

Universal Language Interface Using HP-Style Calls

December 1993 Edition

Part Number 370915A-01

**© Copyright 1989, 1994 National Instruments Corporation.
All Rights Reserved.**

National Instruments Corporate Headquarters

6504 Bridge Point Parkway

Austin, TX 78730-5039

(512) 794-0100

Technical support fax: (800) 328-2203

(512) 794-5678

Branch Offices:

Australia (03) 879 9422, Austria (0662) 435986, Belgium 02/757.00.20,

Canada (Ontario) (519) 622-9310, Canada (Québec) (514) 694-8521,

Denmark 45 76 26 00, Finland (90) 527 2321, France (1) 48 14 24 24,

Germany 089/741 31 30, Italy 02/48301892, Japan (03) 3788-1921,

Netherlands 03480-33466, Norway 32-848400, Spain (91) 640 0085,

Sweden 08-730 49 70, Switzerland 056/20 51 51, U.K. 0635 523545

Limited Warranty

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

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

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

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

the product; owner's abuse, misuse, or negligent acts; and power failure or surges, fire, flood, accident, actions of third parties, or other events outside reasonable control.

Copyright

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

Trademarks

NI-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.

Preface

This manual is intended for use with the *NI-488 MS-DOS Software Reference Manual* (part number 320135-01) or the *NI-488.2 Software Reference Manual for MS-DOS* (part number 320282-01). This manual contains information for programming the NI-488 driver using the Universal Language Interface (ULI).

Organization of This Manual

This manual is organized as follows:

- Chapter 1, *Getting Started*, contains a brief description of the ULI utility and introduces you to the ULI driver file, ULI.COM. It also contains steps for installing the ULI software and the procedures for using the ULI with a generic instrument in a BASIC program.
- Chapter 2, *Terminators*, explains the OUTPUT and INPUT terminators used by the ULI along with graphical models to help you understand the subject.
- Chapter 3, *ULI Functions*, contains a list of the ULI functions. It also contains individual function descriptions along with examples for each function.
- Chapter 4, *Programming Examples*, contains the steps that could be used to program an instrument—in BASICA, QuickBASIC, Turbo BASIC, Microsoft C, Aztec C, FORTRAN, and Turbo Pascal—from your personal computer using the ULI.
- Appendix, *Customer Communication*, contains forms for you to complete to facilitate communication with National Instruments concerning our products.

Conventions Used in This Manual

Throughout this manual, the following conventions are used to distinguish elements of text:

- italic* Italic text denotes emphasis, a cross reference, an introduction to a key concept, or data that can be entered into a string.
- monospace Lowercase text in this font denotes text or characters that are to be literally input from the keyboard, board names, functions, variables, filenames, utilities, and terminators, and for statements and comments taken from program code.
- <> Angle brackets enclose the name of a key on the keyboard—for example, <PageDown>.
- <Enter> Key names begin with a capital letter.

Abbreviations

The following are the abbreviations for units of measure used in this manual.

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

M megabytes of memory
sec second

Acronyms

The following acronyms are used in this manual:

EOI	end or identify
GPIB	General Purpose Interface (IEEE 488) bus
IEEE	Institute of Electrical and Electronic Engineers

Note: References in this manual to IEEE 488 and IEEE 488.2 are referring to the ANSI/IEEE Standard 488.1-1987 and the ANSI/IEEE Standard 488.2-1987, respectively, which define the GPIB specification.

Related Documentation

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

- ANSI/IEEE Standard 488.1-1987, *IEEE Standard Digital Interface for Programmable Instrumentation*
- ANSI/IEEE Standard 488.2-1987, *IEEE Standard Codes, Formats, Protocols, and Common Commands*

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

Contents

Chapter 1

Getting Started	1-1
Introduction	1-1
Installation.....	1-1
Using the ULI.....	1-2
Initializing the System	1-3
Configuring the Device	1-3
Taking Readings.....	1-4
ON SRQ.....	1-5
ON ERROR.....	1-5

Chapter 2

Terminators	2-1
OUTPUT Terminators	2-1
Language Terminator	2-1
GPIB Terminator	2-2
INPUT Terminators.....	2-4
Language Terminator	2-4
GPIB Terminator	2-4

Chapter 3

ULI Functions	3-1
Function Specifications.....	3-3
Function Syntax Conventions.....	3-3
Function Descriptions	3-4
ABORT	3-5
CLEAR.....	3-6
ENTER	3-7
ERRTRAP	3-9
GPIBEOS	3-10
LANGEOS	3-11
LOCAL.....	3-12
LOCAL LOCKOUT.....	3-13
OFFLINE.....	3-14
ONLINE	3-15
OUTPUT.....	3-16
PASS CONTROL.....	3-18
PPOLL.....	3-19
PPOLL CONFIGURE.....	3-20

PPOLL RESPONSE.....	3-21
PPOLL UNCONFIGURE.....	3-22
REMOTE.....	3-23
REQUEST.....	3-24
RESET.....	3-25
SEND.....	3-26
SPOLL.....	3-27
STATUS.....	3-28
TIMEOUT.....	3-29
TRIGGER.....	3-31

Chapter 4

Programming Examples.....	4-1
BASICA Programming Example.....	4-2
QuickBASIC Programming Example.....	4-4
Turbo BASIC Programming Example.....	4-6
Microsoft C Programming Example.....	4-8
Aztec C Programming Example.....	4-10
FORTRAN Programming Example.....	4-12
Turbo Pascal Programming Example 1.....	4-14
Turbo Pascal Programming Example 2.....	4-16

Appendix

Customer Communication.....	A-1
------------------------------------	------------

Figures

Figure 2-1. Character Count Not Included in the OUTPUT Statement...	2-3
Figure 2-2. Character Count Included in the ENTER Statement.....	2-6
Figure 2-3. Character Count Not Included in the ENTER Statement.....	2-8

Tables

Table 3-1. Universal Language Interface Functions.....	3-1
Table 3-2. Timeout Code Values.....	3-29

Chapter 1

Getting Started

This chapter contains a brief description of the ULI utility and introduces you to the ULI driver file, `ULI.COM`. It also contains steps for installing the ULI software and the procedures for using the ULI with a generic instrument in a BASIC program.

Introduction

The Universal Language Interface is a utility that is designed to ease the programming task of those users who are familiar with programming Hewlett-Packard controllers. It can be used on any IBM, IBM PC, or IBM compatible computer. DOS file calls directly access the ULI through the standard I/O commands of a programming language, eliminating the need for a language interface. The Universal Language Interface (ULI) is a low-performance interface to the NI-488 driver. For high performance and maximum flexibility, the standard NI-488 functions should be used.

The ULI driver file `ULI.COM` is included on the distribution diskette that comes with your GPIB interface board. The ULI driver is a `terminate-and-stay-resident` (TSR) utility that interfaces between MS-DOS and the NI-488 driver `GPIB.COM`.

Note: All the descriptions and examples in this chapter and Chapters 2 and 3 are written in BASIC, but can work with most languages, including Microsoft C and TURBO Pascal. Chapter 4 includes several example programs in other programming languages.

Installation

This installation procedure makes the following assumptions:

- Your GPIB interface board has been installed and configured.
- The NI-488 software has been installed and configured.

If you have not completed either of these procedures, refer to the getting started manual that came with your GPIB interface board before going on to install the ULI software.

