GPIB

NI-488.2™ Function Reference Manual for MacOS

July 1997 Edition
Part Number 320898B-01

Internet Support

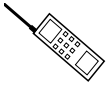
support@natinst.com
E-mail: info@natinst.com
FTP Site: ftp.natinst.com
Web Address: http://www.natinst.com

Bulletin Board Support

BBS United States: (512) 794-5422
BBS United Kingdom: 01635 551422
BBS France: 01 48 65 15 59

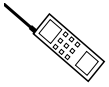
Fax-on-Demand Support

(512) 418-1111



Telephone Support (U.S.)

Tel: (512) 795-8248
Fax: (512) 794-5678



International Offices

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

National Instruments Corporate Headquarters

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

Important Information

Warranty

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

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

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

EXCEPT AS SPECIFIED HEREIN, NATIONAL INSTRUMENTS MAKES NO WARRANTIES, EXPRESS OR IMPLIED, AND SPECIFICALLY DISCLAIMS ANY WARRANTY OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. CUSTOMER'S RIGHT TO RECOVER DAMAGES CAUSED BY FAULT OR NEGLIGENCE ON THE PART OF NATIONAL INSTRUMENTS SHALL BE LIMITED TO THE AMOUNT THERETOFORE PAID BY THE CUSTOMER. NATIONAL INSTRUMENTS WILL NOT BE LIABLE FOR DAMAGES RESULTING FROM LOSS OF DATA, PROFITS, USE OF PRODUCTS, OR INCIDENTAL OR CONSEQUENTIAL DAMAGES, EVEN IF ADVISED OF THE POSSIBILITY THEREOF. This limitation of the liability of National Instruments will apply regardless of the form of action, whether in contract or tort, including negligence. Any action against National Instruments must be brought within one year after the cause of action accrues. National Instruments shall not be liable for any delay in performance due to causes beyond its reasonable control. The warranty provided herein does not cover damages, defects, malfunctions, or service failures caused by owner's failure to follow the National Instruments installation, operation, or maintenance instructions; owner's modification of the product; owner's abuse, misuse, or negligent acts; and power failure or surges, fire, flood, accident, actions of third parties, or other events outside reasonable control.

Copyright

Trademarks

NI-488® and NI-488.2™ are trademarks of National Instruments Corporation.

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

WARNING REGARDING MEDICAL AND CLINICAL USE OF NATIONAL INSTRUMENTS PRODUCTS

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

Table of Contents

About This Manual

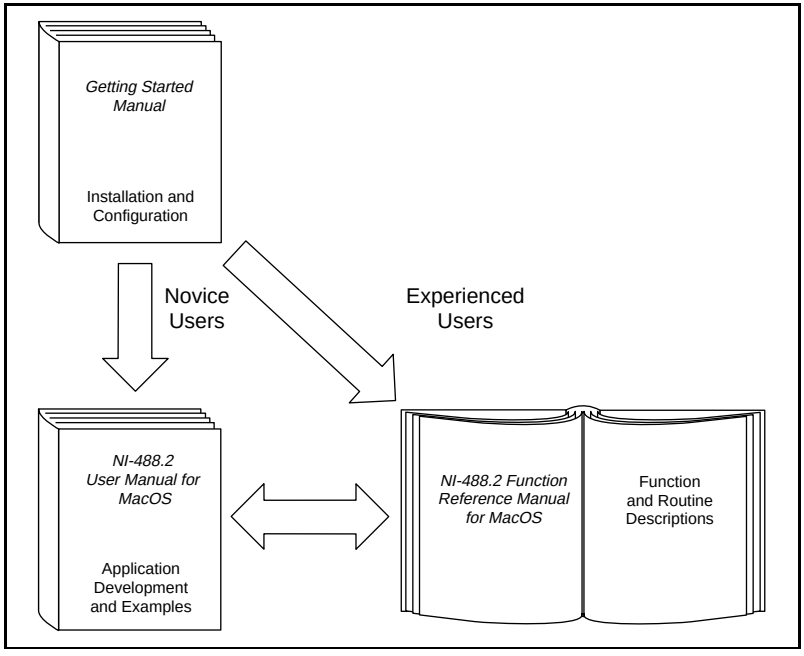
Chapter 1
NI-488 Functions

Chapter 2
NI-488.2 Routines

About This Manual

This manual describes the features and functions of the NI-488.2 software for MacOS. This manual assumes that you are already familiar with the Macintosh operating system.

How to Use This Manual Set



Use the getting started manual that came with your kit to install and configure your GPIB hardware and NI-488.2 software.

Use the *NI-488.2 User Manual for MacOS* to learn the basics of GPIB and how to develop an application program. The user manual also contains debugging information and detailed examples.

Use the *NI-488.2 Function Reference Manual for MacOS* for specific NI-488 function and NI-488.2 routine information, such as format, parameters, and possible errors.

Organization of This Manual

This manual is organized as follows:

- Chapter 1, *NI-488 Functions*, lists the available NI-488 functions and describes the purpose, format, input and output parameters, and possible errors for each function.
- Chapter 2, *NI-488.2 Routines*, lists the available NI-488.2 routines and describes the purpose, format, input and output parameters, and possible errors for each routine.
- Appendix A, *Multiline Interface Messages*, 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.
- Appendix B, *Status Word Conditions*, gives a detailed description of the conditions reported in the status word, `ibsta`.
- Appendix C, *Error Codes and Solutions*, lists a description of each error, some conditions under which it might occur, and possible solutions.
- Appendix D, *Customer Communication*, contains forms you can use to request help from National Instruments or to comment on our products and manuals.
- The *Glossary* contains an alphabetical list and description of terms used in this manual, including abbreviations, acronyms, metric prefixes, mnemonics, and symbols.
- The *Index* contains an alphabetical list of key terms and topics in this manual, including the page where you can find each one.

Conventions Used in This Manual

The following conventions are used in this manual:

<>	Angle brackets enclose the name of a key on the keyboard (for example, <option>). Angle brackets containing numbers separated by an ellipsis represent a range of values associated with a bit or signal name (for example, DBIO<3..0>).
☞	This icon to the left of bold italicized text denotes a note, which alerts you to important information.
bold italic	Bold italic text denotes a note, caution, or warning.
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 488.2-1992, respectively, which define the GPIB.
italic	Italic text denotes emphasis, a cross reference, or an introduction to a key concept.
`monospace`	Text in this font denotes text or characters that should literally enter from the keyboard, sections of code, programming examples, and syntax examples. This font is also used for the proper names of disk drives, paths, directories, programs, subprograms, subroutines, device names, functions, operations, variables, filenames and extensions, and for statements and comments taken from programs. 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 488.2-1992, *IEEE Standard Codes, Formats, Protocols, and Common Commands*
- *Inside Macintosh*, Addison-Wesley Publishing Company, Reading, MA, 1994
- *Macintosh Programmer's Workshop, Version 3.3*, Apple Computer, Inc., Cupertino, CA, 1993

- *Metrowerks CodeWarrior User's Guide*, Metrowerks, Inc., Mooers, NY
- *FutureBASIC*, STAZ Software, Inc., Diamondhead, MS, 1996
- *THINK C User's Manual*, Symantec Corp., Bedford, MA

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

Chapter 1

NI-488 Functions

This chapter lists the available NI-488 functions and describes the purpose, format, input and output parameters, and possible errors for each function.

While using the functions, you might find it helpful to refer to Chapter 2, *Developing Your Application,* and Chapter 5, *GPIB Programming Techniques,* in the *NI-488.2 User Manual for MacOS.*

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.

Format

The format is given for each of the following languages supported by the NI-488.2 software:

- MPW C version 3.0 or higher, THINK C version 4.0 or higher, and Metrowerks CodeWarrior 1.1 or higher
- FutureBASIC II

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 NI-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.

List of NI-488 Functions

The following tables contain alphabetical lists of each NI-488 function along with its purpose. Table 1-1 lists the device-level functions. Table 1-2 lists the board-level functions.

Table 1-1. List of NI-488 Device-Level Functions

Function	Purpose
ibask	Return information about software configuration parameters
ibbna	Change the access board of a device
ibclr	Clear a specific device
ibconfig	Change the software configuration parameters
ibdev	Open and initialize a device
ibeos	Configure the end-of-string (EOS) termination mode or character
ibeot	Enable or disable the automatic assertion of the GPIB EOI line at the end of write I/O operations
ibln	Check for the presence of a device on the bus
ibloc	Go to local
iblock	Lock access to a GPIB-ENET board or device
ibonl	Place the device online or offline
ibpad	Change the primary address
ibpct	Pass control to another GPIB device with Controller capability
ibppc	Parallel poll configure
ibrd	Read data from a device into a user buffer
ibrda	Read data asynchronously from a device into a user buffer
ibrdf	Read data from a device into a file
ibrpp	Conduct a parallel poll

Table 1-1. List of NI-488 Device-Level Functions (Continued)

Function	Purpose
ibrsp	Conduct a serial poll
ibsad	Change or disable the secondary address
ibstop	Abort asynchronous I/O operation
ibtmo	Change or disable the I/O timeout period
ibtrg	Trigger selected device
ibunlock	Unlock access to a GPIB-ENET board or device
ibwait	Wait for GPIB events
ibwrt	Write data to a device from a user buffer
ibwrta	Write data asynchronously to a device from a user buffer
ibwrtf	Write data to a device from a file

Table 1-2. List of NI-488 Board-Level Functions

Function	Purpose
ibask	Return information about software configuration parameters
ibcac	Become Active Controller
ibcmd	Send GPIB commands
ibcmda	Send GPIB commands asynchronously
ibconfig	Change the software configuration parameters
ibdma	Enable or disable DMA
ibeos	Configure the end-of-string (EOS) termination mode or character
ibeot	Enable or disable the automatic assertion of the GPIB EOI line at the end of write I/O operations
ibfind	Open and initialize a GPIB board
ibgts	Go from Active Controller to Standby
ibist	Set or clear the board individual status bit for parallel polls
iblines	Return the status of the eight GPIB control lines
ibln	Check for the presence of a device on the bus
ibloc	Go to local
iblock	Lock access to a GPIB-ENET board or device
ibonl	Place the interface board online or offline
ibpad	Change the primary address
ibppc	Parallel poll configure
ibrd	Read data from a device into a user buffer
ibrda	Read data asynchronously from a device into a user buffer
ibrdf	Read data from a device into a file

Table 1-2. List of NI-488 Board-Level Functions (Continued)

Function	Purpose
ibrpp	Conduct a parallel poll
ibrsc	Request or release system control
ibrsv	Request service and change the serial poll status byte
ibsad	Change or disable the secondary address
ibsic	Assert interface clear
ibsre	Set or clear the Remote Enable (REN) line
ibsrq	Request an SRQ “interrupt routine”
ibstop	Abort asynchronous I/O operation
ibtmo	Change or disable the I/O timeout period
ibunlock	Unlock access to a GPIB-ENET board or device
ibwait	Wait for GPIB events
ibwrt	Write data to a device from a user buffer
ibwrta	Write data asynchronously to a device from a user buffer
ibwrtf	Write data to a device from a file

IBASK

Board Level/Device Level

Purpose

Return information about software configuration parameters.

Format

C

short ibask (short ud, short option, short *value)

FutureBASIC

```
FN ibask%(ud%,option%,value&)
```

Input

ud	Board or device unit descriptor
option	Selects the configuration item whose value is being returned

Output

value	Current value of the selected configuration item
Function Return	The value of ibsta

Description

ibask returns the current value of various configuration parameters for the specified board or device. The current value of the selected configuration item is returned in the integer specified by value. Table 1-3 and Table 1-4 list the valid configuration parameter options for ibask.

Possible Errors

EARG	option is not a valid configuration parameter. See the ibask options listed in Table 1-3 and Table 1-4.
ECAP	option is not supported by the driver in its current configuration.
EDVR	Either ud is invalid or the NI-488.2 driver is not installed.

IBASK

(Continued)

Table 1-3 lists the options you can use with `ibask` when `ud` is a board descriptor or a board index.

Table 1-3. ibask Board Configuration Parameter Options

Options (Constants)	Options (Values)	Returned Information
IbaAUTOPOLL	0x0007	zero = Automatic serial polling is disabled. non-zero = Automatic serial polling is enabled. Refer to the NI-488.2 user manual for more information about automatic serial polling.
IbaCICPROT	0x0008	zero = The CIC protocol is disabled. non-zero = The CIC protocol is enabled. Refer to the NI-488.2 user manual for more information about device-level calls and bus management.
IbaDMA	0x0012	zero = The board will not use DMA for GPIB transfers. non-zero = The board will use DMA for GPIB transfers. See `ibdma`.
IbaEndBitIsNormal	0x001A	zero = The END bit of `ibsta` is set only when EOI or EOI plus the EOS character is received. If the EOS character is received without EOI, the END bit is not set. non-zero = The END bit is set whenever EOI, EOS, or EOI plus EOS is received.
IbaEOSchar	0x000F	The current EOS character of the board. See `ibeos`.

IBASK

(Continued)

Table 1-3. ibask Board Configuration Parameter Options (Continued)

Options (Constants)	Options (Values)	Returned Information
IbaEOScmp	0x000E	zero = A 7-bit compare is used for all EOS comparisons. non-zero = An 8-bit compare is used for all EOS comparisons. See `ibeos`.
IbaEOSrd	0x000C	zero = The EOS character is ignored during read operations. non-zero = Read operation is terminated by the EOS character. See `ibeos`.
IbaEOSwrt	0x000D	zero = The EOI line is not asserted when the EOS character is sent during a write operation. non-zero = The EOI line is asserted when the EOS character is sent during a write operation. See `ibeos`.
IbaEOT	0x0004	zero = The GPIB EOI line is not asserted at the end of a write operation. non-zero = EOI is asserted at the end of a write. See `ibeot`.
IbaHSCableLength	0x001F	0 = High-speed data transfer (HS488) is disabled. 1 to 15 = High-speed data transfer (HS488) is enabled. The number returned represents the number of meters of GPIB cable in your system. See the NI-488.2 user manual for information about high-speed data transfers (HS488).
IbaIst	0x0020	The individual status (`ist`) bit of the board.

IBASK

(Continued)

Table 1-3. ibask Board Configuration Parameter Options (Continued)

Options (Constants)	Options (Values)	Returned Information
IbaPAD	0x0001	The current primary address of the board. See `ibpad`.
IbaPP2	0x0010	zero = The board is in PP1 mode non-zero = The board is in PP2 mode Refer to the NI-488.2 user manual for more information about parallel polls.
IbaPPC	0x0005	The current parallel poll configuration information of the board. See `ibppc`.
IbaReadAdjust	0x0013	0 = Read operations do not have pairs of bytes swapped. 1 = Read operations have each pair of bytes swapped.
IbaRsv	0x0021	The current serial poll status byte of the board.
IbaSAD	0x0002	The current secondary address of the board. See `ibsad`.
IbaSC	0x000A	zero = The board is not the GPIB System Controller. non-zero = The board is the System Controller. See `ibrsc`.
IbaSRE	0x000B	zero = The board does not automatically assert the GPIB REN line when it becomes the System Controller. non-zero = The board automatically asserts REN when it becomes the System Controller. See `ibrsc` and `ibsre`.

IBASK

(Continued)

Table 1-3. ibask Board Configuration Parameter Options (Continued)

Options (Constants)	Options (Values)	Returned Information
IbaTIMING	0x0011	The current bus timing of the board. 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.)
IbaTMO	0x0003	The current I/O timeout of the board. See `ibtmo`.
IbaWriteAdjust	0x0014	0 = Write operations do not have pairs of bytes swapped. 1 = Write operations have each pair of bytes swapped.

IBASK

(Continued)

Table 1-4 lists the options you can use with `ibask` when `ud` is a device descriptor or a device index.

Table 1-4. ibask Device Configuration Parameter Options

Options (Constants)	Options (Values)	Returned Information
IbaBNA	0x0200	The index of the GPIB access board used by the given device descriptor.
IbaEOSchar	0x000F	The current EOS character of the device. See `ibeos`.
IbaEOScmp	0x000E	zero = A 7-bit compare is used for all EOS comparisons. non-zero = An 8-bit compare is used for all EOS comparisons. See `ibeos`.
IbaEOSrd	0x000C	zero = The EOS character is ignored during read operations. non-zero = Read operation is terminated by the EOS character. See `ibeos`.
IbaEOSwrt	0x000D	zero = The EOI line is not asserted when the EOS character is sent during a write operation. non-zero = The EOI line is asserted when the EOS character is sent during a write operation. See `ibeos`.
IbaEOT	0x0004	zero = The GPIB EOI line is not asserted at the end of a write operation. non-zero = EOI is asserted at the end of a write operation. See `ibeot`.

