NI-488.2 Features

2025-03-20

Contents

Programming with NI-488.2 Software

This document explains the philosophy and structure of the National Instruments IEEE 488.2 software for personal computers. It describes the software functions used in writing an NI-488.2 program and the actual process of developing an application program. It also describes utility programs you can use to shorten development time.

Introduction

NI-488.2 is the National Instruments IEEE 488 software. NI-488.2 software, which has been the de facto industry standard for many years, is a high-speed driver with utilities that help in developing and debugging an application program. Because the NI-488.2 driver has high-level commands that automatically handle all bus management, you do not need to learn the programming details of the GPIB hardware board or the IEEE 488.2 protocol. Low-level commands are also available for maximum flexibility and performance.

The NI-488.2 driver was the first loadable IEEE 488 device driver for MS-DOS-based personal computers. It is available for computer platforms including PC compatibles, PS/2, Macintosh, Sun, DEC, HP, and Silicon Graphics, running under operating systems such as DOS, Windows, Windows 2000/XP/Vista, Windows NT, Mac OS, OS/2, UNIX, Solaris, OSF/1, and IRIX. NI-488.2 programs are portable across these different platforms. Major companies such as IBM, Tektronix, Philips, LeCroy, Howtek, Sharp, Perkin-Elmer, Instron, Bruel & Kjaer, Hitachi Nakaworks, and Advantest all use our NI-488.2 software to produce innovative, high-quality products. This large backing has established NI-488.2 as the de facto industry standard.

Subroutine-Structured Driver

The NI-488.2 driver is subroutine structured. This type of driver structure includes special subroutines already programmed by a vendor. A subroutine-structured driver gets its name because these subroutines are called as routines or functions from the programming language in which the application program was written. National Instruments chose a subroutine structure for the NI-488.2 driver rather than a

character-I/O structure, which was chosen by some other vendors. The subroutine structure is faster, easily handles buffered DMA transfers, and uses a structured, hierarchical programming style familiar to users of modern programming languages. National Instruments also offers a character-I/O driver, because it can be helpful for some applications.

The NI-488 driver functions have the following format:

```
ibfunction (ud, parameter list)
```

where `ibfunction` is the NI-488 function name (for example, `ibrd` for read, `ibwrt` for write, and `ibrsp` for request serial poll), `ud` is the unit descriptor for the device or board accessed by the function (for example, `DM5008` or simply `dmm` for a Tektronix digital multimeter), and `parameter list` is the list of arguments for the particular `ibfunction` (for example, a buffer pointer and a count for `ibrd` and `ibwrt`). The parameter list varies slightly for each programming language, but the function names remain constant. There are more than 30 NI-488 functions to cover all IEEE 488 bus management functions.

The NI-488.2 driver contains an additional set of routines. These NI-488.2 routines have the following format:

```
routine (board, parameter list)
```

where `routine` is the NI-488.2 routine name (for example, `FindLstn` for Find All Listeners), `board` is the board number that the routine is accessing, and `parameter list` is the list of arguments for the particular routine. There are more than 20 NI-488.2 routines, which cover all IEEE 488.2 bus management functions.

NI-488.2 -- Structured Programming

Structured, hierarchical programs use subroutines or functions. This is a preferred and familiar style to programmers using compiled programming languages such as Visual Basic, C, FORTRAN, and Pascal.

NI-488 Functions

NI-488.2 software meets a wide spectrum of needs, from low-end applications to the

most sophisticated tasks. NI-488.2 software has two levels of functions—high-level functions for ease-of-use and low-level functions for maximum flexibility and performance.

High-Level Functions

High-level functions hide the IEEE 488 protocol by automatically calling a sequence of NI-488.2 low-level functions. High-level functions access a specific device and take care of the addressing and bus management protocol for that device. NI-488.2 software includes a complete set of high-level functions. You can also build your own customized high-level functions. The high-level functions can combine several GPIB operations, such as sending interface clear (IFC) or remote enable (REN), into one subroutine.

Low-Level Functions

Low-level calls give you the flexibility to make one command control multiple devices or to change the address status of instruments. If you understand the GPIB protocol, you can use low-level functions to control the interface board and the GPIB directly.