To install the ULI software, follow this procedure:

1. Change to the directory created by IBSTART when you installed the NI-488 software. For additional information, refer to the *NI-488 MS-DOS Software Reference Manual*, Section Two, *Installation and Configuration of NI-488 Software* or to the *NI-488.2 Software Reference Manual for MS-DOS*, Chapter 2, *Installation and Configuration of NI-488.2 Software*.
2. Enter the ULI command:

```
ULI      <Enter>
```

A message that explains the results of loading the ULI driver into memory displays on the computer screen. The driver stays resident in memory until the computer is turned off or restarted. If you plan to use the driver frequently, the ULI driver can be automatically installed when the computer is turned on by adding the following line to your AUTOEXEC.BAT file.

```
ULI.COM
```

Note: The AUTOEXEC.BAT file is a startup file located in your operating system. By entering the name of the ULI driver into this startup file, the ULI is installed automatically when the computer is turned on. For more information on the AUTOEXEC.BAT file, refer to the DOS reference manual that came with your operating system.

Using the ULI

The ULI driver uses the board names defined in the NI-488 driver GPIB.COM by IBCONF. The symbolic name given to the GPIB board is "GPIB0". The following paragraphs demonstrate how to use the ULI with a generic instrument in a BASIC program. Example programs for other languages can be found in Chapter 4 in this manual. This instrument has a primary address of 2 and no secondary address. The techniques used in this program are general techniques that will apply to the control of most instruments.

Initializing the System

To establish communication with the ULI driver from a programming language such as BASIC, the OPEN statement is used. In BASIC, two OPEN statements are required: one for output and the other for input. Enter the following two commands into the BASIC program you are writing:

```
100 OPEN "GPIB0" FOR OUTPUT AS #1
110 OPEN "GPIB0" FOR INPUT AS #2
```

The next step is to initialize the bus by sending the Interface Clear (IFC) message. Enter the following command:

```
120 PRINT #1, "ABORT"
```

To examine the status of the bus, you can use one of the following procedures:

- You can enable the applications monitor, as described in Section Six of the *NI-488 MS-DOS Software Reference Manual* or in Chapter 7 of the *NI-488.2 Software Reference Manual for MS-DOS*.
- You can enter the following commands:

```
130 PRINT #1, "STATUS"
140 INPUT #2, IBSTA%, IBERR%, IBCNT%
150 PRINT IBSTA%, IBERR%, IBCNT%
```

IBSTA% contains the status of the last bus call, while IBERR% contains an error message if an error condition exists. IBCNT% contains the number of bytes of data successfully transmitted across the bus. These variables are described more fully in Section Four of the *NI-488 Software Reference Manual for MS-DOS* or in Chapter 3 of the *NI-488.2 Software Reference Manual for MS-DOS*. You must install the applications monitor if you need to examine the bus status after executing each GPIB command.

Configuring the Device

After the bus has been initialized, the device is ready to be initialized. To place the instrument in remote mode, enter the REMOTE command:

```
160 PRINT #1, "REMOTE 2"
```

The 2 after REMOTE is the address of the device to be placed in remote mode.

Now that the instrument is ready to receive commands, the OUTPUT command sends any device-specific commands. For example, sending F1R0S2 to the instrument is done as follows:

```
170 PRINT #1, "OUTPUT 2#6; F1R0S2"
```

The OUTPUT command sends the six bytes of data, F1R0S2, to the specified device (in this case, device with address 2). The address can be just a primary address, such as 2 or 5, or it can include a secondary address, such as 0201.

Note: When including a secondary address, the combined primary address and secondary address must be a 4-digit number. The address 0201 contains a primary address of 2 and a secondary address of 1.

Taking Readings

Once the instrument is in the operating mode, it can take a reading and display it.

```
180 PRINT #1, "ENTER 2"  
190 INPUT #2, RD$  
200 PRINT RD$
```

The ENTER command takes an address with an optional secondary address and programs the handler to read from the instrument. The data is stored in the BASIC variable RD\$ when the INPUT statement is executed. A maximum of 255 bytes of data can be read into a BASIC string variable. This maximum string length is a limitation of BASIC. If the device returns more than 255 bytes, use multiple INPUT statements to bring the data into BASIC.

Once the results have been received, any BASIC string operations can be used to manipulate them.

ON SRQ

BASIC has a method for detecting and servicing external interrupts with its ON PEN GOSUB statement. The implementation of ON PEN GOSUB and SRQ handling is explained under *BASICA/QuickBASIC "ON SRQ" Capability* in Section Four of the *NI-488 Software Reference Manual for MS-DOS* or under *"ON SRQ" Capability* in Chapter 3 of the *NI-488.2 Software Reference Manual for MS-DOS*.

ON ERROR

BASIC has a method for handling errors encountered during the operation of the IEEE 488 bus with its ON ERROR GOTO command. If you prefer to write your own error-handling routine with the ON ERROR command, first turn off the ULI automatic error detection by typing the following command:

```
10 PRINT #1, "ERRTRAP OFF"
```

Then make sure that the applications monitor is not installed.

An error value of 24 indicates a GPIB error. All other error values reference non-GPIB errors. Like the ON SRQ feature, ON ERROR requires a service routine and the code for handling the error when it occurs. The following code segment can be used to implement ON ERROR:

```
50 ON ERROR GOTO 4000
```

The following code services an error condition.

```
4000 'SERVICE ROUTINE FOR ERRORS
4010 IF ERR <> 24 THEN GOTO 4090
4020 ELSE CLOSE
4030 OPEN "GPIB0" FOR INPUT AS #1
4040 OPEN "GPIB0" FOR OUTPUT AS #2
4050 PRINT #1, "STATUS"
4060 INPUT #2, IBSTA%, IBERR%, IBCNT%
4070 PRINT IBSTA%, IBERR%, IBCNT%
4080 RESUME NEXT
4090 PRINT "NON-GPIB ERROR"
5000 RESUME NEXT
```

Chapter 2

Terminators

This chapter explains the OUTPUT and INPUT terminators used by the ULI along with graphical models to help you understand the subject.

To understand the I/O termination functions of the Universal Language Interface, it is important to recognize that all data transfers are two-part operations. The first part describes the communication path between your programming language and the driver software of the GPIB board. The second part describes the communication path between the driver and the GPIB instrument. For example, the BASIC programming language requires all string variables to terminate by a CR LF (carriage return-linefeed) pair.

Note: Most languages can use CR LF as a terminator.

(Some GPIB instruments require a semicolon at the end of each command string.) The Universal Language Interface gives you the flexibility to configure your system to match the requirements of any combination of programming languages and instruments. The following paragraphs examine each class of terminator and the implications for language applications.

OUTPUT Terminators

Language Terminator

When sending data or commands to a GPIB instrument, the Universal Language Interface must know what terminating characters to look for to stop processing the string passed to it by the programming language. For the remainder of this discussion, these characters are referred to collectively as the language terminator. The language terminator used by the Universal Language Interface can be one or two characters in length. The default language terminator is CR LF. This pair of characters is used by the BASIC language to terminate all strings and numbers either with an available I/O call or with a pair of characters being appended explicitly to output strings.

Note: The Universal Language Interface uses the language terminator to recognize the end of the string passed to it by the application program. If the terminator is changed to a value that does not match the value that your language actually uses as a terminator, the program reads the required data incorrectly and/or will not terminate the I/O command. Using CR LF is advised, regardless of the programming language, as almost every language has I/O functions that use the CR LF terminator.

To change the language terminator to some other character(s), the LANGEOS function is used. For example, to set the language terminator to CR LF in BASIC, you enter the following statement:

```
PRINT #1, "LANGEOS CR LF"
```

Note: Since CR LF is the default language terminator, you do not normally need to enter this statement.

GPIB Terminator

For outputs, the GPIB terminator is the character or pair of characters sent to the GPIB device by the driver software of the GPIB board.

These characters replace the language terminator characters. Therefore, if you want to send the same characters to the GPIB instrument that are used for a language terminator by the programming language, the GPIB terminator first must be set to the same values as the language terminator.