IBASK

(Continued)

Table 1-4. ibask Device Configuration Parameter Options (Continued)

Options (Constants)	Options (Values)	Returned Information
IbaPAD	0x0001	The current primary address of the device. See `ibpad`.
IbaReadAdjust	0x0013	0 = Read operations do not have pairs of bytes swapped. 1 = Read operations have each pair of bytes swapped.
IbaREADDR	0x0006	zero = No unnecessary addressing is performed between device-level read and write operations. non-zero = Addressing is always performed before a device-level read or write operation.
IbaSAD	0x0002	The current secondary address of the device. See `ibsad`.
IbaTMO	0x0003	The current I/O timeout of the device. See `ibtmo`.
IbaWriteAdjust	0x0014	0 = Write operations do not have pairs of bytes swapped. 1 = Write operations have each pair of bytes swapped.

IBBNA

Device Level

Purpose

Change the access board of a device.

Format

C

short ibbna (short ud, char bname [])

FutureBASIC

```
FN ibbna%(ud%,bname&)
```

Input

ud	A device unit descriptor
bname	An access board name, for example, gpib0

Output

Function Return	The value of ibsta

Description

ibbna assigns the device described by ud to the access board described by bname. All subsequent bus activity with device ud occurs through the access board bname. If the call succeeds, iberr contains the previous access board index.

Possible Errors

EARG	Either ud does not refer to a device or bname does not refer to a valid board name.
ECIC	The access board is not CIC. See the *Device-Level Calls and Bus Management* section in Chapter 5, *GPIB Programming Techniques*, of the *NI-488.2 User Manual for MacOS.*
EDVR	Either ud is invalid or the NI-488.2 driver is not installed.
ENEB	The access board is not installed or is not properly configured.
EOIP	Asynchronous I/O is in progress.

IBCAC

Board Level

Purpose

Become Active Controller.

Format

C

short ibcac (short ud, short v)

FutureBASIC

```
FN ibcac%(ud%,v%)
```

Input

ud	A board unit descriptor
v	Determines if control is to be taken asynchronously or synchronously

Output

Function Return	The value of ibsta

Description

Using ibcac, the designated GPIB board attempts to become the Active Controller by asserting ATN. If v is zero, the GPIB board takes control asynchronously. If v is non-zero, the GPIB board takes control synchronously.

To take control synchronously, the GPIB board attempts to assert the ATN signal without corrupting transferred data. If this is not possible, the board takes control asynchronously.

To take control asynchronously, the GPIB board asserts ATN immediately without regard for any data transfer currently in progress.

Most applications do not need to use ibcac. Functions that require ATN to be asserted, such as ibcmd, do so automatically.

IBCAC

(Continued)

Possible Errors

EARG	ud is valid but does not refer to an interface board.
ECIC	The interface board is not Controller-In-Charge.
EDVR	Either ud is invalid or the NI-488.2 driver is not installed.
ENEB	The interface board is not installed or is not properly configured.
EOIP	Asynchronous I/O is in progress.

IBCLR

Device Level

Purpose

Clear a specific device.

Format

C

short `ibclr` (short `ud`)

FutureBASIC

```
FN ibclr%(ud%)
```

Input

`ud`	A device unit descriptor

Output

Function Return	The value of `ibsta`

Description

`ibclr` sends the GPIB Selected Device Clear (SDC) message to the device described by `ud`.

Possible Errors

EARG	`ud` is a valid descriptor but does not refer to a device.
EBUS	There are no devices connected to the GPIB.
ECIC	The `access` board is not CIC. See the *Device-Level Calls and Bus Management* section in Chapter 5, *GPIB Programming Techniques*, of the *NI-488.2 User Manual for MacOS.*
EDVR	Either `ud` is invalid or the NI-488.2 driver is not installed.
ENEB	The interface board is not installed or is not properly configured.
EOIP	Asynchronous I/O is in progress.

IBCMD

Board Level

Purpose

Send GPIB commands.

Format

C

short ibcmd (short ud, char *cmd, long cnt)

FutureBASIC

```
FN ibcmd%(ud%,cmd&,cnt&)
```

Input

ud	A board unit descriptor
cmd	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 cmd over the GPIB as command bytes (interface messages). The number of command bytes transferred is returned in the global variable ibcnt. 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.

IBCMD

(Continued)

Possible Errors

EABO	The timeout period expired before all of the command bytes were sent.
EARG	ud is valid but does not refer to an interface board.
ECIC	The interface board is not Controller-In-Charge.
EDVR	Either ud is invalid or the NI-488.2 driver is not installed.
ENEB	The interface board is not installed or is not properly configured.
ENOL	No Listeners are on the GPIB.
EOIP	Asynchronous I/O is in progress.

IBCMDA

Board Level

Purpose

Send GPIB commands asynchronously.

Format

C

short ibcmda (short ud, char *cmd, long cnt)

FutureBASIC

```
FN ibcmda%(ud%,cmd&,cnt&)
```

Input

ud	A board unit descriptor
cmd	Buffer of command bytes to send
cnt	Number of command bytes to send

Output

Function Return	The value of ibsta

Description

ibcmda sends cnt bytes from cmd over the GPIB as command bytes (interface messages). The number of command bytes transferred is returned in the global variable ibcnt. 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.

The asynchronous I/O calls (ibcmda, ibrda, ibwrta) are designed so that applications can perform other non-GPIB operations while the I/O is in progress. Once the asynchronous I/O has begun, further GPIB calls are strictly limited. Any calls that would interfere with the I/O in progress are not allowed, the driver returns EOIP in this case.

IBCMDA

(Continued)

Once the I/O is complete, the application must *resynchronize* with the NI-488.2 driver. Resynchronization is accomplished by using one of the following three functions:

- `ibwait` If the returned `ibsta` mask has the CMPL bit set, the driver and application are resynchronized.
- `ibstop` The I/O is canceled; the driver and application are resynchronized.
- `ibonl` The I/O is canceled and the interface is reset; the driver and application are resynchronized.

Possible Errors

EARG	`ud` is valid but does not refer to an interface board.
ECIC	The interface board is not Controller-In-Charge.
EDVR	Either `ud` is invalid or the NI-488.2 driver is not installed.
ENEB	The interface board is not installed or is not properly configured.
ENOL	No Listeners are on the GPIB.
EOIP	Asynchronous I/O is in progress.

IBCONFIG

Board Level/Device Level

Purpose

Change the software configuration parameters.

Format

C

short ibconfig (short ud, unsigned short option, unsigned short value)

FutureBASIC

```
FN ibconfig%(ud%,option%,value%)
```

Input

ud	Board or device unit descriptor
option	A parameter that selects the software 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 1-5 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 Table 1-5.
ECAP	The driver is not able to make the requested change.
EDVR	Either ud is invalid or the NI-488.2 driver is not installed.
EOIP	Asynchronous I/O is in progress.

IBCONFIG

(Continued)

Table 1-5 lists the options you can use with `ibconfig` when `ud` is a board descriptor or a board index.

Table 1-5. ibconfig Board Configuration Parameter Options

Options (Constants)	Options (Values)	Legal Values
IbcAUTOPOLL	0x0007	zero = Disable automatic serial polling. non-zero = Enable automatic serial polling. Default determined by `NI-488 Config`. Refer to the *NI-488.2 User Manual for MacOS* for more information about automatic serial polling.
IbcCICPROT	0x0008	zero = Disable the CIC protocol. non-zero = Enable the CIC protocol. Default determined by `NI-488 Config`. Refer to the *NI-488.2 User Manual for MacOS* for more information about the CIC protocol.
IbcDMA	0x0012	Identical to `ibdma`. Default determined by `NI-488 Config.`
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.
IbcEOSchar	0x000F	Any 8-bit value. This byte becomes the new EOS character. Default determined by `NI-488 Config`.

IBCONFIG

(Continued)

Table 1-5. ibconfig Board Configuration Parameter Options (Continued)

Options (Constants)	Options (Values)	Legal Values
IbcEOScmp	0x000E	zero = Use 7 bits for the EOS character comparison. non-zero = Use 8 bits for the EOS character comparison. Default determined by `NI-488 Config`.
IbcEOSrd	0x000C	zero = Ignore EOS character during read operations. non-zero = Terminate reads when the EOS character is read match occurs. Default determined by `NI-488 Config`.
IbcEOSwrt	0x000D	zero = Do not assert EOI with the EOS character during write operations. non-zero = Assert EOI with the EOS character during writes operations. Default determined by `NI-488 Config`.
IbcEOT	0x0004	Changes the data termination mode for write operations. Identical to `ibeot`. Default determined by `NI-488 Config`.
IbcHSCableLength	0x001F	0 = High-speed data transfer (HS488) is disabled. 1 to 15 = The number of meters of GPIB cable in your system. The NI-488.2 software uses this information to select the appropriate high-speed data transfer (HS488) mode. Default determined by `NI-488 Config`. See the *NI-488.2 User Manual for MacOS* for information about high-speed data transfers (HS488).

IBCONFIG

(Continued)

Table 1-5. ibconfig Board Configuration Parameter Options (Continued)

Options (Constants)	Options (Values)	Legal Values
IbcPAD	0x0001	Changes the primary address of the board. Identical to `ibpad`. Default determined by `NI-488 Config`.
IbcPP2	0x0010	zero = PP1 mode-remote parallel poll configuration. non-zero = PP2 mode-local parallel poll configuration. Default: zero. Refer to the *NI-488.2 User Manual for MacOS* for more information about parallel polling.
IbcPPC	0x0005	Configures the board for parallel polls. Identical to board-level `ibppc`. 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 `ibtmo` values. Default: zero.
IbcReadAdjust	0x0013	0 = No byte swapping. 0 = No byte swapping. 1 = Swap pairs of bytes during a read. Default: zero.
IbcSAD	0x0002	Changes the secondary address of the board. Identical to `ibsad`. Default determined by `NI-488 Config`.

IBCONFIG

(Continued)

Table 1-5. ibconfig Board Configuration Parameter Options (Continued)

Options (Constants)	Options (Values)	Legal Values
IbcSC	0x000A	Request or release system control. Identical to `ibrsc`. Default determined by `NI-488 Config`.
IbcSendLLO	0x0017	zero = Do not send LLO when putting a device online—`ibfind` or `ibdev`. non-zero = Send LLO when putting a device online—`ibfind` or `ibdev`. Default: zero.
IbcSRE	0x000B	Assert the Remote Enable (REN) line. Identical to `ibsre`. 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). Default determined by `NI-488 Config`. The T1 delay is the GPIB source handshake timing.
IbcTMO	0x0003	Changes the I/O timeout limit of the board. Identical to `ibtmo`. Default determined by `NI-488 Config`.
IbcWriteAdjust	0x0014	0 = No byte swapping. 1 = Swap pairs of bytes during a write. Default: zero.

IBCONFIG

(Continued)

Table 1-6 lists the options you can use with `ibconfig` when `ud` is a device descriptor or a device index.

Table 1-6. ibconfig Device Configuration Parameter Options

Options (Constants)	Options (Values)	Legal Values
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.
IbcEOSchar	0x000F	Any 8-bit value. This byte becomes the new EOS character. Default determined by `NI-488 Config`.
IbcEOScmp	0x000E	zero = Use seven bits for the EOS character comparison. non-zero = Use 8 bits for the EOS character comparison. Default determined by `NI-488 Config`.
IbcEOSrd	0x000C	non-zero = Terminate reads when the EOS character is read. Default determined by `NI-488 Config`.
IbcEOSwrt	0x000D	zero = Do not send EOI with the EOS character during write operations. non-zero = Send EOI with the EOS character during writes. Default determined by `NI-488 Config`.
IbcEOT	0x0004	Changes the data termination method for writes. Identical to `ibeot`. Default determined by `NI-488 Config`.

IBCONFIG

(Continued)

Table 1-6. ibconfig Device Configuration Parameter Options (Continued)

Options (Constants)	Options (Values)	Legal Values
IbcPAD	0x0001	Changes the primary address of the device. Identical to `ibpad`. Default determined by `NI-488 Config`.
IbcReadAdjust	0x0013	0 = No byte swapping. 1 = Swap pairs of bytes during a read. Default: zero.
IbcREADDR	0x0006	zero = No unnecessary readdressing is performed between device-level reads and writes. non-zero = Addressing is always performed before a device-level read or write. Default determined by `NI-488 Config`.
IbcSAD	0x0002	Changes the secondary address of the device. Identical to `ibsad`. Default determined by `NI-488 Config`.
IbcTMO	0x0003	Changes the device I/O timeout limit. Identical to `ibtmo`. Default determined by `NI-488 Config`.
IbcWriteAdjust	0x0014	0 = No byte swapping. 1 = Swap pairs of bytes during a write. Default: zero.

IBDEV

Device Level

Purpose

Open and initialize a device descriptor.

Format

C

```
ud = short ibdev (short boardindex, short pad, short sad, short tmo, short eot,
                  short eos)
```

FutureBASIC

```
ud% = FN ibdev%(boardindex%,pad%,sad%,tmo%,eot%,eos%)
```

Input

`boardindex`	Index of the access board for the device
`pad`	The primary GPIB address of the device
`sad`	The secondary GPIB address of the device
`tmo`	The I/O timeout value
`eot`	EOI mode of the device
`eos`	EOS character and modes

Output

`ud`	Returned device descriptor

Description

`ibdev` acquires a device descriptor to use in subsequent device-level NI-488 functions. It opens and initializes a device descriptor and configures it according to the input parameters.

For more details on the meaning and effect of each input parameter, see the corresponding NI-488 functions for `ibbna`, `ibpad`, `ibsad`, `ibtmo`, `ibeot`, and `ibeos`.

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

IBDEV

(Continued)

`ibdev` acquires and initializes a device descriptor from the set of user-configurable devices (for example, `dev1`, `dev2`, and so on through `dev32`). As a result, it is necessary for an application to use `ibdev` only after all calls to `ibfind` for user-configurable devices have been completed. This is the only way to ensure that `ibdev` and `ibfind` do not both return the same device descriptor.

Possible Errors

EARG	`pad`, `sad`, `tmo`, `eot`, or `eos` is invalid. See the corresponding NI-488 function.
EDVR	Either no device descriptors are available or `boardindex` refers to a GPIB board that is not installed.
ENEB	The interface board is not installed or is not properly configured.

IBDMA

Board Level

Purpose

Enable or disable DMA.

Format

C

short `ibdma` (short `ud`, short `v`)

FutureBASIC

```
FN ibdma%(ud%,v%)
```

Input

`ud`	A board descriptor
`v`	Enable or disable the use of DMA

Output

Function Return	The value of `ibsta`

Description

`ibdma` enables or disables DMA transfers for the board described by `ud`. If `v` is zero then DMA is not used for GPIB I/O transfers. If `v` is non-zero, then DMA is used for GPIB I/O transfers.

Possible Errors

EARG	`ud` is valid but does not refer to an interface board.
EDMA	The interface board is not capable of using DMA.
EDVR	Either `ud` is invalid or the NI-488.2 driver is not installed.
ENEB	The interface board is not installed or is not properly configured.
EOIP	Asynchronous I/O is in progress.

IBEOS

Board Level/Device Level

Purpose

Configure the end-of-string (EOS) termination mode or character.

Format

C

short ibeos (short ud, short v)

FutureBASIC

```
FN ibeos%(ud%,v%)
```

Input

ud	A board or device descriptor
v	EOS mode and character information

Output

Function Return	The value of ibsta

Description

ibeos configures the EOS termination mode or EOS character used by the board or device described by ud. The parameter v describes the new end-of-string (EOS) configuration to use. If v is zero, then the EOS configuration is disabled. Otherwise, the low byte is the EOS character and the upper byte contains flags which define the EOS mode. Table 1-7 describes the different EOS configurations and the corresponding values of v. If no error occurs during the call, then the value of the previous EOS setting is returned in iberr.

IBEOS

(Continued)

Table 1-7. EOS Configurations

Bit	Configuration	Value of v	
		High Byte	Low Byte
A	Terminate read when EOS is detected.	00000100	EOS character
B	Set EOI with EOS on write function.	00001000	EOS character
C	Compare all 8 bits of EOS byte rather than low 7 bits (all read and write functions).	00010000	EOS character

Configuration bits A and C determine how to terminate read I/O operations. If bit A is set and bit C is clear, then a read ends when a byte that matches the low seven bits of the EOS character is received. If bits A and C are both set, then a read ends when a byte that matches all eight bits of the EOS character is received.

Configuration bits B and C determine when a write I/O operation asserts the GPIB EOI line. If bit B is set and bit C is clear, then EOI is asserted when the written character matches the low seven bits of the EOS character. If bits B and C are both set, then EOI is asserted when the written character matches all eight bits of the EOS character.

