AUI Ethernet port (picture), 2-3
    subnet information, 2-4
count variable. *See* ibcntl variable.
customer communication, *xi*, F-1

D

DCAS, 3-2, 3-5
DIP switch, 8-bit
    configuring, 2-3
    description, C-1 to C-2
    settings for operation modes (table), C-2
documentation
    conventions used in manual, *x*
    organization of manual, *ix* to *x*
    related documentation, *xi*
DTAS, 3-2, 3-5

E

EABO error code, 3-9
EADR error code, 3-8
EARG error code, 3-8
EBUS error code, 3-9
ECAP error code, 3-9
ECFG error code, 3-10
ECIC error code, 3-7
EDVR error code, 3-7
ELCK error code, 3-10
END, 3-2, 3-3
ENOL error code, 3-7 to 3-8
EOS, 3-2, 3-3
ERR, 3-2, 3-3
Error codes (table), 3-6
Error variable. *See* iberr variable.
ESAC error code, 3-8
ESP-488 C language library
    function descriptions. *See* function names.
    global variables
        ibcntl, 3-1, 3-11
        iberr, 3-1, 3-6 to 3-10
        ibsta, 3-1, 3-2 to 3-5

ESP-488 examples, 5-3 to 5-5
ESP-488 functions. *See also* function names.
    board-level functions (table), 3-11
    device-level functions (table), 3-11
ESP-488 software. *See also* programming.
    compiling a C program, 5-2
    ESP-488 examples, 5-3 to 5-5
    files on distribution diskette, 1-4
    kit contents, 1-1
    programming considerations, 5-2
    rules for using source code, 5-1
    software description, 1-3
Ethernet address, 2-1
event handling, DCAS and DTAS (note), 3-5

G

GPIB-ENET
    back panel switches (picture), 2-2, C-1
    bottom panel (picture), 2-1
    configuring, 2-2 to 2-3
    description, 1-2
    hardware kits, 1-1
    installation, verifying, 6-1
    specifications, B-1
    top panel and LEDs (picture), 1-2
    troubleshooting problems, 6-1 to 6-2

H

hardware
    description, 1-2
    installation, verifying, 6-1
    problems, troubleshooting, 6-1 to 6-2

I

ibic.c, 5-3
ibic commands (table), 5-4 to 5-5
ibcmd function, 4-2 to 4-3
ibconfig function, 4-4 to 4-12
    board configuration parameter options (table), 4-6 to 4-8
    description, 4-4
    device configuration parameter options (table), 4-10 to 4-11
    option constants (list), 4-5, 4-9
ibcntl variable, 3-1, 3-11
iberr variable
    definition, 3-1
    descriptions (table), 3-6
ibfind function, 4-13 to 4-14
ibonl function, 4-15 to 4-16
ibrd function, 4-17 to 4-18
ibrpp function, 4-19
ibrsp function, 4-20 to 4-21
ibsic function, 4-22
ibsta. *See* status word condition.
ibwait function
    description, 4-23
    examples, 4-24
    wait mask layout (table), 4-25
ibwrt function, 4-26 to 4-27
interface messages, multiline, A-1 to A-3
IP address
    automatic assignment, 2-4
    configuring with IPassign, 2-2, D-1 to D-2
    determining, 2-1 to 2-2
IPassign utility, 2-1, D-1 to D-2
IPsetup, 2-1, D-3 to D-4

L

LACS, 3-2, 3-5
LED descriptions (table), 1-3
LOK, 3-2, 3-4

M

manual. *See* documentation.
multiline interface messages, A-1 to A-3

P

programming
    compiling C programs, 5-2
    considerations, 5-2
    ESP-488 examples, 5-3
    rules for using source code, 5-1

R

READY LED Signaling
    determining status code, E-1 to E-2
    examples, E-2
    overview, E-1
REM, 3-2, 3-4

S

software description, 1-3
specifications, GPIB-ENET, B-1
    electrical characteristics, B-1
    environmental characteristics, B-1
    physical characteristics (table), B-1
SRQI, 3-2, 3-3
status, global variables
    ibcntl, 3-1, 3-11
    iberr, 3-1, 3-6 to 3-10
    ibsta, 3-1, 3-2 to 3-5
status word condition, 3-1, 3-2 to 3-5
    ATN, 3-2, 3-4
    CIC, 3-2, 3-4
    CMPL, 3-2, 3-3
    DCAS, 3-2, 3-5
    definition, 3-1
    DTAS, 3-2, 3-5
    END, 3-2, 3-3
    ERR, 3-2, 3-3
    LACS, 3-2, 3-5
    list of bits (table), 3-2
    LOK, 3-2, 3-4
    REM, 3-2, 3-4
    SRQI, 3-2, 3-3
    TACS, 3-2, 3-4
    TIMO, 3-2, 3-3
subnet configuration, 2-4
switches
    8-bit DIP switch, 2-3, C-1 to C-2
    slide switch, 2-2
        default setting for Ethernet port (picture), 2-3
        setting for AUI Ethernet port (picture), 2-3

T

TACS, 3-2, 3-4
technical support, F-1
TIMO, 3-2, 3-3
troubleshooting hardware problems, 6-1 to 6-2

U

Update utility, 6-2, D-2 to D-3
utilities. *See* configuration utilities.

W

wait mask layout (table), 4-25

Artisan Technology Group is an independent supplier of quality pre-owned equipment

Gold-standard solutions

Extend the life of your critical industrial, commercial, and military systems with our superior service and support.

We buy equipment

Planning to upgrade your current equipment? Have surplus equipment taking up shelf space? We'll give it a new home.

Learn more!

Visit us at **artisantg.com** for more info on price quotes, drivers, technical specifications, manuals, and documentation.

Artisan Scientific Corporation dba Artisan Technology Group is not an affiliate, representative, or authorized distributor for any manufacturer listed herein.

We're here to make your life easier. How can we help you today?

(217) 352-9330 | sales@artisantg.com | artisantg.com

ARTISAN® TECHNOLOGY GROUP