For example, to send CR LF (used by BASIC) to the instrument, you must use the following function call:

```
PRINT #1, "GPIBEOS OUT CR LF"
```

Figure 2-1 shows an example of an OUTPUT command that replaces the language terminator with a specified GPIB terminator.

Note: The default setting of the GPIB terminator is disabled (that is, no termination characters are sent).

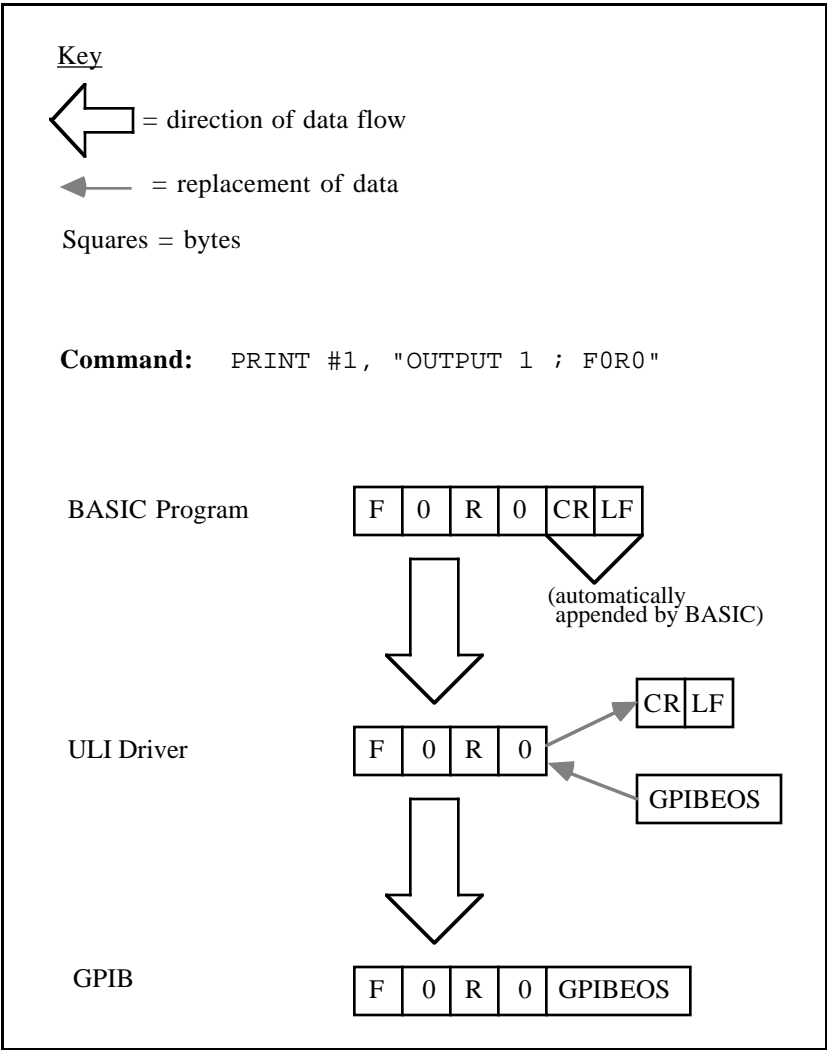


Figure 2-1. Character Count Not Included in the OUTPUT Statement

If a GPIB terminator has been defined, it can be disabled by specifying a count value in an `OUTPUT` statement. The count value, if specified, always has precedence over a GPIB terminator. For example, the statement

```
PRINT #1, "OUTPUT 1#4;FOR2"
```

is a BASIC statement that sends the characters F, O, R, and 2 to the GPIB device, but does *not* send a GPIB terminator because the count value of four does not leave enough space for a terminator character. If the count had been six or more (five for a one-character GPIB terminator), a GPIB terminator would have been included.

INPUT Terminators

Language Terminator

The INPUT language terminator (used by the `ENTER` statement) operates similar to the `OUTPUT` language terminator, except that the direction of the data flow is reversed. Thus, data read from a GPIB instrument is separated from its GPIB terminator (if present). Then the GPIB terminator is replaced with the defined language terminator and appended to the data before being sent to the programming language. It is necessary sometimes to treat data as binary information, rather than as formatted ASCII strings or numeric values. For example, if a count value is provided in the `ENTER` statement, the application receives only as many characters as are requested, thus it may not get the language terminator required by the programming language. In this case, the binary transfer functions of the programming language (such as `INPUT$` in BASIC) can be used. (See examples in the following discussion *GPIB Terminator*.)

GPIB Terminator

Since each instrument manufacturer can use a different method for indicating the completion of a data stream, it is often necessary to use a different GPIB terminator for each type of device connected to your personal computer.

There are four conditions under which a device input terminates: Two conditions require the reception of the GPIB terminator and two conditions terminate based on the occurrence of some predefined or external event.

The four conditions are as follows:

Note: In each condition, the letters on the *data* line represent bytes of data that you can enter into the INPUT string.

Condition 1: A predefined count has been reached.

Example:

```
PRINT #1, "ENTER 16#10"  
DATA$=INPUT$(10,2)
```

data: a b c d e f g h i j

Figure 2-2 shows how Condition 1 is handled by the ULI driver.

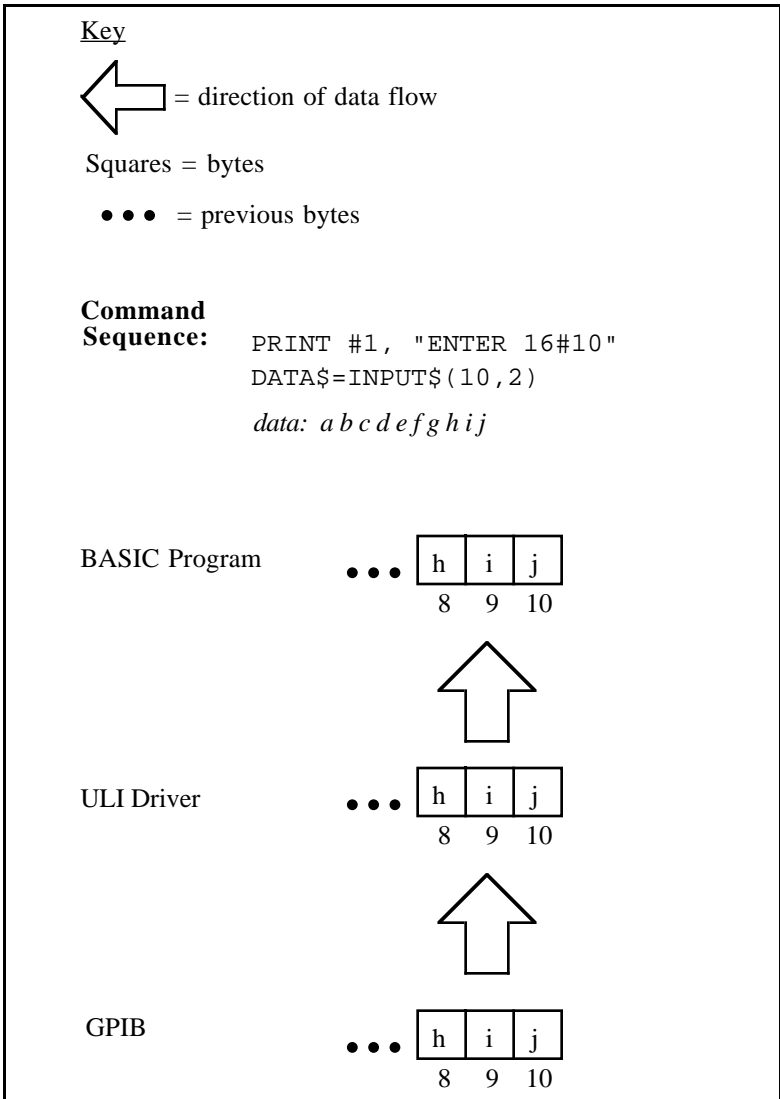


Figure 2-2. Character Count Included in the ENTER Statement

- Condition 2. A one-character GPIB terminator has been defined and received.