☞ **Note:** ***Defining an EOS byte does not cause the driver to automatically send that byte at the end of write I/O operations. In your application the EOS byte must be placed at the end of the data strings that it defines.***

For more information on the termination of I/O operations refer to the *NI-488.2 User Manual for MacOS.*

Possible Errors

EARG	The high byte of v contains invalid bits.
EDVR	Either ud is invalid or the NI-488.2 driver is not installed.
ENEB	The interface board is not installed or is not properly configured.
EOIP	Asynchronous I/O is in progress.

IBEOS

(Continued)

Examples

```
ibeos (ud, 0x140A);  /* Configure the software to end reads on
                        newline character (hex 0A) for the unit
                        descriptor, ud */
ibeos (ud, 0x180A);  /* Configure the software to assert the GPIB
                        EOI line whenever the newline character
                        (hex 0A)is written out by the unit
                        descriptor, ud */
```

IBEOT

Board Level/Device Level

Purpose

Enable or disable the automatic assertion of the GPIB EOI line at the end of write I/O operations.

Format

C

short `ibeot` (short `ud`, short `v`)

FutureBASIC

```
FN ibeot%(ud%,v%)
```

Input

`ud`	A board or device descriptor
`v`	Enables or disables the end of transmission assertion of EOI

Output

Function Return	The value of `ibsta`

Description

`ibeot` enables or disables the assertion of the EOI line at the end of write I/O operations, such as `ibwrt`, for the board or device described by `ud`. If `v` is non-zero, then EOI is asserted when the last byte of a GPIB write is sent. If `v` is zero, then nothing occurs when the last byte is sent. If no error occurs during the call, then the previous value of EOT is returned in `iberr`.

For more information on the termination of I/O operations refer to the *NI-488.2 User Manual for MacOS.*

Possible Errors

EDVR	Either `ud` is invalid or the NI-488.2 driver is not installed.
ENEB	The interface board is not installed or is not properly configured.
EOIP	Asynchronous I/O is in progress.

IBFIND

Board Level/Device Level

Purpose

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

Format

C

```
ud = short ibfind (char udname [])
```

FutureBASIC

```
ud% = FN ibfind%(udname$)
```

Input

udname	A user-configured device or board name

Output

ud	Returned device descriptor

Description

ibfind is used to acquire a descriptor for a board or user-configured device; this board or device descriptor can be used in subsequent NI-488 functions.

ibfind performs the equivalent of an ibonl 1 to initialize the board or device descriptor. The unit 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.

☞ **Note:** ***Using*** ibfind ***to obtain device descriptors is useful only for compatibility with existing applications. New applications should use*** ibdev ***instead of*** ibfind***.*** ibdev ***is more flexible, easier to use, and frees the application from unnecessary device name requirements.***

IBFIND

(Continued)

Possible Errors

EBUS	Device level: There are no devices connected to the GPIB.
ECIC	Device level: The access board is not CIC. See the *Device-Level Calls and Bus Management* section in Chapter 5, *GPIB Programming Techniques*, of the *NI-488.2 User Manual for MacOS.*
EDVR	Either `udname` is not recognized as a board or device name or the NI-488.2 driver is not installed.
ENEB	The interface board is not installed or is not properly configured.

IBGTS

Board Level

Purpose

Go from Active Controller to Standby.

Format

C

short `ibgts` (short `ud`, short `v`)

FutureBASIC

```
FN ibgts%(ud%,v%)
```

Input

`ud`	Board descriptor
`v`	Determines whether to perform acceptor handshaking

Output

Function Return	The value of `ibsta`

Description

`ibgts` causes the board `ud` to go to Standby Controller and the GPIB ATN line to be unasserted. If `v` is non-zero, acceptor handshaking or shadow handshaking is performed until END occurs or until ATN is reasserted by a subsequent `ibcac` call. With this option, the GPIB board can participate in data handshake as an acceptor without actually reading data. If END is detected, the interface board enters a Not Ready For Data (NRFD) handshake holdoff state which results in hold off of subsequent GPIB transfers. If `v` is 0, no acceptor handshaking or holdoff is performed.

Before performing an `ibgts` with shadow handshake, call the `ibeos` function to establish proper EOS modes.

For more information about handshaking, refer to the ANSI/IEEE Standard 488.1-1987.

IBGTS

(Continued)

Possible Errors

EADR	v is non-zero, and either ATN is low or the interface board is a Talker or a Listener.
ARG	ud is valid but does not refer to an interface board.
ECIC	The interface board is not Controller-In-Charge.
EDVR	Either ud is invalid or the NI-488.2 driver is not installed.
ENEB	The interface board is not installed or is not properly configured.
EOIP	Asynchronous I/O is in progress.

IBIST

Board Level

Purpose

Set or clear the board individual status bit for parallel polls.

Format

C

short ibist (short ud, short v)

FutureBASIC

```
FN ibist%(ud%,v%)
```

Input

ud	Board descriptor
v	Indicates whether to set or clear the ist bit

Output

Function Return	The value of ibsta

Description

ibist sets the interface board ist (individual status) bit according to v. If v is zero, the ist bit is cleared; if v is non-zero, ist bit is set. The previous value of the ist bit is returned in iberr.

For more information on parallel polling, refer to the *NI-488.2 User Manual for MacOS.*

Possible Errors

EARG	ud is valid but does not refer to an interface board.
EDVR	Either ud is invalid or the NI-488.2 driver is not installed.
ENEB	The interface board is not installed or is not properly configured.
EOIP	Asynchronous I/O is in progress.

IBLINES

Board Level

Purpose

Return the status of the eight GPIB control lines.

Format

C

short `iblines` (short `ud`, unsigned short `*gpib_lines`);

FutureBASIC

```
FN iblines%(ud%,gpib_lines&)
```

Input

`ud`	Board or device descriptor

Output

`gpiblines`	Returns GPIB control line state information
Function Return	The value of `ibsta`

Description

`iblines` returns the state of the GPIB control lines in `gpiblines`. The low-order byte (bits 0 through 7) of `clines` contains a mask indicating the capability of the GPIB interface board to sense the status of each GPIB control line. The upper byte (bits 8 through 15) contains the GPIB control line state information. The following is a pattern of each byte.

7	6	5	4	3	2	1	0
EOI	ATN	SRQ	REN	IFC	NRFD	NDAC	DAV

To determine if a GPIB control line is asserted, first check the appropriate bit in the lower byte to determine if the line can be monitored. If the line can be monitored (indicated by a 1 in the appropriate bit position), then check the corresponding bit in the upper byte. If the bit is set (1), the corresponding control line is asserted. If the bit is clear (0), the control line is unasserted.

IBLINES

(Continued)

Possible Errors

EARG	ud is valid but does not refer to an interface board.
EDVR	Either ud is invalid or the NI-488.2 driver is not installed.
ENEB	The interface board is not installed or is not properly configured.

Example

```
short lines;
iblines (ud, &lines);
if (lines & ValidREN) { /* check to see if REN is asserted */
   if (lines & BusREN) {
      printf ("REN is asserted");
   }
}
```

IBLN

Board Level/Device Level

Purpose

Check for the presence of a device on the bus.

Format

C

short ibln (short ud, short pad, short sad, short *listen)

FutureBASIC

```
FN ibln%(ud%,pad%,sad%,listen&)
```

Input

ud	Board or device descriptor
pad	The primary GPIB address of the device
sad	The secondary GPIB address of the device

Output

listen	Indicates whether or not a device is present
Function Return	The value of ibsta

Description

ibln determines whether there is a listening device at the GPIB address designated by the pad and sad parameters. If ud is a board descriptor, then the bus associated with that board is tested for Listeners. If ud is a device descriptor, then ibln uses the access board associated with that device to test for Listeners. If a Listener is detected, a non-zero value is returned in listen. If no Listener is found, zero is returned.

The pad parameter can be any valid primary address (a value between 0 and 30). The sad parameter can be any valid secondary address (a value between 96 to 126), or one of the constants NO_SAD or ALL_SAD. The constant NO_SAD designates that no secondary address is to be tested (only a primary address is tested). The constant ALL_SAD designates that all secondary addresses are to be tested.

IBLN

(Continued)

Possible Errors

ECIC	Device level: The access board is not CIC. See the *Device-Level Calls and Bus Management* section in Chapter 5, *GPIB Programming Techniques*, of the *NI-488.2 User Manual for MacOS.*
EDVR	Either ud is invalid or the NI-488.2 driver is not installed.
ENEB	The interface board is not installed or is not properly configured.
EOIP	Asynchronous I/O is in progress.

IBLOC

Board Level/Device Level

Purpose

Go to Local.

Format

C

short `ibloc` (short `ud`)

FutureBASIC

```
FN ibloc%(ud%)
```

Input

ud	Board or device descriptor

Output

Function Return	The value of `ibsta`

Description

Board Level

If the board is not in a lockout state (LOK does not appear in the status word, `ibsta`), `ibloc` places the board in local mode. Otherwise, the call has no effect.

The `ibloc` function is used to simulate a front panel RTL (Return to Local) switch if the computer is used as an instrument.

Device Level

Unless the REN (Remote Enable) line has been unasserted with the `ibsre` function, all device-level functions automatically place the specified device in remote program mode. `ibloc` is used to move devices temporarily from a remote program mode to a local mode until the next device function is executed on that device.

IBLOC

(Continued)

Possible Errors

EBUS	Device level: No devices are connected to the GPIB.
ECIC	Device level: The access board is not CIC. See the *Device-Level Calls and Bus Management* section in Chapter 5, *GPIB Programming Techniques*, of the *NI-488.2 User Manual for MacOS.*
EDVR	Either ud is invalid or the NI-488.2 driver is not installed.
ENEB	The interface board is not installed or is not properly configured.
EOIP	Asynchronous I/O is in progress.

IBLOCK (GPIB-ENET only)

Board Level/Device Level

Purpose

Lock access to a GPIB-ENET board or device.

Format

C

short `iblock` (short `ud`)

FutureBASIC

```
FN iblock%(ud%)
```

Input

`ud`	A board or device descriptor

Output

Function Return	The value of `ibsta`

Description

`iblock` is used to obtain exclusive access to a GPIB-ENET interface.

Board Level

The `iblock` function blocks other processes from accessing the interface designated by `id` while the lock is in effect. The interface is released via an `ibunlock` function call made with the same board descriptor.

Device Level

The `iblock` function blocks other processes from accessing the device designated by `id` while the lock is in effect. The device lock is released via an `ibunlock` function call made with the same device descriptor.

IBLOCK (GPIB-ENET only)

(Continued)

Recommended Usage

In general, the `iblock` function should be used to gain critical access to a GPIB-ENET board or device when multiple processes might be accessing the same board or device. While locked, the software guarantees that subsequent calls made from the privileged board or device are completed without interruption.

Refer also to *IBUNLOCK (GPIB-ENET only)* later in this chapter.

Possible Errors

EDVR	Either `ud` is invalid or the NI-488.2 driver is not installed.
ELCK	Occurs if the GPIB-ENET board or device being locked is already locked by another process.

IBONL

Board Level/Device Level

Purpose

Place the device or interface board online or offline.

Format

C

short ibonl (short ud, short v)

FutureBASIC

```
FN ibonl%(ud%,v%)
```

Input

ud	Board or device descriptor
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 interface board is taken offline. If v is non-zero, the device or interface board is left operational, or online.

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

Possible Errors

EDVR	Either ud is invalid or the NI-488.2 driver is not installed.
ENEB	The interface board is not installed or is not properly configured.

IBPAD

Board Level/Device Level

Purpose

Change the primary address.

Format

C

short ibpad (short ud, short v)

FutureBASIC

```
FN ibpad%(ud%,v%)
```

Input

ud	Board or device descriptor
v	GPIB primary address

Output

Function Return	The value of ibsta

Description

ibpad sets the primary GPIB address of the board or device to v, an integer ranging from 0 to 30. If no error occurs during the call, then iberr contains the previous GPIB primary address.

Possible Errors

EARG	v is not a valid primary GPIB address; it must be in the range 0 to 30.
EDVR	Either ud is invalid or the NI-488.2 driver is not installed.
ENEB	The interface board is not installed or is not properly configured.
EOIP	Asynchronous I/O is in progress.

IBPCT

Device Level

Purpose

Pass control to another GPIB device with Controller capability.

Format

C

short `ibpct` (short `ud`)

FutureBASIC

```
FN ibpct%(ud%)
```

Input

`ud`	Device descriptor

Output

Function Return	The value of `ibsta`

Description

`ibpct` passes Controller-in-Charge status to the device indicated by `ud`. The access board automatically unasserts the ATN line and goes to Controller Idle State. This function assumes that the device has Controller capability.

Possible Errors

EARG	`ud` is valid but does not refer to a device.
EBUS	No devices are connected to the GPIB.
ECIC	The access board is not CIC. See the *Device-Level Calls and Bus Management* section in Chapter 5, *GPIB Programming Techniques*, of the *NI-488.2 User Manual for MacOS.*
EDVR	Either `ud` is invalid or the NI-488.2 driver is not installed.
ENEB	The interface board is not installed or is not properly configured.
EOIP	Asynchronous I/O is in progress.

IBPPC

Board Level/Device Level

Purpose

Parallel poll configure.

Format

C

short ibppc (short ud, short v)

FutureBASIC

```
FN ibppc%(ud%,v%)
```

Input

ud	Board or device descriptor
v	Parallel poll enable/disable value

Output

Function Return	The value of ibsta

Description

Board Level

If ud is a board descriptor, ibppc performs a local parallel poll configuration using the parallel poll configuration value v. Valid parallel poll messages are 96 to 126 (hex 60 to hex 7E) or zero to send PPD. If no error occurs during the call, then iberr contains the previous value of the local parallel poll configuration.

Device Level

If ud is a device descriptor, ibppc enables or disables the device from responding to parallel polls. The device is addressed and sent the appropriate parallel poll message (PPE) or Disable (PPD). Valid parallel poll messages are 96 to 126 (hex 60 to hex 7E) or zero to send PPD. If no error occurs during the call, then iberr contains the previous value of the device parallel poll configuration.

For more information on parallel polling, refer to the *NI-488.2 User Manual for MacOS.*

IBPPC

(Continued)

Possible Errors

EARG	v does not contain a valid PPE or PPD message.
EBUS	Device level: No devices are connected to the GPIB.
ECAP	Board level: The board is not configured to perform local parallel poll configuration (see `ibconfig`, option `IbcPP2`).
ECIC	Device level: The access board is not CIC. See the *Device-Level Calls and Bus Management* section in Chapter 5, *GPIB Programming Techniques*, of the *NI-488.2 User Manual for MacOS.*
EDVR	Either `ud` is invalid or the NI-488.2 driver is not installed.
ENEB	The interface board is not installed or is not properly configured.
EOIP	Asynchronous I/O is in progress.

IBRD

Board Level/Device Level

Purpose

Read data from a device into a user buffer.

Format

C

short `ibrd` (short `ud`, char `*rdbuffer`, long `cnt`)

FutureBASIC

```
FN ibrd%(ud%,rdbuffer&,cnt&)
```

Input

`ud`	Board or device descriptor
`cnt`	Number of bytes to be read from the GPIB

Output

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

Description

Board Level

If `ud` is a board descriptor, `ibrd` reads up to `cnt` bytes of data from a GPIB device and places it into the buffer specified by `rdbuffer`. 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 `ibcnt`.

IBRD

(Continued)

Device Level

If ud is a device descriptor, ibrd addresses the GPIB, reads up to cnt bytes of data, and places the data into the buffer specified by rdbuffer. 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 ibcnt.

Possible Errors

EABO	Either cnt bytes or END was not received within the timeout period or 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.
	Device level: A conflict exists between the device GPIB address and the GPIB address of the device access board. Use ibpad and ibsad.
EBUS	Device level: No devices are connected to the GPIB.
ECIC	Device level: The access board is not CIC. See the *Device-Level Calls and Bus Management* section in Chapter 5, *GPIB Programming Techniques*, of the *NI-488.2 User Manual for MacOS.*
EDVR	Either ud is invalid or the NI-488.2 driver is not installed.
ENEB	The interface board is not installed or is not properly configured.
EOIP	Asynchronous I/O is in progress.

IBRDA

Board Level/Device Level

Purpose

Read data asynchronously from a device into a user buffer.

Format

C

short ibrda (short ud, char *rdbuffer, long cnt)

FutureBASIC

```
FN ibrda%(ud%,rdbuffer&,cnt&)
```

Input

ud	Board or device descriptor
cnt	Number of bytes to be read from the GPIB

Output

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

Description

Board Level

If ud is a board descriptor, ibrda reads up to cnt bytes of data from a GPIB device and places the data into the buffer specified by rdbuffer. A board-level ibrda 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 board is not the CIC and the CIC sends the Device Clear message on the GPIB. The actual number of bytes transferred is returned in the global variable ibcnt.