The following table shows a Visual Basic language comparison of the number of low-level functions that make up one high-level function used to serial poll a specified driver.

Table 1. Comparison of NI-488 High-Level and Low-Level Functions in Visual Basic

High-Level Function	Low-Level Function

`CALL ibrsp (dvm%, status%)`	`cmd$ = "?" + chr$(&H18) + "G!"`
	`CALL ibcmd (gpib0%, cmd$)`
	`status$ = space$(1)`
	`CALL ibrd (gpib0%, status$)`
	`cmd$ = "_?" + chr$(&H19)`
	`CALL ibcmd (gpib0%, cmd$)`

NI-488.2 Routines

The NI-488.2 routines consist of high-level routines and low-level routines. More accurately, these routines are in the following groups:

- Simple Device I/O
- Multiple Device I/O
- Multiple Device Control
- Bus Management
- Low-Level I/O

You can use the NI-488.2 routines with compliant 488.2-compatible devices to achieve greater predictability of instrument behavior and programming correctness, and increased programming similarity between the instruments from different manufacturers.

Language Interface

When you use functions and routines designed by National Instruments to access the NI-488.2 driver, you must define them to the programming language you are using. Each application program includes a declaration file, which defines the proper use of parameters for the functions and routines (such as `vbib-32.bas` for Visual Basic, `windecl.h` for Windows C programs, and `decl-32.h` for Windows 95 C programs).

Because NI-488.2 bypasses operating system entry points, the program must also interface with the device driver. To interface the program with the device driver, you must use a language interface written by National Instruments. This language interface is specific to a particular programming language. It can link to a compiled language program or load into an interpretive language program. The language interface first locates and opens the NI-488.2 driver. Then, it maps the subroutine calling conventions of the programming language to the calling conventions expected by the NI-488.2 driver. This process of including a declaration file and linking files to the compiled program is familiar to compiled language users. Interpretive language users, however, must load the language interface at the beginning of a program.

The following table contains a complete list of the Visual Basic NI-488 functions, their parameters, and a short description of each.

CALL Syntax	Description
`ibask (ud%, option%, value%)`	Checks current configuration parameters
`ibcac (ud%, v%)`	Become Active Controller
`ibclr (ud%)`	Clear specified device
`ibcmd (ud%, cmdbuf$)`	Send commands from string
`ibcmda (ud%, cmdbuf$)`	Send commands asynchronously from string
`ibconfig (ud%, option%, value%)`	Set current configuration parameters
`ibdev (BdIndx%, pad%, sad%, tmo%, eot%, eos%, ud%)`	Open and initialize a device descriptor
`ibdma (ud%,v%)`	Enable/disable DMA
`ibeos (ud%,v%)`	Change/disable EOS mode
`ibeot (ud%,v%)`	Enable/disable END message
`ibfind (udname$,ud%)`	Open device and return unit descriptor
`ibgts (ud%,v%)`	Go from Active Controller to standby
`ibist (ud%,v%)`	Set/clear ist
`iblines (ud%, clines%)`	Returns status of the GPIB control lines
`ibln (ud%,pad%,sad%,listen%)`	Check for the presence of a device on the bus
`ibloc (ud%)`	Go to local
`ibonl (ud%,v%)`	Place device online/offline
`ibpad (ud%,v%)`	Change primary address
`ibpct (ud%)`	Pass control
`ibppc (ud%,v%)`	Parallel poll configure
`ibrd (ud%, rdbuf$)`	Read data to string
`ibrda (ud%, rdbuf$)`	Read data asynchronously to string
`ibrdf (ud%,flname$)`	Read data to file
`ibrpp (ud%,ppr%)`	Conduct a parallel poll
`ibrsc (ud%,v%)`	Request/release system control
`ibrsp (ud%,spr%)`	Return serial poll byte

CALL Syntax	Description
`ibrsv (ud%,v%)`	Request service
`ibsad (ud%,v%)`	Change secondary address
`ibsic (ud%)`	Send interface clear
`ibsre (ud%,v%)`	Set/clear remote enable line
`ibstop (ud%)`	Abort asynchronous operation
`ibtmo (ud%,v%)`	Change/disable time limit
`ibtrg (ud%)`	Trigger selected device
`ibwait (ud%,mask%)`	Wait for selected event
`ibwrt (ud%, wrtbuf$)`	Write data from string
`ibwrta (ud%, wrtbuf$)`	Write data asynchronously from string
`ibwrtf (ud%,flname$)`	Write data from file

The following table contains a complete list of the Visual Basic NI-488.2 routines, their parameters, and a short description of each.