Example:

```
PRINT #1, "GPIBEOS IN CR"  
PRINT #1, "ENTER 16"  
INPUT #2, DATA$
```

data: a b c d e CR

- Condition 3. A two-character GPIB terminator has been defined and received in the correct order.

Example:

```
PRINT #1, "GPIBEOS IN CHR(\x30)  
          CHR(\x31) "  
PRINT #1, "ENTER 16"  
INPUT #2, DATA$
```

data: a b c d e f 0 1 (ASCII 0 = hex 30, ASCII 1 = hex 31)

Note: If either hex 30 *or* hex 31 is received, but not both (or if hex 31 *and* hex 30 are received in that order), the input does not terminate. If hex 30 *and* hex 31 are received in that order, the input terminates.

Figure 2-3 shows how Conditions 2 and 3 are handled by the ULI driver.

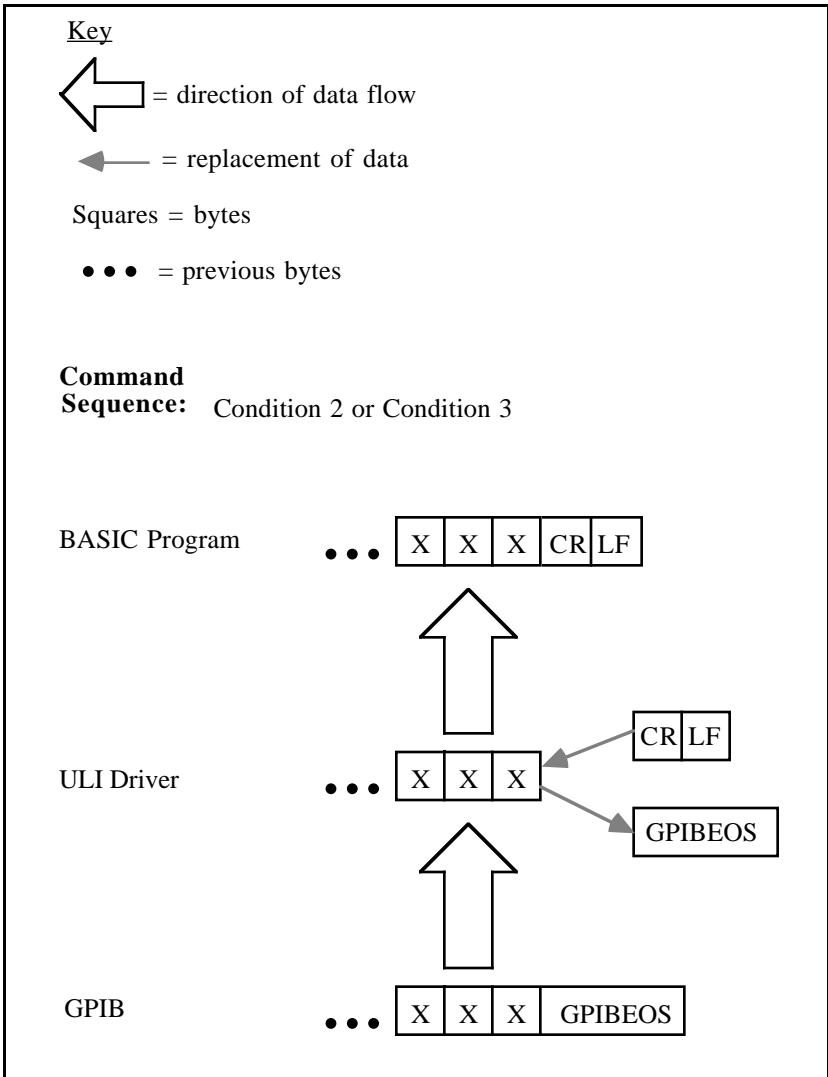


Figure 2-3. Character Count Not Included in the ENTER Statement

Condition 4. The GPIB EOI line has been asserted with the last data byte.

Example:

```
PRINT #1,ENTER 16"  
INPUT #2,DATA$
```

data: a b c d e f g h

EOI is asserted with the last character (*h*).

Note: The GPIB terminator characters will not be sent to the language; they are used exclusively for communication between the device and the driver software.

Chapter 3

ULI Functions

This chapter contains a list of the ULI functions. It also contains individual function descriptions along with examples for each function.

Table 3-1 contains a complete list of the ULI functions and a short description of each.

Table 3-1. Universal Language Interface Functions

Function	Description
ABORT	Initializes the bus and terminates a bus transfer if a transfer is in progress
CLEAR	Clears selected device/devices
ENTER	Reads data from the GPIB
ERRTRAP	Enables or disables automatic error trapping
GPIBEOS GPIB data	Selects the termination characters for
LANGEOS	Sets the termination characters for data passed to and from an application program
LOCAL	Places the addressed device/devices or all devices in local mode
LOCAL LOCKOUT	Inhibits all devices from being controlled from their front panel
OFFLINE	Places the device or interface board offline

(continues)

Table 3-1. Universal Language Interface Functions (continued)

Function	Description
ONLINE	Places the interface board online
OUTPUT	Sends data to a device or multiple devices
PASS CONTROL	Passes control to another controller on bus
PPOLL	Conducts a parallel poll of devices
PPOLL CONFIGURE	Configures parallel poll response of devices
PPOLL RESPONSE	Sets the ist bit on the GPIB interface board for parallel poll responses
PPOLL UNCONFIGURE devices	Disables the parallel poll response of devices
REMOTE	Sets the REN bus line and places devices in remote mode
REQUEST	Requests service from the Controller
RESET	Sets the configuration parameters to default values
SEND	Sends GPIB management commands
SPOLL	Serial polls a device
STATUS	Returns the status of the previous bus call
TIMEOUT	Sets the time limit
TRIGGER	Triggers selected device/devices

Function Specifications

The function descriptions, which comprise the rest of this chapter, give the proper syntax for each function. Prior to using these commands, the ULI driver must be opened for input and output using the OPEN command.

```
OPEN "GPIB0" FOR OUTPUT AS #1
OPEN "GPIB0" FOR INPUT AS #2
```

The primary address can be a 1-digit or 2-digit integer. A combined primary address and secondary address must be a 4-digit integer. An instrument at primary address 3 and secondary address 1 is selected with the combined address 0301.

Function Syntax Conventions

The following conventions are used in the syntax descriptions for the ULI functions:

255 Characters	No command or <i>data</i> part of the command can be more than 255 characters long.
UPPERCASE TEXT	Items in uppercase text, such as ENTER or ABORT, must be entered exactly as shown.
Blank space	Blank spaces in commands are generally ignored; thus, LOCAL LOCKOUT is the same as LOCALLOCKOUT. Spaces are <i>not</i> ignored in two places: the data part of an OUTPUT command and between the arguments of a SEND command.
# and ;	Enter the # character exactly as shown. Enter the ; character exactly as shown.
[]	Items enclosed in square brackets ([response]) are optional.

- | Multiple items enclosed in square brackets separated by vertical lines ([*item1*] | [*item2*] | [*item3*]) are optional. Any one or none can be chosen. No more than one item can be selected.
- < > Items enclosed in angle brackets (<*addr*>) must be replaced by an appropriate value. For example, *addr* can be replaced with primary address 2; thus, the number 2 would be entered.

Function Descriptions

The remainder of this chapter contains a detailed description of each ULI function with at least one example of how to use each one. The functions are listed in alphabetical order for easy reference.