Device Level

If ud is a device descriptor, ibrda addresses the GPIB, begins an asynchronous read of up to cnt bytes of data from a GPIB device, and places the data into the memory location specified by rdbuffer. The operation terminates normally when cnt bytes have been

IBRDA

(Continued)

received or END is received. The operation terminates with an error if no devices are connected to the GPIB. The actual number of bytes transferred is returned in the global variable `ibcnt`.

Board and Device Level

The asynchronous I/O calls (`ibcmda`, `ibrda`, `ibwrta`) are designed so that applications can perform other non-GPIB operations while the I/O is in progress. Once the asynchronous I/O has begun, further GPIB calls are strictly limited. Any calls that would interfere with the I/O in progress are not allowed; the driver returns EOIP in this case.

Once the I/O is complete, the application must *resynchronize* with the NI-488.2 driver. Resynchronization is accomplished by using one of the following three functions:

- `ibwait` If the returned `ibsta` mask has the CMPL bit set, then the driver and application are resynchronized.
- `ibstop` The I/O is canceled; the driver and application are resynchronized.
- `ibonl` The I/O is canceled and the interface is reset; the driver and application are resynchronized.

Possible Errors

EABO	Board level: a Device Clear message was received from the CIC.
EADR	Board level: The GPIB is not correctly addressed. Use `ibcmd` to address the GPIB. Device level: A conflict exists between the device GPIB address and the GPIB address of the device access board. Use `ibpad` and `ibsad`.
EBUS	Device level: No devices are connected to the GPIB.
ECIC	Device level: The access board is not CIC. See the *Device-Level Calls and Bus Management* section in Chapter 5, *GPIB Programming Techniques*, of the *NI-488.2 User Manual for MacOS.*
EDVR	Either `ud` is invalid or the NI-488.2 driver is not installed.
ENEB	The interface board is not installed or is not properly configured.
EOIP	Asynchronous I/O is in progress.

IBRDF

Board Level/Device Level

Purpose

Read data from a device into a file.

Format

C

short ibrdf (short ud, char *flname)

FutureBASIC

```
FN ibrdf%(ud%,flname$)
```

Input

ud	Board or device descriptor
flname	Name of file into which data is read

Output

Function Return	The value of ibsta

Description

Board Level

If ud is a board descriptor, ibrdf reads up to cnt bytes of data from a GPIB device and places the data into the file specified by flname. A board-level ibrdf 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 ibcnt.

Device Level

If ud is a device descriptor, ibrdf addresses the GPIB, reads up to cnt bytes of data from a GPIB device, and places the data into the file specified by flname. The operation terminates normally when cnt bytes have been received or END is received. The

IBRDF

(Continued)

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 `ibcnt`.

Possible Errors

EABO	Either `cnt` bytes or END was not received within the timeout period, or `ud` is a board descriptor and Device Clear was received after the read operation began.
EADR	Board level: The GPIB is not correctly addressed. Use `ibcmd` to address the GPIB. Device level: A conflict exists between the device GPIB address and the GPIB address of the device access board. Use `ibpad` and `ibsad`.
EBUS	Device level: No devices are connected to the GPIB.
ECIC	Device level: The access board is not CIC. See the *Device-Level Calls and Bus Management* section in Chapter 5, *GPIB Programming Techniques*, of the *NI-488.2 User Manual for MacOS.*
EDVR	Either `ud` is invalid or the NI-488.2 driver is not installed.
EFSO	`ibrdf` could not access `flname`.
ENEB	The interface board is not installed or is not properly configured.
EOIP	Asynchronous I/O is in progress.

IBRPP

Board Level/Device Level

Purpose

Conduct a parallel poll.

Format

C

short ibrpp (short ud, char *ppr)

FutureBASIC

```
FN ibrpp%(ud%,ppr&)
```

Input

ud	Board or device descriptor

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.

For more information on parallel polling, refer to the *NI-488.2 User Manual for MacOS.*

Possible Errors

EBUS	Device level: No devices are connected to the GPIB.
ECIC	Device level: The access board is not CIC. See the *Device-Level Calls and Bus Management* section in Chapter 5, *GPIB Programming Techniques*, of the *NI-488.2 User Manual for MacOS.*
EDVR	Either ud is invalid or the NI-488.2 driver is not installed.
ENEB	The interface board is not installed or is not properly configured.
EOIP	Asynchronous I/O is in progress.

IBRSC

Board Level

Purpose

Request or release system control.

Format

C

short ibrsc (short ud, short v)

FutureBASIC

```
FN ibrsc%(ud%,v%)
```

Input

ud	Board descriptor
v	Determines if system control is to be requested or released

Output

Function Return	The value of ibsta

Description

ibrsc requests or releases the capability to send Interface Clear (IFC) and Remote Enable (REN) messages to devices. If v is zero, the board releases system control and functions requiring System Controller capability are not allowed. If v is non-zero, functions requiring System Controller capability are subsequently allowed. If no error occurs during the call, then iberr contains the previous System Controller state of the board.

Possible Errors

EARG	ud is a valid descriptor but does not refer to a board.
EDVR	Either ud is invalid or the NI-488.2 driver is not installed.
ENEB	The interface board is not installed or is not properly configured.
EOIP	Asynchronous I/O is in progress.

IBRSP

Device Level

Purpose

Conduct a serial poll.

Format

C

short `ibrsp` (short `ud`, char `*spr`)

FutureBASIC

```
FN ibrsp%(ud%,spr&)
```

Input

`ud`	Device descriptor

Output

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

Description

The `ibrsp` function is used to serial poll the device `ud`. 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. If the automatic serial polling feature is enabled, the device might have already been polled. In this case, `ibrsp` returns the previously acquired status byte.

For more information on serial polling, refer to the *NI-488.2 User Manual for MacOS.*

Possible Errors

EABO	The serial poll response could not be read within the serial poll timeout period.
EARG	`ud` is a valid descriptor but does not refer to a device.
EBUS	No devices are connected to the GPIB.
ECIC	The access board is not CIC. See the *Device-Level Calls and Bus Management* section in Chapter 5, *GPIB Programming Techniques*, of the *NI-488.2 User Manual for MacOS.*

IBRSP

(Continued)

EDVR	Either `ud` is invalid or the NI-488.2 driver is not installed.
ENEB	The interface board is not installed or is not properly configured.
EOIP	Asynchronous I/O is in progress.
ESTB	Autopolling is enabled and the serial poll queue has overflowed. Disable automatic serial polling or call `ibrsp` more often to keep the queue from overflowing.

IBRSV

Board Level

Purpose

Request service and change the serial poll status byte.

Format

C

short ibrsv (short ud, short v)

FutureBASIC

```
FN ibrsv%(ud%,v%)
```

Input

ud	Board descriptor
v	Serial poll status byte

Output

Function Return	The value of ibsta

Description

ibrsv requests service from the Controller and provides the Controller with an application-dependent status byte when the Controller serial polls the GPIB board.

The value v is the status byte that the GPIB board returns when serial polled by the Controller-In-Charge. If bit 6 (hex 40) is set in v, the GPIB board requests service from the Controller by asserting the GPIB SRQ line. When ibrsv is called and an error does not occur, the previous status byte is returned in iberr.

Possible Errors

EARG	ud is a valid descriptor but does not refer to a board.
EDVR	Either ud is invalid or the NI-488.2 driver is not installed.
ENEB	The interface board is not installed or is not properly configured.
EOIP	Asynchronous I/O is in progress.

IBSAD

Board Level/Device Level

Purpose

Change or disable the secondary address.

Format

C

short ibsad (short ud, short v)

FutureBASIC

```
FN ibsad%(ud%,v%)
```

Input

ud	Board or device descriptor
v	GPIB secondary address

Output

Function Return	The value of ibsta

Description

ibsad changes the secondary GPIB address of the board or device to v, an integer in the range 96 to 126 (hex 60 to hex 7E) or zero. If v is zero, secondary addressing is disabled. If no error occurs during the call, then the previous secondary address is returned in iberr.

Possible Errors

EARG	v is non-zero and outside the legal range 96 to 126.
EDVR	Either ud is invalid or the NI-488.2 driver is not installed.
ENEB	The interface board is not installed or is not properly configured.
EOIP	Asynchronous I/O is in progress.

IBSIC

Board Level

Purpose

Assert interface clear.

Format

C

short `ibsic` (short `ud`)

FutureBASIC

```
FN ibsic%(ud%)
```

Input

`ud`	Board descriptor

Output

Function Return	The value of `ibsta`

Description

`ibsic` asserts the GPIB interface clear (IFC) line for at least 100 µs if the GPIB board is System Controller. This initializes the GPIB and makes the interface board 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	`ud` is a valid descriptor but does not refer to a board.
EDVR	Either `ud` is invalid or the NI-488.2 driver is not installed.
ENEB	The interface board is not installed or is not properly configured.
EOIP	Asynchronous I/O is in progress.
ESAC	Board does not have System Controller capability.

IBSRE

Board Level

Purpose

Set or clear the Remote Enable line.

Format

C

short `ibsre` (short `ud`, short `v`)

FutureBASIC

```
FN ibsre%(ud%,v%)
```

Input

`ud`	Board descriptor
`v`	Indicates whether to set or clear the REN line

Output

Function Return	The value of `ibsta`

Description

If `v` is non-zero, the GPIB Remote Enable (REN) line is asserted. If `v` is zero, REN is unasserted. The previous value of REN is returned in `iberr`.

REN is used by devices to choose between local and remote modes of operation. A device should not actually enter remote mode until it receives its listen address.

Possible Errors

EARG	`ud` is a valid descriptor but does not refer to a board.
EDVR	Either `ud` is invalid or the NI-488.2 driver is not installed.
ENEB	The interface board is not installed or is not properly configured.
EOIP	Asynchronous I/O is in progress.
ESAC	Board does not have System Controller capability.

IBSRQ

Board Level

Purpose

Request an SRQ interrupt routine.

Format

C

short `ibsrq` (void (`*func`) (void))

FutureBASIC

```
FN ibsrq%(func&)
```

Input

`func`	C interrupt-handling routine

Description

`ibsrq` establishes a call to the C routine `func` whenever the SRQI bit is set in the status word (`ibsta`). If SRQI is set, the language interface calls `func` before returning to the application program. If `ibsrq` is called with `funcname` equal to NULL, SRQ servicing is turned off.

☞ **Note:** ***You must disable automatic serial polling with*** `ibconfig` ***(option*** `IbcAUTOPOLL`***) before using this function. Also, device-level calls should not be used when*** `ibsrq` ***is in effect. Device-level calls mask the SRQI bit, preventing*** `func` ***from being called.***

IBSTOP

Board Level/Device Level

Purpose

Abort asynchronous I/O operation.

Format

C

short `ibstop` (short `ud`)

FutureBASIC

```
FN ibstop%(ud%)
```

Input

`ud`	Board or device descriptor

Output

Function Return	The value of `ibsta`

Description

The `ibstop` function aborts any asynchronous read, write, or command operation that is in progress and resynchronizes the application with the driver. If asynchronous I/O is in progress, the error bit is set in the status word, `ibsta`, and EABO is returned, indicating that the I/O was successfully stopped.

Possible Errors

EABO	Asynchronous I/O was successfully stopped.
EDVR	Either `ud` is invalid or the NI-488.2 driver is not installed.
ENEB	The interface board is not installed or is not properly configured.

IBTMO

Board Level/Device Level

Purpose

Change or disable the I/O timeout period.

Format

C

short ibtmo (short ud, short v)

FutureBASIC

```
FN ibtmo%(ud%,v%)
```

Input

ud	Board or device descriptor
v	Timeout duration code

Output

Function Return	The value of ibsta

Description

The timeout period is set to v. The timeout period is used to select the maximum duration allowed for a synchronous operation (for example, ibrd and ibwait). If the operation does not complete before the timeout period elapses, then the operation is aborted and TIMO is returned in ibsta. See Table 1-8 for a list of valid timeout values. These timeout values represent the minimum timeout period. The actual period might be longer.

Table 1-8. Timeout Code Values

Constant	Value of v	Minimum Timeout
TNONE	0	disabled - no timeout
T10us	1	10 µs
T30us	2	30 µs
T100us	3	100 µs

IBTMO

(Continued)

Table 1-8. Timeout Code Values (Continued)

Constant	Value of v	Minimum Timeout
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

Possible Errors

EARG	v is invalid.
EDVR	Either ud is invalid or the NI-488.2 driver is not installed.
ENEB	The interface board is not installed or is not properly configured.

IBTRG

Device Level

Purpose

Trigger selected device.

Format

C

short `ibtrg` (short `ud`)

FutureBASIC

```
FN ibtrg%(ud%)
```

Input

`ud`	Device descriptor

Output

Function Return	The value of `ibsta`

Description

`ibtrg` sends the Group Execute Trigger (GET) message to the device described by `ud`.

Possible Errors

EARG	`ud` is a valid descriptor but does not refer to a device.
EBUS	No devices are connected to the GPIB.
ECIC	The access board is not CIC. See the *Device-Level Calls and Bus Management* section in Chapter 5, *GPIB Programming Techniques*, of the *NI-488.2 User Manual for MacOS*.
EDVR	Either `ud` is invalid or the NI-488.2 driver is not installed.
ENEB	The interface board is not installed or is not properly configured.
EOIP	Asynchronous I/O is in progress.

IBUNLOCK (GPIB-ENET only)

Board Level/Device Level

Purpose

Unlock access to a GPIB-ENET board or device.

Format

C

short `ibunlock` (short `ud`)

FutureBASIC

```
FN ibunlock%(ud%)
```

Input

`ud`	A board or device descriptor

Output

Function Return	The value of `ibsta`

Description

The `ibunlock` function releases the lock on the board or device connection requested by `iblock`.

Board Level

When the `iblock` function has been used to lock access to a board, an `ibunlock` function call made with the same board descriptor unlocks access to the board.

Device Level

When the `iblock` function has been used to lock access to a device, an `ibunlock` function call made with the same device descriptor unlocks access to the device.

IBUNLOCK (GPIB-ENET only)

(Continued)

Recommended Usage

In general, use `ibunlock` to release your lock on a board or device connection. It is recommended that `ibunlock` be used immediately after critical board or device accesses are made to a locked interface.

Refer also to *IBLOCK (GPIB-ENET only)* earlier in this chapter.

Possible Errors

EDVR	Either `ud` is invalid or the NI-488.2 driver is not installed.
ELCK	Occurs if the GPIB-ENET board or device being locked is locked by another process.

IBWAIT

Board Level/Device Level

Purpose

Wait for GPIB events.

Format

C

short `ibwait` (short `ud`, short `mask`)

FutureBASIC

```
FN ibwait%(ud%,mask%)
```

Input

`ud`	Board or device descriptor
`mask`	Bit mask of GPIB events to wait on

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 the updated `ibsta` status word. 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 they are described in Table 1-9. If `ud` is a device descriptor, the only valid wait mask bits are TIMO, END, RQS and CMPL. If `ud` is a board descriptor, all wait mask bits are valid except for RQS. You can configure the timeout period using the `ibtmo` function.

Possible Errors

EARG	The bit set in `mask` is invalid.
EBUS	Device level: No devices are connected to the GPIB.

IBWAIT

(Continued)

ECIC — Device level: The access board is not CIC. See the *Device-Level Calls and Bus Management* section in Chapter 5, *GPIB Programming Techniques*, of the *NI-488.2 User Manual for MacOS.*

EDVR — Either ud is invalid or the NI-488.2 driver is not installed.

ENEB — The interface board is not installed or is not properly configured.

ESRQ — Device level: If RQS is set in the wait mask, then ESRQ indicates that the *Stuck SRQ* condition exists. For more information on serial polling, refer to the *NI-488.2 User Manual for MacOS.*

Table 1-9. Wait Mask Layout

Mnemonic	Usage Level	Hex Value	Description
TIMO	bd/dev	4000	Time limit exceeded
END	bd/dev	2000	GPIB board detected END or EOS
SRQI	board	1000	SRQ asserted (board only)
RQS	device	800	Device requesting service (device only)
CMPL	bd/dev	100	I/O completed
LOK	board	80	GPIB board is in Lockout State
REM	board	40	GPIB board is in Remote State
CIC	board	20	GPIB board is CIC
ATN	board	10	Attention is asserted
TACS	board	8	GPIB board is Talker
LACS	board	4	GPIB board is Listener
DTAS	board	2	GPIB board is in Device Trigger State
DCAS	board	1	GPIB board is in Device Clear State

IBWRT

Board Level/Device Level

Purpose

Write data to a device from a user buffer.