ABORT

- Purpose:** Initializes the bus and terminates a bus data transfer if a transfer is in progress.
- Format:** ABORT
- Response:** None
- Remarks:** The ABORT command causes the GPIB board to become System Controller (SC) and Controller-In-Charge (CIC). It then asserts the Interface Clear (IFC) line followed by the Attention (ATN) line, which stops any bus transaction.
- Using the ABORT command to initialize the bus is a shortcut to the normal procedure.

Example:

Initialize the bus or terminate a bus data transfer.

```
PRINT #1, "ABORT"
```

CLEAR

Purpose: Clears selected device/devices.

Format: CLEAR [`<addr>`[`, <addr>`]]

`addr` is the primary address of a device (and optional secondary address).

A comma (,) separates multiple addresses.

Response: None

Remarks: The CLEAR command clears the internal or device functions of a specified device by sending the Selected Device Clear (SDC) message or Device Clear (DCL) if no address is present.

Examples:

1. Clear all devices.

```
PRINT #1, "CLEAR"
```

2. Clear device 2.

```
PRINT #1, "CLEAR 2"
```

ENTER

Purpose: Reads data from the GPIB.

Format: ENTER [<addr>] [# <count>] [DMA]
Preserve Blanks

addr is the IEEE-bus device address.

count is the number of characters to read.

DMA turns Direct Memory Access (DMA) on.

Preserve Blanks preserves leading blanks of a string variable that may be part of a device's output format. BASIC normally removes these blanks.

Response: Device-specific data

Remarks: If a device address is present, that device is addressed to talk. If no address is present, the GPIB board must already be configured to receive data, either as a result of an immediately preceding ENTER command or as a result of a SEND command. If the count is specified, count characters will be read from the device. Otherwise, the subsequent INPUT command terminates upon detection of the GPIBEOS input terminator.

Examples:

1. Read a line of data from device 2.

```
PRINT #1, "ENTER 2"  
LINE INPUT #2, A$
```

The Line Input statement ignores delimiters in the input stream. See the BASIC reference manual that is supplied with your version of BASIC for additional information.

2. Read 10 bytes of data from device 16.

```
PRINT #1, "ENTER 16#10"  
B$ = INPUT$(10, 2)
```

3. Read data using GPIBEOS input terminator.

```
PRINT #1, "ENTER 2"  
INPUT #2, A$
```

ERRTRAP

Purpose: Enables or disables automatic error trapping.

Format: ERRTRAP [ON|OFF]

Response: None

Remarks: If ERRTRAP is disabled, the only error message that will appear on your screen is *Device I/O Error*. You must enable ERRTRAP to receive GPIB-specific error messages (such as *No Listener*). The default setting for this feature is ON (enabled).

Example:

Disable automatic error detection.

```
PRINT #1, "ERRTRAP OFF"
```

GPIBEOS

Purpose: Selects the termination characters for GPIB data.

Format: GPIBEOS [IN|OUT] <term>

IN means input terminator.

OUT means output terminator.

If neither IN nor OUT are present, then both terminators are set.

term is the character that will be used as a terminator.

Response: None

Remarks: See the terminator discussion in Chapter 2, *Terminators*. term can be:

CR Carriage Return

LF Line Feed

CHR(#) # is integer between 0 and 255

'X X is any printable ASCII character

The GPIBEOS terminator assignments are cancelled if this command is executed without parameters.

Examples:

1. Set both terminators to CR.

```
PRINT #1, "GPIBEOS CR"
```

2. Set the input terminator to LF.

```
PRINT #1, "GPIBEOS IN LF"
```

3. Set the output terminator to CR LF.

```
PRINT #1, "GPIBEOS OUT CR LF"
```

LANGEOS

Purpose: Sets the termination characters for data passed to and from an application program.

Format: LANGEOS [IN|OUT] <term>

IN means input terminator.

OUT means output terminator.

If neither IN nor OUT are present, both terminators are set the same.

term is the character that will be used as a terminator.

Response: None

Remarks: See the terminator discussion in Chapter 2, *Terminators*. term can be:

CR Carriage Return

LF Line Feed

CHR(#) # is integer between 0 and 255

' X X is any printable ASCII character

Examples:

1. Set both terminators to CR.

```
PRINT #1, "LANGEOS CR"
```

2. Set the input terminator to LF.

```
PRINT #1, "LANGEOS IN LF"
```

3. Set the output terminator to CR LF.

```
PRINT #1, "LANGEOS OUT CR LF"
```

LOCAL

Purpose: Places the addressed device/devices or all devices in local mode.

Format: LOCAL [<addr>[, <addr>]]

addr is the primary address of the device (optional secondary address).

A comma (,) separates multiple primary addresses.

Response: None

Remarks: The Go To Local (GTL) message is sent to the device, placing it in manual operation mode if an address is present; otherwise, the Remote Enable line is unasserted.

Examples:

1. Unassert the Remote Enable line.

```
PRINT #1, "LOCAL "
```

2. Send Go To Local to devices 2 and 5.

```
PRINT #1, "LOCAL 2,5 "
```

LOCAL LOCKOUT

Purpose: Inhibits all devices from being controlled from their front panel.

Format: LOCAL LOCKOUT

Response: None

Example:

Send Local Lockout bus command.

```
PRINT #1, "LOCAL LOCKOUT"
```

OFFLINE

Purpose: Places the device or interface board offline which is equivalent to disconnecting its GPIB cable from the other devices.

Format: OFFLINE

Response: None

Example:

Place the device offline.

```
PRINT #1, "OFFLINE"
```

ONLINE

Purpose: Places interface board online which restores the default configuration strings of a device or interface board.

Format: ONLINE

Response: None

Example:

Place interface board online.

```
PRINT #1, "ONLINE"
```

OUTPUT

Purpose: Sends data to a device or multiple devices.

Format: OUTPUT [`<addr>`[`,``<addr>`]] [`#` `<count>`]
 [`DMA`] [`END`|`NOEND`]; `<data>`

`addr` is the address of the device (with an optional secondary address).

`count` is the number of characters to send.

`DMA` turns Direct Memory Access (DMA) on.

`END`|`NOEND` determines whether or not the EOI line is asserted at the end of a data transmission.

`data` is the information that is being sent to the device or devices.

A semicolon (`;`) separates the command from the device-dependent data.

Response: None

Remarks: If a device address is present, that device is addressed to listen. If no address is present, the GPIB board must already be configured to transmit data, either as a result of an immediately preceding `OUTPUT` command or as a result of a `SEND` command. If the count is specified, that exact number of characters will be sent to the device. Otherwise, the `OUTPUT` command adds the GPIBEOS output terminator.

Examples:

1. Send `CURV?` `<GPIBEOS>` to device 2.

Note: You must have already sent the `GPIBEOS` command to set up the EOS termination character.

```
PRINT #1, "OUTPUT 2;CURV?"
```

2. Send FOR0 to device 10, 11, and 12.

```
PRINT #1, "OUTPUT 10,11,12#4;FOR0"
```

PASS CONTROL

Purpose: Passes control to another controller on the bus.

Format: PASS CONTROL <addr>
addr is the address of a device.

Response: None

Remarks: The GPIB board can regain control by issuing an ABORT command.

Example:

Control is passed to device 2.

```
PRINT #1, "PASS CONTROL 2"
```

PPOLL

Purpose: Conducts a parallel poll of devices.

Format: PPOLL

Response: Number between 0 and 255 that is the parallel poll response.

Remarks: Devices on the GPIB bus are polled in parallel (up to eight devices). Not every GPIB device supports parallel polls.

Example:

Conduct a parallel poll.

```
PRINT #1 "PPOLL"
```

```
INPUT #2, PPRESP%
```

PPOLL CONFIGURE

Purpose: Configures the parallel poll response of devices.

Format: PPOLL CONFIGURE <addr> ; <number>

addr is the primary address of the device to be configured.

A semicolon (;) separates the command from the device-dependent data.

number is the pattern that specifies which line is asserted by the device on a parallel poll and whether the parallel poll will be enabled or disabled.

Response: None

Remarks: See *IBPPC* in Section Four A of the *NI-488 MS-DOS Software Reference Manual* or in Chapter 5 of the *NI-488.2 Software Reference Manual for MS-DOS* for a complete discussion of parallel poll configurations.

Example:

Send decimal 41 to device 2 for configuring the parallel poll.

```
PRINT #1, "PPOLL CONFIGURE 2;41"
```

PPOLL RESPONSE

- Purpose:** Sets the individual status (ist) bit on the GPIB interface board for parallel poll responses.
- Format:** PPOLL RESPONSE <number>
- number indicates status of the GPIB interface board . If number is non-zero, the ist bit is set. If number is zero, the ist bit is cleared.
- Response:** None
- Remarks:** See *IBPPC* in Section Four A of the *NI-488 Software Reference Manual for MS-DOS* or in Chapter 5 of the *NI-488.2 Software Reference Manual for MS-DOS* for a complete discussion of parallel poll configurations.
- Example:**

Sets the individual status bit.

```
PRINT #1, "PPOLL RESPONSE 2"
```

PPOLL UNCONFIGURE

Purpose: Disables the parallel poll response for devices.

Format: PPOLL UNCONFIGURE [<addr>[,<addr>]]

Response: None

Example:

1. Unconfigure device 2.

```
PRINT #1, "PPOLL UNCONFIGURE 2"
```

2. Unconfigure all devices.

```
PRINT #1, "PPOLL UNCONFIGURE"
```

REMOTE

Purpose: Sets the Remote Enable (REN) bus line and places devices in remote mode.

Format: REMOTE [*<addr>*[, *<addr>*]]

addr is the primary address of a device (optional secondary address).

A comma (,) separates multiple primary addresses.

Response: None

Remarks: The REN bus line is asserted. If a device address is specified, that device is also addressed to listen in order to place it in remote programming mode.

Examples:

1. Assert Remote Enable.

```
PRINT #1, "REMOTE"
```

2. Assert Remote Enable and address devices 2 and 5 to listen.

```
PRINT #1, "REMOTE 2, 5"
```

REQUEST

Purpose: Request service and/or set or change the serial poll status byte.

Format: REQUEST <number>

number is the status byte that the GPIB board provides when serial polled by another device that is the GPIB CIC. If bit 6 (the hex 40 bit) of the number is set, the GPIB board additionally requests service from the Controller by asserting the GPIB SRQ line.

Response: None

Remarks: The REQUEST command is used to request service from the Controller using the Service Request (SRQ) signal and to provide a system-dependent status byte when the Controller serial polls the GPIB board.

Example:

Request service with a status byte of &H41.

```
PRINT #1, "REQUEST &H41";
```

RESET

Purpose: Sets the configuration parameters (primary and secondary address, LANGEOS, and GPIBEOS characters) to their default values.

Format: RESET

Response: None

Example:

Restore address and EOS parameters.

```
PRINT #1, "RESET"
```

SEND

Purpose: Sends GPIB management commands.

Format: SEND [MTA|TALK <addr>] [MLA|LISTEN <addr>] [UNL] [UNT] [DATA <number> [, <number> , <...>]]

MTA is the talk address of the GPIB board.

TALK *addr* combines the talk address of a device with primary address *addr*.

MLA is the listen address of the GPIB board.

LISTEN *addr* combines the listen address of a device with primary address *addr*.

UNL is the unlisten command message. It is always defined to be &H3F.

UNT is the untalk command message. It is always defined to be &H5F.

DATA is the device-specific data to be transferred.

number is an integer between 0 and 255.

A comma (,) separates multiple primary addresses.

Response: None

Remarks: Gives byte-by-byte control of data and transfers on the bus. These commands give the maximum flexibility and control to the GPIB controller.

Example:

Send two data bytes, a decimal 1 and a decimal 2, to the device at address 1 from the device at address 2.

```
PRINT #1, "SEND LISTEN 1 TALK 2 DATA 1,2"
```

SPOLL

Purpose: Serial polls a device.

Format: SPOLL [<addr>]

addr is the primary address of a device.

Response: An integer between 0 and 255 that is the Serial Poll response of the device.

Remarks: See *IPRSP* in Section Four A of the *NI-488 Software Reference Manual for MS-DOS* or in Chapter 5 of the *NI-488.2 Software Reference Manual for MS-DOS*. Refer also to *Automatic Serial Polling* in Section Four of the *NI-488 MS-DOS Software Reference Manual* or in Chapter 5 of the *NI-488.2 Software Reference Manual for MS-DOS*.

Example:

1. Serial poll device 2.

```
PRINT #1, "SPOLL 2"
```

2. Receive the serial poll status.

```
INPUT #2, SP%
```

STATUS

Purpose: Returns the status of the previous bus call.

Format: STATUS

Response: Returns the status variables: IBSTA%, IBERR%, and IBCNT%.

Remarks: IBSTA%, IBERR%, and IBCNT% correspond to status, error, and count variables, respectively, as described in Section Four, *NI-488 Software Characteristics* of the *NI-488 Software Reference Manual for MS-DOS* or in Chapter 3, *Understanding the NI-488.2 Software*, of the *NI-488.2 Software Reference Manual for MS-DOS*. This command is not necessary if the applications monitor and/or automatic error detection have been enabled.

Example:

Check status

```
PRINT #1, "STATUS"
```

```
INPUT #2, IBSTA%, IBERR%, IBCNT%.
```

TIMEOUT

Purpose: Sets the time limit.

Format: TIMEOUT <number>

number is the code for the length of time for a timeout condition to occur.

Response: None

Remarks: Each data or control transaction on the bus must be completed within a certain time limit. This limit can be set to any one of the limits shown in Table 3-2.

Table 3-2. Timeout Code Values

Number	Minimum Timeout
0	disabled
1	10 μsec
2	30 μsec
3	100 μsec
4	300 μsec
5	1 msec
6	3 msec
7	10 msec
8	30 msec
9	100 msec
10	300 msec
11	1 sec
12	3 sec
13	10 sec
14	30 sec
15	100 sec
16	300 sec
17	1000 sec

Example:

Set the timeout period to 10 sec.

```
PRINT #1, "TIMEOUT 13"
```

TRIGGER

Purpose: Triggers selected device/devices.

Format: TRIGGER [<addr>[, <addr>]]

addr is the primary address of a device (optional secondary address).

A comma (,) separates multiple addresses.

Response: None

Remarks: The TRIGGER command sends a Group Execute Trigger (GET) message to each specified device. If no devices are specified, GET will only be received by those devices previously addressed to listen.

Example:

Issue a TRIGGER command to devices 2 and 5.

```
PRINT #1 "TRIGGER 2, 5"
```

Chapter 4

Programming Examples

This chapter contains the steps that could be used to program an instrument—in BASICA, QuickBASIC, Turbo BASIC, Microsoft C, Aztec C, FORTRAN, and Turbo Pascal—from your personal computer using the ULI.

The following examples illustrate the programming steps that could be used to program an instrument from your personal computer using the Universal Language Interface.