Format

C

short `ibwrt` (short `ud`, char `*wrtbuffer`, long `cnt`)

FutureBASIC

```
FN ibwrt%(ud%,wrtbuffer&,cnt&)
```

Input

`ud`	Board or device descriptor
`wrtbuffer`	Address of the buffer containing the bytes to write
`cnt`	Number of bytes to be written

Output

Function Return	The value of `ibsta`

Description

Board Level

If `ud` is a board descriptor, `ibwrt` writes `cnt` bytes of data from the buffer specified by `wrtbuffer` 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 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 `ibcnt`.

IBWRT

(Continued)

Device Level

If `ud` is a device descriptor, `ibwrt` addresses the GPIB and writes `cnt` bytes from the memory location specified by `wrtbuffer` 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. The actual number of bytes transferred is returned in the global variable `ibcnt`.

Possible Errors

EABO	Either `cnt` bytes were not sent within the timeout period, or 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.
	Device level: A conflict exists between the device GPIB address and the GPIB address of the device access board. Use `ibpad` and `ibsad`.
EBUS	Device level: No devices are connected to the GPIB.
ECIC	Device level: The access board is not CIC. See the *Device-Level Calls and Bus Management* section in Chapter 5, *GPIB Programming Techniques*, of the *NI-488.2 User Manual for MacOS.*
EDVR	Either `ud` is invalid or the NI-488.2 driver is not installed.
ENEB	The interface board is not installed or is not properly configured.
ENOL	No Listeners were detected on the bus.
EOIP	Asynchronous I/O is in progress.

IBWRTA

Board Level/Device Level

Purpose

Write data asynchronously to a device from a user buffer.

Format

C

short ibwrta (short ud, char *wrtbuffer, long cnt)

FutureBASIC

```
FN ibwrta%(ud%,wrtbuffer&,cnt&)
```

Input

ud	Board or device descriptor
wrtbuffer	Address of the buffer containing the bytes to write
cnt	Number of bytes to be written

Output

Function Return	The value of ibsta

Description

Board Level

If ud is a board descriptor, ibwrta begins an asynchronous write of cnt bytes of data from the buffer pointed to by wrtbuffer to a GPIB device. A board-level ibwrta 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 the board is not CIC and the CIC sends the Device Clear message on the GPIB. The actual number of bytes transferred is returned in the global variable ibcnt.

Device Level

If ud is a device descriptor, ibwrta addresses the GPIB and writes cnt bytes from the buffer wrtbuffer to a GPIB device. The operation terminates normally when cnt bytes

IBWRTA

(Continued)

have been sent. The operation terminates with an error if no devices are connected to the GPIB. The actual number of bytes transferred is returned in the global variable `ibcnt`.

Board and Device Level

The asynchronous I/O calls (`ibcmda`, `ibrda`, `ibwrta`) are designed so that applications can perform other non-GPIB operations while the I/O is in progress. Once the asynchronous I/O has begun, further GPIB calls are strictly limited. Any calls that would interfere with the I/O in progress are not allowed; the driver returns EOIP in this case.

Once the I/O is complete, the application must *resynchronize* with the NI-488.2 driver. Resynchronization is accomplished by using one of the following three functions:

- `ibwait` If the returned `ibsta` mask has the CMPL bit set, then the driver and application are resynchronized.
- `ibstop` The I/O is canceled; the driver and application are resynchronized.
- `ibonl` The I/O is canceled and the interface is reset; the driver and application are resynchronized.

Possible Errors

EABO	Board level: a Device Clear message was received from the CIC.
EADR	Board level: The GPIB is not correctly addressed. Use `ibcmd` to address the GPIB. Device level: A conflict exists between the device GPIB address and the GPIB address of the device access board. Use `ibpad` and `ibsad`.
EBUS	Device level: No devices are connected to the GPIB.
ECIC	Device level: The access board is not CIC. See the *Device-Level Calls and Bus Management* section in Chapter 5, *GPIB Programming Techniques*, of the *NI-488.2 User Manual for MacOS.*
ENEB	The interface board is not installed or is not properly configured.
ENOL	No Listeners were detected on the bus.
EOIP	Asynchronous I/O is in progress.

IBWRTF

Board Level/Device Level

Purpose

Write data to a device from a file.

Format

C

short ibwrtf (short ud, char flname [])

FutureBASIC

```
FN ibwrtf%(ud%,flname$)
```

Input

ud	Board or device descriptor
flname	Name of file containing the data to be written

Output

Function Return	The value of ibsta

Description

Board Level

If ud is a board descriptor, ibwrtf writes cnt bytes of data from the file flname to a GPIB device. A board-level ibwrtf 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 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 ibcnt.

Device Level

If ud is a device descriptor, ibwrtf addresses the GPIB and writes cnt bytes from the file flname 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

IBWRTF

(Continued)

the timeout period. The actual number of bytes transferred is returned in the global variable `ibcnt`.

Possible Errors

EABO	Either the file could not be transferred within the timeout period or 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. Device level: A conflict exists between the device GPIB address and the GPIB address of the device access board. Use `ibpad` and `ibsad`.
EBUS	Device level: No devices are connected to the GPIB.
ECIC	Device level: The access board is not CIC. See the *Device-Level Calls and Bus Management* section in Chapter 5, *GPIB Programming Techniques*, of the *NI-488.2 User Manual for MacOS.*
EDVR	Either `ud` is invalid or the NI-488.2 driver is not installed.
EFSO	`ibwrtf` could not access `flname`.
ENEB	The interface board is not installed or is not properly configured.
EOIP	Asynchronous I/O is in progress.

Chapter 2

NI-488.2 Routines

This chapter lists the available NI-488.2 routines and describes the purpose, format, input and output parameters, and possible errors for each routine.

While using the routines, you might find it helpful to refer to Chapter 2, *Developing Your Application*, and Chapter 5, *GPIB Programming Techniques*, in the *NI-488.2 User Manual for MacOS*.

Routine Names

The routines in this chapter are listed alphabetically.

Purpose

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

Format

The format is given for each of the languages supported by the NI-488.2 software:

- MPW C version 3.0 or higher, THINK C version 4.0 or higher, and Metrowerks CodeWarrior version 1.1 or higher
- FutureBASIC II

Input and Output

The input and output parameters for each routine are listed. Most of the NI-488.2 routines have an input parameter which is either a single address or a list of addresses. The address parameter is a 16-bit integer that has two components: the low byte is a valid primary address (0 to 30), and the high byte is a valid secondary address (`NO_SAD(0)` or 96 to 126). A list of addresses is an array of single addresses. You must mark the end of this list with the constant `NOADDR`. An empty address list is either an array with only the `NOADDR` constant in it, or a `NULL` pointer.

Description

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

Possible Errors

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

List of NI-488.2 Routines

The following table contains an alphabetical list of each NI-488.2 routine.

Table 2-1. List of NI-488.2 Routines

Routine	Purpose
`AllSpoll`	Serial poll all devices
`DevClear`	Clear a single device
`DevClearList`	Clear multiple devices
`EnableLocal`	Enable operations from the front panel of devices (leave remote programming mode)
`EnableRemote`	Enable remote GPIB programming for devices
`FindLstn`	Find listening devices on the GPIB
`FindRQS`	Determine which device is requesting service
`PassControl`	Pass control to another device with Controller capability
`PPoll`	Perform a parallel poll on the GPIB
`PPollConfig`	Configure a device for parallel polls
`PPollUnconfig`	Unconfigure devices for parallel polls
`RcvRespMsg`	Read data bytes from a device that is already addressed to talk
`ReadStatusByte`	Serial poll a single device
`Receive`	Read data bytes from a device
`ReceiveSetup`	Address a device to be a Talker and the interface board ID to be a Listener in preparation for `RcvRespMsg`
`ResetSys`	Reset and initialize IEEE 488.2-compliant devices

Table 2-1. List of NI-488.2 Routines (Continued)

Routine	Purpose
`Send`	Send data bytes to a device
`SendCmds`	Send GPIB command bytes
`SendDataBytes`	Send data bytes to devices that are already addressed to listen
`SendIFC`	Reset the GPIB by sending interface clear
`SendList`	Send data bytes to multiple GPIB devices
`SendLLO`	Send the Local Lockout (LLO) message to all devices
`SendSetup`	Set up devices to receive data in preparation for `SendDataBytes`
`SetRWLS`	Place devices in remote with lockout state
`TestSRQ`	Determine the current state of the GPIB Service Request (SRQ) line
`TestSys`	Cause the IEEE 488.2-compliant devices to conduct self tests
`Trigger`	Trigger a device
`TriggerList`	Trigger multiple devices
`WaitSRQ`	Wait until a device asserts the GPIB Service Request (SRQ) line

AllSpoll

Purpose

Serial poll all devices.

Format

C

void AllSpoll (short board, short addresslist [], short resultlist [])

FutureBASIC

```
FN AllSpoll (board%,@addresslist%(0),@resultlist%(0))
```

Input

board	The interface board number
addresslist	A list of device addresses that is terminated by NOADDR

Output

resultlist	A list of serial poll response bytes corresponding to device addresses in addresslist

Description

AllSpoll serial polls all of the devices described by addresslist. It stores the poll responses in resultlist and the number of responses in ibcnt.

Possible Errors

EABO	One of the devices timed out instead of responding to the serial poll; ibcnt contains the index of the timed-out device.
EARG	An invalid address (out of range) appears in addresslist; ibcnt is the index of the invalid address in the addresslist array.
EBUS	No devices are connected to the GPIB.
ECIC	The interface board is not the Controller-In-Charge; see SendIFC.
EDVR	Either board is invalid (out of range) or the NI-488.2 driver is not installed.
ENEB	The interface board is not installed or is not properly configured.
EOIP	Asynchronous I/O is in progress.

DevClear

Purpose

Clear a single device.

Format

C

void `DevClear` (short `board`, short `address`)

FutureBASIC

```
FN DevClear (board%,address%)
```

Input

`board`	The interface board number
`address`	Address of the device you want to clear

Description

`DevClear` sends the Selected Device Clear (SDC) GPIB message to the device described by `address`. If `address` is the constant `NOADDR`, then the Universal Device Clear (DCL) message is sent to all devices.

Possible Errors

EARG	An `address` parameter is invalid (out of range).
EBUS	No devices are connected to the GPIB.
ECIC	The interface board is not the Controller-In-Charge; see `SendIFC`.
EDVR	Either `board` is invalid (out of range) or the NI-488.2 driver is not installed.
ENEB	The interface board is not installed or is not properly configured.
EOIP	Asynchronous I/O is in progress.

DevClearList

Purpose

Clear multiple devices.

Format

C

void `DevClearList` (short `board`, short `addresslist []`)

FutureBASIC

```
FN DevClearList (board%,@addresslist%(0))
```

Input

`board`	The interface board number
`addresslist`	A list of device addresses terminated by `NOADDR` that you want to clear

Description

`DevClearList` sends the Selected Device Clear (SDC) GPIB message to all the device addresses described by `addresslist`. If `addresslist` contains only the constant `NOADDR`, then the Universal Device Clear (DCL) message is sent to all the devices on the bus.

Possible Errors

EARG	An invalid address (out of range) appears in `addresslist`; `ibcnt` is the index of the invalid address in the `addresslist` array.
EBUS	No devices are connected to the GPIB.
ECIC	The interface `board` is not the Controller-In-Charge; see `SendIFC`.
EDVR	Either `board` is invalid (out of range) or the NI-488.2 driver is not installed.
ENEB	The interface board is not installed or is not properly configured.
EOIP	Asynchronous I/O is in progress.

EnableLocal

Purpose

Enable operations from the front panel of devices (leave remote programming mode).

Format

C

void `EnableLocal` (short `board`, short `addresslist []`)

FutureBASIC

```
FN EnableLocal (board%,@addresslist%(0))
```

Input

`board`	The interface board number
`addresslist`	A list of device addresses that is terminated by `NOADDR`

Description

`EnableLocal` sends the Go To Local (GTL) GPIB message to all the devices described by `addresslist`. This places the devices in local mode. If `addresslist` contains only the constant `NOADDR`, then the Remote Enable (REN) GPIB line is unasserted.

Possible Errors

EARG	An invalid address (out of range) appears in `addresslist`; `ibcnt` is the index of the invalid address in the `addresslist` array.
EBUS	No devices are connected to the GPIB.
ECIC	The interface board is not the Controller-In-Charge; see `SendIFC`.
EDVR	Either `board` is invalid (out of range) or the NI-488.2 driver is not installed.
ENEB	The interface board is not installed or is not properly configured.
EOIP	Asynchronous I/O is in progress.
ESAC	The interface board is not configured as System Controller.

EnableRemote

Purpose

Enable remote GPIB programming for devices.

Format

C

void `EnableRemote` (short `board`, short `addresslist []`)

FutureBASIC

```
FN EnableRemote (board%,@addresslist%(0))
```

Input

`board`	The interface board number
`addresslist`	A list of device addresses that is terminated by `NOADDR`

Description

`EnableRemote` asserts the Remote Enable (REN) GPIB line. All devices described by `addresslist` are put in a listen-active state.

Possible Errors

EARG	An invalid address (out of range) appears in `addresslist`; `ibcnt` is the index of the invalid address in the `addresslist` array.
EBUS	No devices are connected to the GPIB.
ECIC	The interface board is not the Controller-In-Charge; see `SendIFC`.
EDVR	Either `board` is invalid (out of range) or the NI-488.2 driver is not installed.
ENEB	The interface board is not installed or is not properly configured.
EOIP	Asynchronous I/O is in progress.
ESAC	The interface board is not configured as System Controller.

FindLstn

Purpose

Find listening devices on the GPIB.

Format

C

void FindLstn (short board, Addr4882_t addresslist [ ], Addr4882_t resultlist [ ], short limit)

FutureBASIC

```
FN FindLstn (board%,@addresslist%(0),@resultlist%(0),limit%)
```

Input

board	The interface board number
addresslist	A list of primary addresses that is terminated by NOADDR
limit	Total number of entries that can be placed in resultlist

Output

resultlist	Addresses of all listening devices found by FindLstn are placed in this array.

Description

FindLstn tests all of the primary addresses in addresslist. If a device is present at a primary address given in addresslist, then the primary address is stored in resultlist. Otherwise, all secondary addresses of the primary address are tested, and the addresses of any devices found are stored in resultlist. No more than limit addresses are stored in resultlist; ibcnt contains the actual number of addresses stored in resultlist.

Possible Errors

EARG	An invalid primary address (out of range) appears in addresslist; ibcnt is the index of the invalid address in the addresslist array.
EBUS	No devices are connected to the GPIB.
ECIC	The interface board is not the Controller-In-Charge; see SendIFC.

FindLstn

(Continued)

EDVR	Either `board` is invalid (out of range) or the NI-488.2 driver is not installed.
ENEB	The interface board is not installed or is not properly configured.
EOIP	Asynchronous I/O is in progress.
ETAB	The number of devices found on the GPIB exceed `limit`.

FindRQS

Purpose

Determine which device is requesting service.

Format

C

void FindRQS (short board, short addresslist [], short *result)

FutureBASIC

```
FN FindRQS (board%,@addresslist%(0),result&)
```

Input

board	The interface board number
addresslist	List of device addresses that is terminated by NOADDR

Output

result	Serial poll response byte of the device that is requesting service

Description

FindRQS serial polls the devices described by addresslist, in order, until it finds a device which is requesting service. The serial poll response byte is then placed in result. ibcnt contains the index of the device requesting service in addresslist. If none of the devices are requesting service, then the index corresponding to NOADDR in addresslist is returned in ibcnt and ETAB is returned in iberr.

Possible Errors

EARG	An invalid address (out of range) appears in addresslist; ibcnt is the index of the invalid address in the addresslist array.
EBUS	No devices are connected to the GPIB.
ECIC	board is not the Controller-In-Charge; see SendIFC.
EDVR	Either board is invalid (out of range) or the NI-488.2 driver is not installed.
ENEB	board is not installed or is not properly configured.

FindRQS

(Continued)

EOIP	Asynchronous I/O is in progress.
ETAB	None of the devices in `addresslist` are requesting service or `addresslist` contains only `NOADDR`. `ibcnt` contains the index of `NOADDR` in `addresslist`.

PassControl

Purpose

Pass control to another device with Controller capability.

Format

C

void PassControl (short board, short address)

FutureBASIC

```
FN PassControl (board%,address%)
```

Input

board	The interface board number
address	Address of the device to which you want to pass control

Description

PassControl sends the Take Control (TCT) GPIB message to the device described by address. That device becomes Controller-In-Charge and board is no longer CIC.

Possible Errors

EARG	The address parameter is invalid (out of range) or NOADDR.
EBUS	No devices are connected to the GPIB.
ECIC	The interface board is not the Controller-In-Charge; see SendIFC.
EDVR	Either board is invalid (out of range) or the NI-488.2 driver is not installed.
ENEB	The interface board is not installed or is not properly configured.
EOIP	Asynchronous I/O is in progress.

PPoll

Purpose

Perform a parallel poll on the GPIB.

Format

C

void `PPoll` (short `board`, short `*result`)

FutureBASIC

```
FN PPoll (board%, result&)
```

Input

`board`	The interface board number

Output

`result`	The parallel poll result

Description

`PPoll` conducts a parallel poll and the result is placed in `result`. Each of the eight bits of `result` represents the status information for each device configured for a parallel poll. The interpretation of the status information is based on the latest parallel poll configuration command sent to each device (see `PPollConfig` and `PPollUnconfig`). The Controller can use parallel polling to obtain one-bit, device-dependent status messages from up to eight devices simultaneously.

For more information on parallel polling, refer to the *NI-488.2 User Manual for MacOS.*

Possible Errors

EBUS	No devices are connected to the GPIB.
ECIC	The interface board is not the Controller-In-Charge; see `SendIFC`.
EDVR	Either `board` is invalid (out of range) or the NI-488.2 driver is not installed.
ENEB	The interface board is not installed or is not properly configured.
EOIP	Asynchronous I/O is in progress.

PPollConfig

Purpose

Configure a device to respond to parallel polls.

Format

C

void `PPollConfig` (short `board`, short `address`, short `dataline`, short `sense`)

FutureBASIC

```
FN PPollConfig (board%,address%,dataline%,sense%)
```

Input

`board`	The interface board number
`address`	Address of the device to be configured
`dataline`	Data line (a value in the range of 1 to 8) on which the device responds to parallel polls
`sense`	Sense (either 0 or 1) of the parallel poll response

Description

`PPollConfig` configures the device described by `address` to respond to parallel polls by asserting or not asserting the GPIB data line, `dataline`. If `sense` equals the individual status (`ist`) bit of the device, then the assigned GPIB data line is asserted during a parallel poll. Otherwise, the data line is not asserted during a parallel poll. The Controller can use parallel polling to obtain one-bit, device-dependent status messages from up to eight devices simultaneously.

For more information on parallel polling, refer to the *NI-488.2 User Manual for MacOS.*

Possible Errors

EARG	The `address` parameter is invalid (out of range) or `NOADDR`; `dataline` is not in the range 1 to 8, or `Sense` is not 0 or 1.
EBUS	No devices are connected to the GPIB.
ECIC	The interface board is not the Controller-In-Charge; see `SendIFC`.
EDVR	Either `board` is invalid (out of range) or the NI-488.2 driver is not installed.

PPollConfig

(Continued)

ENEB	The interface board is not installed or is not properly configured.
EOIP	Asynchronous I/O is in progress.

PPollUnconfig

Purpose

Unconfigure devices for parallel polls.

Format

C

void PPollUnconfig (short board, short addresslist [])

FutureBASIC

```
FN PPollUnconfig (board%,@addresslist%(0))
```

Input

board	The interface board number
addresslist	A list of device addresses that is terminated by NOADDR

Description

PPollUnconfig unconfigures all the devices described by addresslist for parallel polls. If addresslist contains only the constant NOADDR, then the Parallel Poll Unconfigure (PPU) GPIB message is sent to all GPIB devices. The devices unconfigured by this function do not participate in subsequent parallel polls.

For more information on parallel polling, refer to the *NI-488.2 User Manual for MacOS.*

Possible Errors

EARG	An invalid address (out of range) appears in addresslist; ibcnt is the index of the invalid address in the addresslist array.
EBUS	No devices are connected to the GPIB.
ECIC	The interface board is not the Controller-In-Charge; see SendIFC.
EDVR	Either board is invalid (out of range) or the NI-488.2 driver is not installed.
ENEB	The interface board is not installed or is not properly configured.
EOIP	Asynchronous I/O is in progress.

RcvRespMsg

Purpose

Read data bytes from a device that is already addressed to talk.

Format

C

void RcvRespMsg (short board, char data [], long cnt, short termination)

FutureBASIC

```
FN RcvRespMsg (board%,data&,cnt&,termination%)
```

Input

board	The interface board number
cnt	Number of bytes read
termination	Description of the data termination mode (STOPend or an 8-bit EOS character)

Output

data	Stores the received data bytes

Description

RcvRespMsg reads up to cnt bytes from the GPIB and places these bytes into data. Data bytes are read until either cnt data bytes have been read or the termination condition is detected. If the termination condition is STOPend, the read is stopped when a byte is received with the EOI line asserted. Otherwise, the read is stopped when the 8-bit EOS character is detected. The actual number of bytes transferred is returned in the global variable ibcnt.

RcvRespMsg assumes that the interface board is already in listen-active state and a device is already addressed to be a Talker (see ReceiveSetup or Receive).

Possible Errors

EABO	The I/O timeout period elapsed before all the bytes were received.
EADR	The interface board is not in the listen-active state; use ReceiveSetup to address the GPIB properly.

RcvRespMsg

(Continued)

EARG	The `termination` parameter is invalid. It must be either `STOPend` or an 8-bit EOS character.
ECIC	The interface board is not the Controller-In-Charge; see `SendIFC`.
EDVR	Either `board` is invalid (out of range) or the NI-488.2 driver is not installed.
ENEB	The interface board is not installed or is not properly configured.
EOIP	Asynchronous I/O is in progress.

ReadStatusByte

Purpose

Serial poll a single device.

Format

C

void ReadStatusByte (short board, short address, short *result)

FutureBASIC

```
FN ReadStatusByte (board%,address%,result&)
```

Input

board	The interface board number
address	A device address

Output

result	Serial poll response byte

Description

ReadStatusByte serial polls the device described by address. The response byte is stored in result.

Possible Errors

EABO	The device times out instead of responding to the serial poll.
EARG	The address parameter is invalid (out of range).
EBUS	No devices are connected to the GPIB.
ECIC	The interface board is not the Controller-In-Charge; see SendIFC.
EDVR	Either board is invalid (out of range) or the NI-488.2 driver is not installed.
ENEB	The interface board is not installed or is not properly configured.
EOIP	Asynchronous I/O is in progress.

Receive

Purpose

Read data bytes from a device.

Format

C

void Receive (short board, short address, char data [], unsigned long cnt, short termination)

FutureBASIC

```
FN Receive (board%,address%,data&,cnt&,termination%)
```

Input

board	The interface board number
address	Address of a device to receive data
cnt	Number of bytes to read
termination	Description of the data termination mode (STOPend or an EOS character)

Output

data	Stores the received data bytes

Description

Receive addresses the device described by address to talk and the interface board to listen. Then up to cnt bytes are read and placed into the buffer. Data bytes are read until either cnt bytes have been read or the termination condition is detected. If the termination condition is STOPend, the read is stopped when a byte is received with the EOI line asserted. Otherwise, the read is stopped when the 8-bit EOS character is detected. The actual number of bytes transferred is returned in the global variable ibcnt.

Possible Errors

EABO	The I/O timeout period elapsed before all the bytes were received.
EARG	The address or termination parameter is invalid (out of range), or address is NOADDR.

Receive

(Continued)

EBUS	No devices are connected to the GPIB.
ECIC	The interface board is not the Controller-In-Charge; see `SendIFC`.
EDVR	Either `board` is invalid (out of range) or the NI-488.2 driver is not installed.
ENEB	The interface board is not installed or is not properly configured.
EOIP	Asynchronous I/O is in progress

ReceiveSetup

Purpose

Address a device to be a Talker and the interface board to be a Listener in preparation for RcvRespMsg.

Format

C

void ReceiveSetup (short board, short address)

FutureBASIC

```
FN ReceiveSetup (board%,address%)
```

Input

board	The interface board number
address	Address of a device to be talk addressed

Description

ReceiveSetup makes the device described by address talker-active and makes the interface board listen-active. This call is usually followed by a call to RcvRespMsg to transfer data from the device to the interface board. This routine is particularly useful to make multiple calls to RcvRspMsg; it eliminates the need to readdress the device to receive every block of data.

Possible Errors

EARG	The address parameter is invalid (out of range).
EBUS	No devices are connected to the GPIB.
ECIC	The interface board is not the Controller-In-Charge; see SendIFC.
EDVR	Either board is invalid (out of range) or the NI-488.2 driver is not installed.
ENEB	The interface board is not installed or is not properly configured.
EOIP	Asynchronous I/O is in progress.

ResetSys

Purpose

Reset and initialize IEEE 488.2-compliant devices.

Format

C

void `ResetSys` (short `board`, short `addresslist []`)

FutureBASIC

```
FN ResetSys (board%,@addresslist%(0))
```

Input

`board`	The interface board number
`addresslist`	A list of device addresses that is terminated by `NOADDR`

Description

The reset and initialization take place in three steps. The first step resets the GPIB by asserting the Remote Enable (REN) line and then the Interface Clear (IFC) line. The second step clears all of the devices by sending the Universal Device Clear (DCL) GPIB message. The final step causes IEEE 488.2-compliant devices to perform device-specific reset and initialization. This step is accomplished by sending the message `"*RST\n"` to the devices described by `addresslist`.

Possible Errors

EABO	I/O operation is aborted.
EARG	An invalid address (out of range) appears in `addresslist` (`ibcnt` is the index of the invalid address in the `addresslist` array), or the `addresslist` is empty.
EBUS	No devices are connected to the GPIB.
ECIC	The interface board is not the Controller-In-Charge; see `SendIFC`.
EDVR	Either `board` is invalid (out of range) or the NI-488.2 driver is not installed.
ENEB	The interface board is not installed or is not properly configured.

ResetSys

(Continued)

ENOL	No Listeners are on the GPIB.
EOIP	Asynchronous I/O is in progress.
ESAC	Board is not System Controller.

Send

Purpose

Send data bytes to a device.

Format

C

void Send (short board, short address, char data [], long cnt, short eotmode)

FutureBASIC

```
FN gpibSend (board%,address%,data&,cnt&,eotmode%)
```

Input

board	The interface board number
address	Address of a device to which data is sent
data	The data bytes to be sent
cnt	Number of bytes to be sent
eotmode	The data termination mode: DABend, NULLend, or NLend

Description

Send addresses the device described by address to listen and the interface board to talk. Then cnt bytes from data are sent to the device. The last byte is sent with the EOI line asserted if eotmode is DABend. The last byte is sent *without* the EOI line asserted if eotmode is NULLend. If eotmode is NLend then a new line character ('\n') is sent with the EOI line asserted after the last byte of data. The actual number of bytes transferred is returned in the global variable ibcnt.

Possible Errors

EABO	The I/O timeout period has expired before all of the bytes were sent.
EARG	The address parameter is invalid (out of range or the constant NOADDR), or data is empty and the eotmode is DABend.
EBUS	No devices are connected to the GPIB.
ECIC	The interface board is not the Controller-In-Charge; see SendIFC.

Send

(Continued)

EDVR	Either `board` is invalid (out of range) or the NI-488.2 driver is not installed.
ENEB	The interface board is not installed or is not properly configured.
ENOL	No Listeners are on the GPIB to accept the data bytes.
EOIP	Asynchronous I/O is in progress.

SendCmds

Purpose

Send GPIB command bytes.

Format

C

void SendCmds (short board, char commands [], unsigned long cnt)

FutureBASIC

```
FN SendCmds (board%,commands&,cnt&)
```

Input

board	The interface board number
commands	Command bytes to be sent
cnt	Number of bytes to be sent

Description

SendCmds sends cnt command bytes from commands over the GPIB as command bytes (interface messages). The number of command bytes transferred is returned in the global variable ibcnt. Refer to Appendix A, *Multiline Interface Messages*, for a listing of the defined interface messages.

Use command bytes to configure the state of the GPIB, not to send instructions to GPIB devices. Use Send or SendList to send device-specific instructions.

Possible Errors

EABO	The I/O timeout period expired before all of the command bytes were sent.
ECIC	The interface board is not the Controller-In-Charge; see SendIFC.
EDVR	Either board is invalid (out of range) or the NI-488.2 driver is not installed.
ENEB	The interface board is not installed or is not properly configured.
ENOL	No devices are connected to the GPIB.
EOIP	Asynchronous I/O is in progress.

SendDataBytes

Purpose

Send data bytes to devices that are already addressed to listen.

Format

C

void SendDataBytes (short board, char data [], long cnt, short eotmode)

FutureBASIC

```
FN SendDataBytes (board%,data&,cnt&,eotmode%)
```

Input

board	The interface board number
data	The data bytes to be sent
cnt	Number of bytes to be sent
eotmode	The data termination mode: DABend, NULLend, or NLend

Description

SendDataBytes sends cnt number of bytes from the buffer to devices which are already addressed to listen. The last byte is sent with the EOI line asserted if eotmode is DABend; the last byte is sent *without* the EOI line asserted if eotmode is NULLend. If eotmode is NLend then a new line character ('\n') is sent with the EOI line asserted after the last byte. The actual number of bytes transferred is returned in the global variable ibcnt.

SendDataBytes assumes that the interface board is in talk-active state and that devices are already addressed as Listeners on the GPIB (see SendSetup, Send, or SendList).

Possible Errors

EABO	The I/O timeout period expired before all of the bytes were sent.
EADR	The interface board is not talk-active; use SendSetup to address the GPIB properly.
EARG	The eotmode parameter is invalid (it can be only DABend, NULLend, or NLend), or data is empty and the eotmode is DABend.
ECIC	The interface board is not the Controller-In-Charge; see SendIFC.

SendDataBytes

(Continued)

EDVR	Either `board` is invalid (out of range) or the NI-488.2 driver is not installed.
ENEB	The interface board is not installed or is not properly configured.
ENOL	No Listeners are on the GPIB to accept the data bytes; use `SendSetup` to address the GPIB properly.
EOIP	Asynchronous I/O is in progress.

SendIFC

Purpose

Reset the GPIB by sending interface clear.

Format

C

void `SendIFC` (short `board`)

FutureBASIC

```
FN SendIFC (board%)
```

Input

`board`	The interface board number

Description

`SendIFC` is used as part of GPIB initialization. It forces the interface board to be Controller-In-Charge of the GPIB. It also ensures that the connected devices are all unaddressed and that the interface functions of the devices are in their idle states.

Possible Errors

EDVR	Either `board` is invalid (out of range) or the NI-488.2 driver is not installed.
ENEB	The interface board is not installed or is not properly configured.
EOIP	Asynchronous I/O is in progress.
ESAC	The interface board is not configured as the System Controller; see `ibrsc`.

SendList

Purpose

Send data bytes to multiple GPIB devices.

Format

C

void SendList (short board, short addresslist [], char data [], long cnt,
short eotmode)

FutureBASIC

```
FN SendList (board%,@addresslist%(0),data&,cnt&,eotmode%)
```

Input

board	The interface board number
addresslist	A list of device addresses to send data to
data	The data bytes to be sent
cnt	Number of bytes transmitted
eotmode	The data termination mode: DABend, NULLend, or NLend.

Description

SendList addresses the devices described by addresslist to listen and the interface board to talk. Then, cnt bytes from buffer are sent to the devices. The last byte is sent with the EOI line asserted if eotmode is DABend. The last byte is sent *without* the EOI line asserted if eotmode is NULLend. If eotmode is NLend, then a new line character ('\n') is sent with the EOI line asserted after the last byte. The actual number of bytes transferred is returned in the global variable ibcnt.

Possible Errors

EABO	The I/O timeout period expired before all of the bytes were sent.
EARG	An invalid address (out of range) appears in addresslist (ibcnt is the index of the invalid address in the addresslist array), the eotmode parameter is invalid (eotmode can be only DABend, NULLend, or NLend), or data is empty and the eotmode is DABend.
EBUS	No devices are connected to the GPIB.

SendList

(Continued)

ECIC	The interface board is not the Controller-In-Charge; see `SendIFC`.
EDVR	Either `board` is invalid (out of range) or the NI-488.2 driver is not installed.
ENEB	The interface board is not installed or is not properly configured.
EOIP	Asynchronous I/O is in progress.

SendLLO

Purpose

Send the Local Lockout (LLO) message to all devices.

Format

C

void `SendLLO` (short `board`)

FutureBASIC

```
FN SendLLO (board%)
```

Input

`board`	The interface board number

Description

`SendLLO` sends the GPIB Local Lockout (LLO) message to all devices. While Local Lockout is in effect, only the Controller-In-Charge can alter the state of the devices by sending appropriate GPIB messages. `SendLLO` is reserved for use in unusual local/remote situations. In most cases, use `SetRWLS` to place devices in Remote With Lockout State.