BASICA Programming Example

```

1      'Demo program for using the Universal
2      'Language Interface with BASICA or GW-BASIC.
3      'Program reads from and writes to a device.
4      'If SRQ line goes high, the program
5      'will suspend normal operation and
6      'execute the interrupt service routine at
7      'line 1000.

9      'Set up an 'ON SRQ' interrupt and enable
10     'checking.
15     ON PEN GOSUB 1000
20     PEN ON

25     'Initialize the input and output channels
30     OPEN "gpib0" FOR OUTPUT AS #1
40     OPEN "gpib0" FOR INPUT AS #2

45     'Initialize the bus and reset to default
46     'parameters
50     PRINT #1, "ABORT"
55     PRINT #1, "RESET"

60     'Place the device in remote state
65     PRINT #1, "REMOTE 1"

70     'Send program commands to the device
75     PRINT #1, "OUTPUT 1; F1R0S2"

80     'Check the status of the bus
85     PRINT #1, "STATUS"
90     INPUT #2, IBSTA%, IBERR%, IBCNT%
95     PRINT "IBSTA=0x";HEX$(IBSTA%)
100    PRINT "IBERR=";IBERR%
105    PRINT "IBCNT=";IBCNT%

110    'Read data from the device
115    PRINT #1, "ENTER 1#20"
120    RD$=INPUT$(20,2)
121    'or PRINT #1,"ENTER 1"
122    ' INPUT #2, RD$
123    'if device asserts EOI on last byte of write
130    PRINT RD$

```

```
135 PRINT "Waiting for SRQ interrupt.  
136 Hit 'q' to quit."  
140 WHILE CHAR$ <> "q"  
150 CHAR$ = INKEY$  
160 WEND  
999 END  
  
1000 REM *** SRQ interrupt service routine ***  
1005 PRINT "SRQ interrupt!!!"  
1010 PRINT #1, "SPOLL 1"  
1020 INPUT #2, SP%  
1030 PRINT "Poll byte: "; SP%; " (decimal)"  
1040 RETURN
```

QuickBASIC Programming Example

'Program to open device 1, send command an
'read response. If SRQ line goes high,
'the program will suspend normal operation and
'execute the interrupt service routine at label
'INTSRVC. Set up an 'ON SRQ' interrupt and enable
'checking.

```
ON PEN GOSUB INTSRVC
PEN ON
```

```
'Initialize the input and output channels
OPEN "gpib0" FOR OUTPUT AS #1
OPEN "gpib0" FOR INPUT AS #2
```

```
'Initialize the bus and reset to default parameters
PRINT #1, "ABORT"
PRINT #1, "RESET"
```

```
'Place the device in remote state
PRINT #1, "REMOTE 1"
```

```
'Send program commands to the device
PRINT #1, "OUTPUT 1; F1R0S2"
```

```
'Check the status of the bus
PRINT #1, "STATUS"
INPUT #2, IBSTA%, IBERR%, IBCNT%
PRINT "IBSTA=0x"; HEX$(IBSTA%)
PRINT "IBERR="; IBERR%
PRINT "IBCNT="; IBCNT%
```

```
'Read data from the device
PRINT #1, "ENTER 1#20" 'or PRINT #1, "ENTER 1"
RD$ = INPUT$(20, 2) ' INPUT #2, RD$
PRINT RD$ 'if device asserts EOI
' on last byte of write
```

```
PRINT "Waiting for SRQ interrupt. Hit 'q' to quit."
WHILE char$ <> "q"
    char$ = INKEY$
WEND
END
```

```
INTSRVC:
REM *** SRQ interrupt service routine ***
PRINT "SRQ interrupt!!!"
PRINT #1, "SPOLL 1"
INPUT #2, SP%
PRINT "Poll byte: "; SP%; " (decimal)"
RETURN
```

Turbo BASIC Programming Example

'Demo program for using the Universal Language

Interface with Turbo Basic.

'Program reads from and writes to a device.

'If SRQ line goes high, the program will

'suspend normal operation and execute the

'interrupt service routine at line 1000.

'Set up an 'ON SRQ' interrupt and enable checking.

ON PEN GOSUB INTSRVC

PEN ON

'Initialize the input and output channels

OPEN "gpib0" FOR OUTPUT AS #1

OPEN "gpib0" FOR BINARY AS #2

'Initialize the bus and reset to default parameters

PRINT #1, "ABORT"

PRINT #1, "RESET"

'Place the device in remote state

PRINT #1, "REMOTE 1"

'Send program commands to the device

PRINT #1, "OUTPUT 1; FIR0S2"

'Check the status of the bus

PRINT #1, "STATUS"

GET\$ #2, 18, STATUS\$

'Parse and ignore header byte

'Each variable is 5 bytes

PRINT "IBSTA=";MID\$(STATUS\$, 2, 5)

PRINT "IBERR=";MID\$(STATUS\$, 8, 5)

PRINT "IBCNT=";MID\$(STATUS\$, 14, 5)

'Read data from the device

PRINT #1, "ENTER 1#20"

GET\$ #2, 20, RD\$

PRINT RD\$

'or PRINT #1, "ENTER 1"

' GET\$ #2, 255, RD\$

'if device asserts EOI on

'last byte of write

```
PRINT "Waiting for SRQ interrupt. Hit 'q' to quit."  
WHILE char$ <> "q"  
    char$ = INKEY$  
WEND  
END
```

```
INTSRVC:  
REM *** SRQ interrupt service routine ***  
PRINT "SRQ interrupt!!!"  
PRINT #1, "SPOLL 1"  
GET$ #2, 3, SP$  
PRINT "Poll byte: "; SP$; " (decimal)"  
RETURN
```

Microsoft C Programming Example

```

/* Demo program for using the Universal Language
Interface with Microsoft C 4.0 (and later
versions), Microsoft Quick C 2.0, and TurboC 2.0.
Program reads from, writes to, and polls a device.
A new line (\n) must terminate command string.
*/

#include <stdio.h>
#include <stdlib.h>

main ()
{
    char *buf, iberr[6], ibcnt[6], ibsta[6], header;
    FILE *bd;
    int i, sp;

/* Get file handle for I/O */
    buf = (char *) malloc( 300);
    if ( buf==NULL) exit(1);
    bd = fopen( "gpib0","r+");

/* Initialize the bus and reset to default parameters */
    fputs( "ABORT\n", bd);
    fputs( "RESET\n", bd);
    fflush( bd);

/* Send commands to program the device */
    fputs( "OUTPUT 1;F1R0S2\n", bd);
    fflush( bd);

/* Check status of the bus */
    fputs( "STATUS\n", bd);
    fflush( bd);
    rewind( bd);
    if( fscanf( bd, "%c %5c,%5c,%5c"
        , &header, ibsta, iberr, ibcnt)==EOF) {
        puts( "No status bytes read.");
    } else {

```

```

        printf( "STA=%.5s ERR=%.5s CNT=%.5s\n"
                , ibsta, iberr, ibcnt);
    }/* if-else */
    rewind( bd);

/* Read data from the device */
    fputs( "ENTER 1#20\n", bd);
    fflush( bd);
    rewind( bd);
    if( fgets( buf, 300, bd)==NULL) {
        puts("No bytes read.");
    } else {
        printf( "You entered: %s\n", buf);
    } /* if-else */
    rewind( bd);

/* Poll a device */
    fputs( "SPOLL 1\n", bd);
    fflush( bd);
    rewind( bd);
    if( fscanf( bd, "%3c", buf)==EOF) {
        puts( "No poll byte read.");
    } else {
        printf( "Poll byte : %.3s \n", buf);
    } /* if-else */
    rewind( bd);

    free( buf);
    fclose( bd);
}/* main */

```

Aztec C Programming Example

```
/* Demo program for using the Universal Language
Interface with Aztec C 3.20E.
Program reads from, writes to, and polls a device.
A new line (\n) must terminate command string.
*/

#include <stdio.h>

#define SEEK_SET 0

void rewind();

main ()
{
    char *buf, iberr[6], ibcnt[6], ibsta[6], header;
    FILE *bd;
    int i, sp;

    /* Get file handle for I/O */
    buf = (char *) malloc( 300);
    if ( buf==NULL) exit(1);
    bd = fopen( "gpib0","r+");

    /* Initialize the bus and reset to default parameters */
    fputs( "ABORT\n", bd);
    fputs( "RESET\n", bd);
    fflush( bd);

    /* Send commands to program the device */
    fputs( "OUTPUT 1;FIR0S2\n", bd);
    fflush( bd);

    /* Check status of the bus */
    fputs( "STATUS\n", bd);
    fflush( bd);
    rewind( bd);
    if( fscanf( bd, "%c %5c,%5c,%5c"
                , &header, ibsta, iberr, ibcnt)==EOF)
    {
```

```

                puts( "No status bytes read.");
    } else {
        printf( "STA=%.5s ERR=%.5s CNT=%.5s\n"
            , ibsta, iberr, ibcnt);
    } /* if-else */
    rewind( bd);

/* Read data from the device */
    fputs( "ENTER 1#20\n", bd);
    fflush( bd);
    rewind( bd);
    if( fgets( buf, 300, bd)==NULL) {
        puts("No bytes read.");
    } else {
        printf( "You entered: %s\n", buf);
    } /* if-else */
    rewind( bd);

/* Poll a device */
    fputs( "SPOLL 1\n", bd);
    fflush( bd);
    rewind( bd);
    if( fscanf( bd, "%3c", buf)==EOF) {
        puts( "No poll byte read.");
    } else {
        printf( "Poll byte : %.3s \n", buf);
    } /* if-else */
    rewind( bd);

    free( buf);
    fclose( bd);
} /* main */

void rewind( fp)
FILE *fp;
{
    fseek( fp, 0L, SEEK_SET);
    clearerr( fp);
} /* rewind */

```

FORTRAN Programming Example

- c HP Style Calls DEMO Program
- c Demo program for using the Universal Language
- c Interface with MS Fortran 4.1. Program reads
- c from, writes to, and polls a device.

Character Reading*80
Integer Length

- c Initialize the I/O channel
 - Open (1,File='gpib0',Status='OLD',
 - 1 Access='SEQUENTIAL',Form='FORMATTED',
 - 2 Mode='READWRITE')
- c Initialize the bus and reset to default parameters
 - Write(1,*)'ABORT'
 - Write(1,*)'RESET'
- c Place the device in remote mode
 - Write(1,*)'REMOTE 1'
- c Send commands to program device
 - Write(1,*)'OUTPUT 1;F1R0S2'
- c Check the status of the bus
 - Write(1,*)'STATUS'
 - Read(1,'(A)')Reading
 - Rewind 1
 - Write(*,*)'IBSTA=',Reading(2:6)
 - Write(*,*)'IBERR=',Reading(8:12)
 - Write(*,*)'IBSTA=',Reading(14:18)
- c Read data from device
 - Write(1,*)'ENTER 1#20'
 - Read(1,'(A)')Reading
 - Rewind 1
 - Write(*,*)'You entered:',
 - 1 Reading(1:Length(Reading,80))

```
c  Poll the device
    Write(1,*)'SPOLL 1'
    Read(1,'(A)')Reading
    Rewind 1
    Write(*,*)'Poll byte: ',Reading(1:3)

    Close (1)
    END

    Integer Function Length(Data,Max)
    Character Data*127
    Length=Max-1
    Do 10 c=1,Max-2
    If (Data(c:c) .eq. char(10) .and.
1     Data(c+1:c+1) .eq. char(13)) then
        Length=c-1
    Goto 20
    EndIf
10    Continue
20    Continue
    END
```

Turbo Pascal Programming Example 1

This program reads the response from the device directly into a string type variable. Notice that the length of the string is explicitly calculated. Alternatively, data could be read into a character array where no length calculation is needed. However, this format is convenient for screen output.

Program HP_Calls

var

```
    bd: text;
    s: string[255];
    i: integer;
```

Begin

```
    { Open device GPIB0 }

    Assign(bd, 'gpib0');
    Rewrite(bd);

    { Send F1R1S2 to device 16 }

    Writeln(bd, 'OUTPUT 16; F1R1S2');

    { Send ENTER command to device 16 }

    Writeln(bd, 'ENTER 16');
    Reset(bd);

    { Read data into a string }

    i := 1;
    while (not Eof(bd)) do
        begin
            Read(bd, s[i]);
            i := i + 1;
        end;

    { Calculate string length and store in header }      { byte (not
necessary for character array) }
```

```
s[0] := chr(i-3);  
  
{ Write to screen }  
  
Writeln ('Reading = ', s)  
Close(bd);
```

End.

Turbo Pascal Programming Example 2

```

Program HP_Calls;
{ Demo program for using the Universal Language
  Interface with Turbo Pascal 5.0.
  Program reads from, writes to, and polls a
  device. }

```

```

var
    gpib: text;
    data: string[255];
    status: string[20];

begin
{ Open device GPIB0 }
    Assign(gpib,'gpib0');
    Rewrite(gpib);

{ Reset and configure board }
    Writeln(gpib, 'ABORT');
    Writeln(gpib, 'RESET');

{ Place device in remote mode }
    Writeln(gpib, 'REMOTE 1');

{ Send data to device }
    Writeln(gpib, 'OUTPUT 1;F2R0S1');

{ Check status of the bus }
    Writeln(gpib, 'STATUS');
    Reset(gpib);
    Readln(gpib, status);
    Writeln('IBSTA = ', copy(status, 2, 5));
    Writeln('IBERR = ', copy(status, 8, 5));
    Writeln('IBCNT = ', copy(status, 14, 5));

{ Get data from device }
    Rewrite(gpib);
    Writeln(gpib, 'ENTER 1#20');
    Reset(gpib);
    Readln(gpib, data);
    Writeln('You entered: ',data);

```

```
{ Perform a serial poll }
  Rewrite(gpib);
  Writeln(gpib, 'SPOLL 1');
  Reset(gpib);
  Readln(gpib, data);
  Writeln('Poll byte: ',data, ' (decimal)');

  Close(gpib);
end.
```

Appendix

Customer Communication

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

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

Corporate Headquarters

(512) 795-8248

Technical support fax: (800) 328-2203
(512) 794-5678

Branch Offices	Phone Number	Fax Number
Australia	(03) 879 9422	(03) 879 9179
Austria	(0662) 435986	(0662) 437010-19
Belgium	02/757.00.20	02/757.03.11
Denmark	45 76 26 00	45 76 71 11
Finland	(90) 527 2321	(90) 502 2930
France	(1) 48 14 24 00	(1) 48 14 24 14
Germany	089/741 31 30	089/714 60 35
Italy	02/48301892	02/48301915
Japan	(03) 3788-1921	(03) 3788-1923
Netherlands	03480-33466	03480-30673
Norway	32-848400	32-848600
Spain	(91) 640 0085	(91) 640 0533
Sweden	08-730 49 70	08-730 43 70
Switzerland	056/20 51 51	056/20 51 55
U.K.	0635 523545	0635 523154

Technical Support Form

Technical support is available at any time by fax. Include the information from your configuration form. Use additional pages if necessary.

Name _____

Company _____

Address _____

Fax (____) _____ Phone (____) _____

Computer brand _____

Model _____ Processor _____

Operating system _____

Speed _____MHz RAM _____M

Display adapter _____

Mouse _____yes _____no

Other adapters installed _____

Hard disk capacity _____M Brand _____

Instruments used _____

National Instruments hardware product model _____

Rev. _____

Configuration _____

National Instruments software product _____

Ver. _____

Configuration _____

(continues)

The problem is _____

List any error messages _____

The following steps will reproduce the problem _____