Possible Errors

EBUS	No devices are connected to the GPIB.
ECIC	The interface board is not the Controller-In-Charge; see `SendIFC`.
EDVR	Either `board` is invalid (out of range) or the NI-488.2 driver is not installed.
ENEB	The interface board is not installed or is not properly configured.
EOIP	Asynchronous I/O is in progress.
ESAC	The interface board is not configured as System Controller.

SendSetup

Purpose

Set up devices to receive data in preparation for SendDataBytes.

Format

C

void SendSetup (short board, short addresslist [])

FutureBASIC

```
FN SendSetup (board%,@addresslist%(0))
```

Input

board	The interface board number
addresslist	A list of device addresses that is terminated by NOADDR

Description

SendSetup makes the devices described by addresslist listen-active and makes the interface board talk-active. This call is usually followed by SendDataBytes to actually transfer data from the interface board to the devices. SendSetup is particularly useful to set up the addressing before making multiple calls to SendDataBytes; it eliminates the need to readdress the devices for every block of data.

Possible Errors

EARG	The addresslist is empty, or an invalid address (out of range) appears in addresslist; ibcnt is the index of the invalid address in the addresslist array.
EBUS	No devices are connected to the GPIB.
ECIC	The interface board is not the Controller-In-Charge; see SendIFC.
EDVR	Either board is invalid (out of range) or the NI-488.2 driver is not installed.
ENEB	The interface board is not installed or is not properly configured.
EOIP	Asynchronous I/O is in progress.

SetRWLS

Purpose

Place devices in Remote With Lockout State.

Format

C

void SetRWLS (short board, short addresslist [])

FutureBASIC

```
FN SetRWLS (board%,@addresslist%(0))
```

Input

board	The interface board number
addresslist	A list of device addresses terminated by NOADDR

Description

SetRWLS places the devices described by addresslist in remote mode by asserting the Remote Enable (REN) GPIB line. Then those devices are placed in lockout state by the Local Lockout (LLO) GPIB message. You cannot program those devices locally until the Controller-In-Charge releases the Local Lockout. To release the Local Lockout, use the EnableLocal NI-488.2 routine.

Possible Errors

EARG	An invalid address (out of range) appears in addresslist (ibcnt is the index of the invalid address in the addresslist array), or the addresslist is empty.
EBUS	No devices are connected to the GPIB.
ECIC	The interface board is not the Controller-In-Charge; see SendIFC.
EDVR	Either board is invalid (out of range) or the NI-488.2 driver is not installed.
ENEB	The interface board is not installed or is not properly configured.
EOIP	Asynchronous I/O is in progress.
ESAC	The interface board is not configured as System Controller.

TestSRQ

Purpose

Determine the current state of the GPIB Service Request (SRQ) line.

Format

C

void `TestSRQ` (short `board`, short `*result`)

FutureBASIC

```
FN TestSRQ (board%,result&)
```

Input

`board`	The interface board number

Output

`result`	State of the SRQ line: non-zero if the line is asserted, zero if the line is not asserted

Description

`TestSRQ` returns the current state of the GPIB SRQ line in `result`. If SRQ is asserted, then `result` contains a non-zero value. Otherwise, `result` contains a zero. Use `TestSRQ` to get the current state of the GPIB SRQ line. Use `WaitSRQ` to wait until SRQ is asserted.

Possible Errors

EDVR	Either `board` is invalid (out of range) or the NI-488.2 driver is not installed.
ENEB	The interface board is not installed or is not properly configured.

TestSys

Purpose

Cause IEEE 488.2-compliant devices to conduct self tests.

Format

C

void TestSys (short board, short addresslist [], short resultlist [])

FutureBASIC

```
FN TestSys (board%,@addresslist%(0),@resultlist%(0))
```

Input

board	The interface board number
addresslist	A list of device addresses terminated by NOADDR

Output

resultlist	A list of test results; each entry corresponds to an address in addresslist

Description

TestSys sends the "*TST\n" message to the IEEE 488.2-compliant devices described by addresslist. The "*TST\n" message instructs them to conduct their self-test procedures. A 16-bit test result code is read from each device and stored in resultlist. A test result of 0\n indicates that the device passed its self test. Any other value indicates that the device failed its self test. Refer to the manual that came with your device to determine the meaning of the failure code. A test result of –1 indicates that the I/O timeout period elapsed before the device sent its result code. ibcnt contains the number of devices that failed.

TestSys

(Continued)

Possible Errors

EABO	The interface board timed out before receiving a result from a device; `ibcnt` contains the index of the timed-out device. –1 is stored as the test result for the timed-out device.
EARG	An invalid address (out of range) appears in `addresslist` (`ibcnt` is the index of the invalid address in the `addresslist` array), or the `addresslist` is empty.
EBUS	No devices are connected to the GPIB.
ECIC	The interface board is not the Controller-In-Charge; see `SendIFC`.
EDVR	Either `board` is invalid (out of range) or the NI-488.2 driver is not installed.
ENEB	The interface board is not installed or is not properly configured.
ENOL	No Listeners are on the GPIB.
EOIP	Asynchronous I/O is in progress.

Trigger

Purpose

Trigger a device.

Format

C

void `Trigger` (short `board`, short `address`)

FutureBASIC

```
FN Trigger (board%,address%)
```

Input

`board`	The interface board number
`address`	Address of a device to be triggered

Description

`Trigger` sends the Group Execute Trigger (GET) GPIB message to the device described by `address`. If `address` is the constant `NOADDR`, the Group Execute Trigger message is sent to all devices that are currently listen-active on the GPIB.

Possible Errors

EARG	The `address` parameter is invalid (out of range).
EBUS	No devices are connected to the GPIB.
ECIC	The interface board is not the Controller-In-Charge; see `SendIFC`.
EDVR	Either `board` is invalid (out of range) or the NI-488.2 driver is not installed.
ENEB	The interface board is not installed or is not properly configured.
EOIP	Asynchronous I/O is in progress.

TriggerList

Purpose

Trigger multiple devices.

Format

C

void `TriggerList` (short `board`, short `addresslist []`)

FutureBASIC

```
FN TriggerList (board%,@addresslist%(0))
```

Input

`board`	The interface board number
`addresslist`	A list of device addresses terminated by `NOADDR`

Description

`TriggerList` sends the Group Execute Trigger (GET) GPIB message to the devices included in `addresslist`. If `addresslist` contains only `NOADDR`, the Group Execute Trigger message is sent to all devices that are currently listen-active on the GPIB.

Possible Errors

EARG	An invalid address (out of range) appears in `addresslist`; `ibcnt` is the index of the invalid address in the `addresslist` array.
EBUS	No devices are connected to the GPIB.
ECIC	The interface board is not the Controller-In-Charge; see `SendIFC`.
EDVR	Either `board` is invalid (out of range) or the NI-488.2 driver is not installed.
ENEB	The interface board is not installed or is not properly configured.
EOIP	Asynchronous I/O is in progress.

WaitSRQ

Purpose

Wait until a device asserts the GPIB Service Request (SRQ) line.

Format

C

void `WaitSRQ` (short `board`, short `*result`)

FutureBASIC

```
FN WaitSRQ (board%,result&)
```

Input

`board`	The interface board number

Output

`result`	State of the SRQ line: non-zero if line is asserted, zero if line not asserted

Description

`WaitSRQ` waits until either the GPIB SRQ line is asserted or the timeout period has expired (see `ibtmo`). When `WaitSRQ` returns, `result` contains a non-zero value if SRQ is asserted. Otherwise, `result` contains a zero. Use `TestSRQ` to get the current state of the GPIB SRQ line. Use `WaitSRQ` to wait until SRQ is asserted.

Possible Errors

EDVR	Either `board` is invalid (out of range) or the NI-488.2 driver is not installed.
ENEB	The interface board is not installed or is not properly configured.

Multiline Interface Messages

Appendix
A

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	CFE	3F	077	63	?	UNL

Message Definitions

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

* This multiline interface message is a proposed extension to the IEEE 488.1 specification to support the HS488 high-speed protocol.

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,CFG1
42	102	66	B	MTA2	62	142	98	b	MSA2,PPE, CFG2
43	103	67	C	MTA3	63	143	99	c	MSA3,PPE,CFG3
44	104	68	D	MTA4	64	144	100	d	MSA4,PPE,CFG4
45	105	69	E	MTA5	65	145	101	e	MSA5,PPE,CFG5
46	106	70	F	MTA6	66	146	102	f	MSA6,PPE,CFG6
47	107	71	G	MTA7	67	147	103	g	MSA7,PPE,CFG7
48	110	72	H	MTA8	68	150	104	h	MSA8,PPE,CFG8
49	111	73	I	MTA9	69	151	105	i	MSA9,PPE,CFG9
4A	112	74	J	MTA10	6A	152	106	j	MSA10,PPE,CFG10
4B	113	75	K	MTA11	6B	153	107	k	MSA11,PPE,CFG11
4C	114	76	L	MTA12	6C	154	108	l	MSA12,PPE,CFG12
4D	115	77	M	MTA13	6D	155	109	m	MSA13,PPE,CFG13
4E	116	78	N	MTA14	6E	156	110	n	MSA14,PPE,CFG14
4F	117	79	O	MTA15	6F	157	111	o	MSA15,PPE,CFG15
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	

Message Definitions

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

Status Word Conditions

Appendix
B

This appendix gives a detailed description of the conditions reported in the status word, `ibsta`.

For information about how to use `ibsta` in your application program, refer to Chapter 2, *Developing Your Application*, in the *NI-488.2 User Manual for MacOS.*

If a function call returns an ENEB or EDVR error, all status word bits except the ERR bit are cleared, indicating that it is not possible to obtain the status of the GPIB board.

Each bit in `ibsta` can be set for NI-488 device calls (dev), NI-488 board calls and NI-488.2 calls (brd), or both (dev, brd).

The following table lists the status word bits.

Table B-1. 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 interrupt received
RQS	11	800	dev	Device requesting service
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

Table B-1. Status Word Bits (Continued)

Mnemonic	Bit Pos.	Hex Value	Type	Description
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`. Appendix C, *Error Codes and Solutions*, 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`, `ibwrt`, `Receive`, `Send`, and `SendCmds`) 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. If the GPIB board is performing a shadow handshake as a result of the `ibgts` function, any other function can return a status word with the END bit set if the END condition occurs before or during that call. 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 operationracter. 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 board is CIC, the GPIB SRQ line is asserted, and the automatic serial poll capability is disabled. SRQI is cleared either when the GPIB board ceases to be the CIC or when the GPIB SRQ line is unasserted.

RQS (dev)

RQS appears in the status word only after a device-level call and indicates that the device is requesting service. RQS is set whenever bit 6 is asserted in the serial poll status byte of the device. The serial poll that obtains the status byte can be the result of a call to `ibrsp`, or the poll might be automatic if automatic serial polling is enabled. Do not issue an `ibwait` on RQS for a device that does not respond to serial polls. RQS is cleared when an `ibrsp` reads the serial poll status byte that caused the RQS.

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 board is in a lockout state. While LOK is set, the `EnableLocal` routine or `ibloc` function is inoperative for that board. LOK is set whenever the GPIB board detects that the Local Lockout (LLO) message has been sent either by the GPIB board or by another Controller. LOK is cleared when the System Controller unasserts the Remote Enable (REN) GPIB line.

REM (brd)

REM indicates whether or not the board is in the remote state. REM is set whenever the Remote Enable (REN) GPIB line is asserted and the GPIB board detects that its listen address has been sent either by the GPIB board or by another Controller. REM is cleared in the following situations:

- When REN becomes unasserted
- When the GPIB board as a Listener detects that the Go to Local (GTL) command has been sent either by the GPIB board or by another Controller
- When the `ibloc` function is called while the LOK bit is cleared in the status word

CIC (brd)

CIC indicates whether the GPIB board is the Controller-In-Charge. CIC is set when the `SendIFC` routine or `ibsic` function is executed while the GPIB board is System Controller or when another Controller passes control to the GPIB board. CIC is cleared whenever the GPIB board detects Interface Clear (IFC) from the System Controller, or when the GPIB board 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 board is addressed as a Talker. TACS is set whenever the GPIB board detects that its talk address (and secondary address, if enabled) has been sent either by the GPIB board itself or by another Controller. TACS is cleared whenever the GPIB board 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 board is addressed as a Listener. LACS is set whenever the GPIB board detects that its listen address (and secondary address, if enabled) has been sent either by the GPIB board itself or by another Controller. LACS is also set whenever the GPIB board shadow handshakes as a result of the `ibgts` function. LACS is cleared whenever the GPIB board detects the Unlisten (UNL) command, its own talk address, Interface Clear (IFC), or that the `ibgts` function has been called without shadow handshake.

DTAS (brd)

DTAS indicates whether the GPIB board has detected a device trigger command. DTAS is set whenever the GPIB board, 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.

DCAS (brd)

DCAS indicates whether the GPIB board has detected a device clear command. DCAS is set whenever the GPIB board detects that the Device Clear (DCL) command has been sent by another Controller, or whenever the GPIB board 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.

Error Codes and Solutions

Appendix
C

This appendix lists a description of each error, some conditions under which it might occur, and possible solutions.

The following table lists the GPIB error codes.

Table C-1. GPIB Error Codes

Error Mnemonic	iberr Value	Meaning
EDVR	0	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 (timeout)
ENEB	7	Nonexistent GPIB board
EDMA	8	No DMA channel available
EOIP	10	Asynchronous I/O in progress
ECAP	11	No capability for operation
EFSO	12	File system error
EBUS	14	GPIB bus error
ESTB	15	Serial poll status byte queue overflow

Table C-1. GPIB Error Codes (Continued)

Error Mnemonic	iberr Value	Meaning
ESRQ	16	SRQ stuck in ON position
ETAB	20	Table problem
ELCK	21	Board or device is locked

EDVR (0)

EDVR is returned when the board or device name passed to `ibfind` is not configured in the software.

EDVR is also returned when an invalid unit descriptor is passed to any function call.

EDVR is also returned when the driver is not installed. In this case, `ibcnt` contains a system level error code.

Solutions

- Use `ibdev` to open a device without specifying its symbolic name.
- Use only device or board names that are configured in the utility program `NI-488 Config` as parameters in the `ibfind` function.
- Use the unit descriptor returned from the `ibfind` function as the first parameter in subsequent NI-488 functions. Examine the variable after the `ibfind` and before the failing function to make sure it was not corrupted.
- Make sure the NI-488.2 driver is installed by checking to see if `NI-488 INIT` is in the `Extensions` folder in the `System Folder`.

ECIC (1)

ECIC is returned when one of the following board functions or routines is called while the board is not CIC:

- Any device-level NI-488 functions that affect the GPIB
- Any board-level NI-488 functions that issue GPIB command bytes such as `ibcmd`, `ibcmda`, `ibln`, `ibrpp`

- ibcac, ibgts
- Any of the NI-488.2 routines that issue GPIB command bytes such as SendCmds, PPoll, Send, Receive

Solutions

- Use ibsic or SendIFC to make the GPIB board become CIC on the GPIB.
- Use ibrsc 1 to make sure your GPIB board 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. For a device write, this error indicates that the GPIB address configured for that device in the software does not match the GPIB address of any device connected to the bus, that the GPIB cable is not connected to the device, or that the device is not powered on.

ENOL can occur in situations in which the GPIB board is not the CIC and the Controller asserts ATN before the write call in progress has ended.

Solutions

- Make sure that the GPIB address of your device matches the GPIB address of the device to which you want to write data.
- 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.
- Call ibpad (or ibsad, if necessary) to match the configured address to the device switch settings.
- Reduce the write byte count to that which is expected by the Controller.

EADR (3)

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

EADR is also returned by the function `ibgts` when the shadow-handshake feature is requested and the GPIB ATN line is already unasserted. In this case, the shadow handshake is not possible and the error is returned to notify you of that fact.

Solutions

- Make sure that the GPIB board is addressed correctly before calling `ibrd`, `ibwrt`, `RcvRespMsg`, or `SendDataBytes`.
- Avoid calling `ibgts` except immediately after an `ibcmd` call. (`ibcmd` causes ATN to be asserted.)

EARG (4)

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

- `ibtmo` called with a value not in the range 0 through 17
- `ibpad` or `ibsad` called with invalid addresses
- `ibppc` called with invalid parallel poll configurations
- A board-level NI-488 call made with a valid device descriptor or a device-level NI-488 call made with a board descriptor
- An NI-488.2 routine called with an invalid address
- `PPollConfig` called with an invalid data line or sense bit

Solutions

- Make sure that the parameters passed to the NI-488 function or NI-488.2 routine are valid.
- Do not use a device descriptor in a board function or vice-versa.

ESAC (5)

ESAC results when `ibsic`, `ibsre`, `SendIFC`, or `EnableRemote` is called when the GPIB board does not have System Controller capability.

Solutions

Give the GPIB board System Controller capability by calling `ibrsc 1` or by using `NI-488 Config` to configure that capability into the software.

EABO (6)

EABO indicates that an I/O operation has been canceled, usually due to a timeout condition. Other causes for this error are calling `ibstop` or receiving 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 `ibtmo`.
- Make sure that you have configured your device to send data before you request data.

ENEB (7)

ENEB occurs when there is no GPIB board present. This happens when the board is not physically plugged into the system, or there is a conflict in the system.

Solutions

Verify that all GPIB interfaces and external controller boxes are plugged in securely, powered on, and configured properly in the GPIB configuration.

EDMA (8)

EDMA occurs when the driver is unable to allocate a DMA channel.

Solutions

Verify that other boards are not using all seven available DMA channels. Disconnect the RTSI connector from the other DMA boards temporarily.

EOIP (10)

EOIP occurs when an asynchronous I/O operation has not finished before some other call is made. During asynchronous I/O, you can only use `ibstop`, `ibwait`, and `ibonl`, or perform other non-GPIB operations. Once the asynchronous I/O has begun, further GPIB calls other than `ibstop`, `ibwait`, or `ibonl` are strictly limited. If a call might interfere with the I/O operation in progress, the driver returns EOIP.

Solutions

Resynchronize the driver and the application before making any further GPIB calls. Resynchronization is accomplished by using one of the following three functions:

- `ibwait` If the returned `ibsta` contains CMPL then the driver and application are resynchronized.
- `ibstop` The I/O is canceled; the driver and application are resynchronized.
- `ibonl` The I/O is canceled and the interface is reset; the driver and application are resynchronized.

ECAP (11)

ECAP results when your GPIB board 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 your GPIB interface board and the driver both have the needed capability.

EFSO (12)

EFSO results when an `ibrdf` or `ibwrtf` call encounters a problem performing a file operation. Specifically, this error indicates that the function is unable to open, create, seek, write, or close the file being accessed. The specific system error code for this condition is contained in `ibcnt`.

Solutions

- Make sure the file is in the same folder as your application.
- Make sure there is enough room on the disk to hold the file.

EBUS (14)

EBUS results when certain GPIB bus errors occur during device 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 `ibtmo` function. 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 for the driver to send command bytes, increase the timeout period.

ESTB (15)

ESTB is reported only by the `ibrsp` function. ESTB indicates that one or more serial poll status bytes received from automatic serial polls have been discarded because of a lack of storage space. Several older status bytes are available; however, the oldest is being returned by the `ibrsp` call.

Solutions

- Call `ibrsp` more frequently to empty the queue.
- Disable autopolling with the `ibconfig` function or the `NI-488 Config` utility.

ESRQ (16)

ESRQ occurs only during the `ibwait` function or the `WaitSRQ` routine. ESRQ indicates that a wait for RQS is not possible because the GPIB SRQ line is stuck on. This situation can be caused by the following events:

- Usually, a device unknown to the software is asserting SRQ. Because the software does not know of this device, it can never serial poll the device and unassert SRQ.
- A GPIB bus tester or similar equipment might be forcing the SRQ line to be asserted.
- A cable problem might exist involving the SRQ line.

Although the occurrence of ESRQ warns you of a definite GPIB problem, it does not affect GPIB operations, except that you cannot depend on the RQS bit while the condition lasts.

Solutions

Check to see if other devices not used by your application are asserting SRQ. Disconnect them from the GPIB if necessary.

ETAB (20)

ETAB occurs only during the `FindLstn`, `FindRQS`, and `ibevent` functions. ETAB indicates that there was some problem with a table used by these functions.

- In the case of `FindLstn`, ETAB means that the given table did not have enough room to hold all the addresses of the Listeners found.
- In the case of `FindRQS`, ETAB means that none of the devices in the given table were requesting service.
- In the case of `ibevent`, ETAB means the event queue overflowed and event information was lost.

Solutions

In the case of `FindLstn`, increase the size of result arrays. In the case of `FindRQS`, check to see if other devices not used by your application are asserting SRQ. Disconnect them from the GPIB if necessary. In the case of ETAB returned from `ibevent`, call `ibevent` more often to empty the queue.

ELCK (21)

ELCK occurs if the requested GPIB-ENET board or device is being used through another connection.

Solutions

Wait for the lock on the board or device to be released, or try using `ibunlock` if you previously used `iblock` to lock access to the connection.

Customer Communication

Appendix D

For your convenience, this appendix contains forms to help you gather the information necessary to help us solve your technical problems and a form you can use to comment on the product documentation. When you contact us, we need the information on the Technical Support Form and the configuration form, if your manual contains one, about your system configuration to answer your questions as quickly as possible.

National Instruments has technical assistance through electronic, fax, and telephone systems to quickly provide the information you need. Our electronic services include a bulletin board service, an FTP site, a Fax-on-Demand system, and e-mail support. If you have a hardware or software problem, first try the electronic support systems. If the information available on these systems does not answer your questions, we offer fax and telephone support through our technical support centers, which are staffed by applications engineers.

Electronic Services

Bulletin Board Support

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

United States: (512) 794-5422
Up to 14,400 baud, 8 data bits, 1 stop bit, no parity

United Kingdom: 01635 551422
Up to 9,600 baud, 8 data bits, 1 stop bit, no parity

France: 01 48 65 15 59
Up to 9,600 baud, 8 data bits, 1 stop bit, no parity

FTP Support

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

Fax-on-Demand Support

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

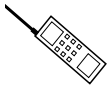
E-Mail Support (currently U.S. only)

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

`support@natinst.com`

Telephone and Fax Support

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



	Telephone	Fax
Australia	03 9879 5166	03 9879 6277
Austria	0662 45 79 90 0	0662 45 79 90 19
Belgium	02 757 00 20	02 757 03 11
Canada (Ontario)	905 785 0085	905 785 0086
Canada (Quebec)	514 694 8521	514 694 4399
Denmark	45 76 26 00	45 76 26 02
Finland	09 725 725 11	09 725 725 55
France	01 48 14 24 24	01 48 14 24 14
Germany	089 741 31 30	089 714 60 35
Hong Kong	2645 3186	2686 8505
Israel	03 5734815	03 5734816
Italy	02 413091	02 41309215
Japan	03 5472 2970	03 5472 2977
Korea	02 596 7456	02 596 7455
Mexico	5 520 2635	5 520 3282
Netherlands	0348 433466	0348 430673
Norway	32 84 84 00	32 84 86 00
Singapore	2265886	2265887
Spain	91 640 0085	91 640 0533
Sweden	08 730 49 70	08 730 43 70
Switzerland	056 200 51 51	056 200 51 55
Taiwan	02 377 1200	02 737 4644
United States	512 795 8248	512 794 5678
United Kingdom	01635 523545	01635 523154

Technical Support Form

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

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

Name _____

Company _____

Address _____

_____

Fax (___)_____ Phone (___) _____

Computer brand _____ Model _____ Processor_____

Operating system (include version number) _____

Clock speed _____MHz RAM _____MB Display adapter _____

Mouse ___yes ___no Other adapters installed _____

Hard disk capacity _____MB Brand _____

Instruments used _____

_____

National Instruments hardware product model _____ Revision _____

Configuration _____

National Instruments software product _____ Version _____

Configuration _____

The problem is: _____

_____

_____

_____

_____

List any error messages: _____

_____

_____

The following steps reproduce the problem:_____

_____

_____

_____

_____

_____

Documentation Comment Form

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

Title: *NI-488.2™ Function Reference Manual for MacOS*

Edition Date: July 1997

Part Number: 320898B-01

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

_____

_____

_____

_____

_____

_____

_____

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

_____

_____

_____

_____

_____

_____

_____

Thank you for your help.

Name _____

Title _____

Company _____

Address _____

_____

Phone (___)_____ Fax (___) _____

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

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

Glossary

Prefix	Meanings	Value
n-	nano-	10^{-9}
µ-	micro-	10^{-6}
m-	milli-	10^{-3}
k-	kilo-	10^{3}
M-	mega-	10^{6}

A

acceptor handshake — Listeners use this GPIB interface function to receive data, and all devices use it to receive commands. *See* source handshake and handshake.

access board — The GPIB board that controls and communicates with the devices on the bus that are attached to it.

ANSI — American National Standards Institute.

ASCII — American Standard Code for Information Interchange.

asynchronous — An action or event that occurs at an unpredictable time with respect to the execution of a program.

automatic serial polling (autopolling) — A feature of the NI-488.2 software in which serial polls are executed automatically by the driver whenever a device asserts the GPIB SRQ line.

B

board-level function — A rudimentary function that performs a single operation.

C

CFE	Configuration Enable is the GPIB command which precedes CFGn and is used to place devices into their configuration mode.
CFGn	These GPIB commands (CFG1 through CFG15) follow CFE and are used to configure all devices for the number of meters of cable in the system so that HS488 transfers occur without errors.
CIC	*See* Controller-In-Charge.
Controller-In-Charge (CIC)	The device that manages the GPIB by sending interface messages to other devices.
CPU	Central processing unit.

D

DAV (Data Valid)	One of the three GPIB handshake lines. *See* handshake.
DCL	Device Clear is the GPIB command used to reset the device or internal functions of all devices. *See* SDC.
Device Clear	*See* DCL.
device-level function	A function that combines several rudimentary board operations into one function so that the user does not have to be concerned with bus management or other GPIB protocol matters.
DIO1 through DIO8	The GPIB lines that are used to transmit command or data bytes from one device to another.
DLL	Dynamic link library.
DMA (direct memory access)	High-speed data transfer between the GPIB board and memory that is not handled directly by the CPU. Not available on some systems. *See* programmed I/O.
driver	Device driver software installed within the operating system.

E

END or END message	A message that signals the end of a data string. END is sent by asserting the GPIB End or Identify (EOI) line with the last data byte.
EOI	A GPIB line that is used to signal either the last byte of a data message (END) or the parallel poll Identify (IDY) message.
EOS or EOS byte	A 7- or 8-bit end-of-string character that is sent as the last byte of a data message.
EOT	End of transmission.
ESB	The Event Status bit is part of the IEEE 488.2-defined status byte which is received from a device responding to a serial poll.

G

GET	Group Execute Trigger is the GPIB command used to trigger a device or internal function of an addressed Listener.
Go To Local	*See* GTL.
GPIB	General Purpose Interface Bus is the common name for the communications interface system defined in ANSI/IEEE Standard 488.1-1987 and ANSI/IEEE Standard 488.2-1987.
GPIB address	The address of a device on the GPIB, composed of a primary address (MLA and MTA) and an optional secondary address (MSA). The GPIB board has both a GPIB address and an I/O address.
GPIB board	Refers to the National Instruments family of GPIB interface boards.
Group Executed Trigger	*See* GET.
GTL	Go To Local is the GPIB command used to place an addressed Listener in local (front panel) control mode.

H

handshake	The mechanism used to transfer bytes from the Source Handshake function of one device to the Acceptor Handshake function of another device. The three GPIB lines DAV, NRFD, and NDAC are used in an

	interlocked fashion to signal the phases of the transfer, so that bytes can be sent asynchronously (for example, without a clock) at the speed of the slowest device. For more information about handshaking, refer to the ANSI/IEEE Standard 488.1-1987.
hex	Hexadecimal; a number represented in base 16, for example decimal 16 = hex 10.
high-level function	*See* device-level function.
Hz	Hertz.

I

I/O (Input/Output)	In the context of this manual, the transmission of commands or messages between the computer via the GPIB board and other devices on the GPIB.
I/O address	The address of the GPIB board from the point of view of the CPU, as opposed to the GPIB address of the GPIB board. Also called port address or board address.
`ibcnt`	After each NI-488.2 I/O function, this global variable contains the actual number of bytes transmitted.
`iberr`	A global variable that contains the specific error code associated with a function call that failed.
`IBIC 488.2`	`IBIC 488.2`, the Interface Bus Interactive Control utility, is used to communicate with GPIB devices, troubleshoot problems, and develop your application.
`ibsta`	At the end of each function call, this global variable (status word) contains status information.
IEEE	Institute of Electrical and Electronic Engineers.
interface message	A broadcast message sent from the Controller to all devices and used to manage the GPIB.
`ist`	An Individual Status bit of the status byte used in the Parallel Poll Configure function.

K

KB	Kilobytes.

L

LAD (Listen Address)	*See* MLA.
language interface	Code that enables an application program that uses NI-488 functions or NI-488.2 routines to access the driver.
listen address	*See* MLA.
Listener	A GPIB device that receives data messages from a Talker.
low-level function	*See* board-level function.

M

m	Meters.
MAV	The Message Available bit is part of the IEEE 488.2-defined status byte which is received from a device responding to a serial poll.
MB	Megabytes of memory.
memory-resident	Resident in RAM.
MLA (My Listen Address)	A GPIB command used to address a device to be a Listener. It can be any one of the 31 primary addresses.
MSA (My Secondary Address)	My Secondary Address is the GPIB command used to address a device to be a Listener or a Talker when extended (two byte) addressing is used. The complete address is a MLA or MTA address followed by an MSA address. There are 31 secondary addresses for a total of 961 distinct listen or talk addresses for devices.
MTA (My Talk Address)	A GPIB command used to address a device to be a Talker. It can be any one of the 31 primary addresses.
multitasking	The concurrent processing of more than one program or task.

N

NDAC (Not Data Accepted)	One of the three GPIB handshake lines. *See* handshake.
`NI-488 Config`	The NI-488.2 driver configuration control panel utility.
NRFD (Not Ready For Data)	One of the three GPIB handshake lines. *See* handshake.

P

parallel poll	The process of polling all configured devices at once and reading a composite poll response. *See* serial poll.
PIO	*See* programmed I/O.
PPC (Parallel Poll Configure)	Parallel Poll Configure is the GPIB command used to configure an addressed Listener to participate in polls.
PPD (Parallel Poll Disable)	Parallel Poll Disable is the GPIB command used to disable a configured device from participating in polls. There are 16 PPD commands.
PPE (Parallel Poll Enable)	Parallel Poll Enable is the GPIB command used to enable a configured device to participate in polls and to assign a DIO response line. There are 16 PPE commands.
PPU (Parallel Poll Unconfigure)	Parallel Poll Unconfigure is the GPIB command used to disable any device from participating in polls.
programmed I/O	Low-speed data transfer between the GPIB board and memory in which the CPU moves each data byte according to program instructions. *See* DMA.

R

RAM	Random-access memory.
resynchronize	The NI-488.2 software and the user application must resynchronize after asynchronous I/O operations have completed.
RQS	Request Service.

S

s	Seconds.
SDC	Selected Device Clear is the GPIB command used to reset internal or device functions of an addressed Listener. *See* DCL and IFC.
serial poll	The process of polling and reading the status byte of one device at a time. *See* parallel poll.
Service Request	*See* SRQ.
source handshake	The GPIB interface function that transmits data and commands. Talkers use this function to send data, and the Controller uses it to send commands. *See* acceptor handshake and handshake.
SPD (Serial Poll Disable)	Serial Poll Disable is the GPIB command used to cancel an SPE command.
SPE (Serial Poll Enable)	Serial Poll Enable is the GPIB command used to enable a specific device to be polled. That device must also be addressed to talk. *See* SPD.
SRQ (Service Request)	The GPIB line that a device asserts to notify the CIC that the device needs servicing.
status byte	The IEEE 488.2-defined data byte sent by a device when it is serially polled.
status word	*See* ibsta.
synchronous	Refers to the relationship between the NI-488.2 driver functions and a process when executing driver functions is predictable; the process is blocked until the driver completes the function.
System Controller	The single designated Controller that can assert control (become CIC of the GPIB) by sending the Interface Clear (IFC) message. Other devices can become CIC only by having control passed to them.

T

TAD (Talk Address)	*See* MTA.
Talker	A GPIB device that sends data messages to Listeners.

TCT	Take Control is the GPIB command used to pass control of the bus from the current Controller to an addressed Talker.
timeout	A feature of the NI-488.2 driver that prevents I/O functions from hanging indefinitely when there is a problem on the GPIB.
TLC	An integrated circuit that implements most of the GPIB Talker, Listener, and Controller functions in hardware.

U

ud (unit descriptor)	A variable name and first argument of each function call that contains the unit descriptor of the GPIB interface board or other GPIB device that is the object of the function.
UNL	Unlisten is the GPIB command used to unaddress any active Listeners.
UNT	Untalk is the GPIB command used to unaddress an active Talker.

Index

E

F

G

I

L

M

N

O

P

R

S

T

U

W