Table 2. Visual Basic NI-488.2 Routines

`ibstop (ud%)`	Abort asynchronous operation
`ibtmo (ud%, v%)`	Change/disable time limit
`ibtrg (ud%)`	Trigger selected device
`ibwait (ud%, mask%)`	Wait for selected event
`ibwrt (ud&, sstr, cnt&)`	Write data from string
`ibwrta (ud&, sstr, cnt&)`	Write data asynchronously from string
`ibwrtf (ud%, flname$)`	Write data from file
`AllSpoll (board%, addresslist%(0), resultlist%(0))`	Serial poll all devices
`DevClear (board%, address%)`	Clear a single device
`DevClearList (board%, addresslist%(0))`	Clear multiple devices
`EnableLocal (board%, addresslist%(0))`	Enable operations from the front of a device

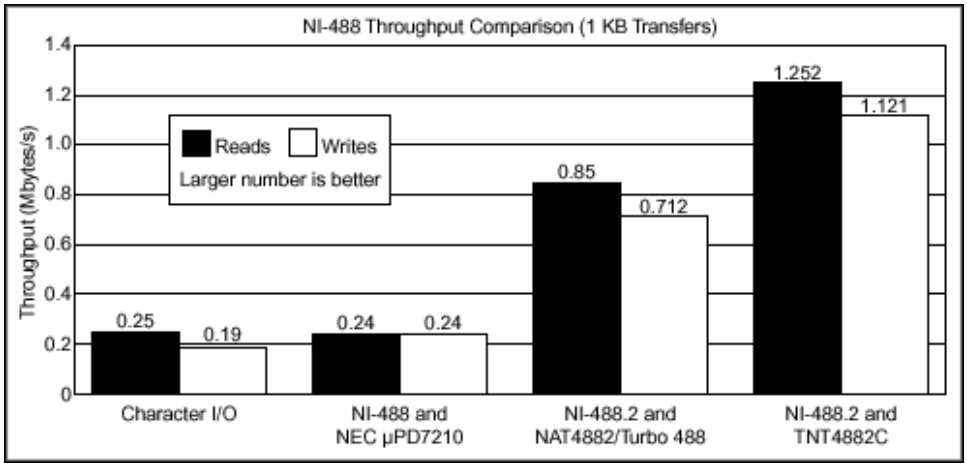
`EnableRemote (board%, addresslist%(0))`	Enable remote GPIB programming of devices
`FindLstn (board%,addresslist%(0),resultlist%(0), limit%)`	Find all Listeners
`FindRQS (board%, addresslist%(0), result%)`	Determine which device is requesting service
`PassControl (board%, address%)`	Pass control to another device with Controller capability
`PPoll (board%, result%)`	Perform a parallel poll
`PPollConfig (board%, address%, dataline%, sense%)`	Configure a device for parallel polls
`PPollUnconfig (board%, addresslist%(0))`	Unconfigure devices for parallel polls
`RcvRespMsg (board%, data$, termination%)`	Read data bytes from already addressed device
`ReadStatusByte (board%, address%, result%)`	Serial poll a single device to get its status byte
`Receive (board%, address%, data$, termination%)`	Read data bytes from a GPIB device
`ReceiveSetup (board%, address%)`	Prepare a particular device to send data bytes and prepare the GPIB board to read them
`ResetSys (board%, addresslist%(0))`	Initialize a GPIB system on three levels
`Send (board%, address%, data$, eotmode%)`	Send data bytes to a single GPIB device
`SendCmds (board%, commands$)`	Send GPIB command bytes
`SendDataBytes (board%, data$, eotmode%)`	Send data bytes to already addressed devices
`SendIFC (board%)`	Clear the GPIB interface functions with IFC
`SendList (board%, addresslist%(0), data$, eotmode%)`	Send data bytes to multiple GPIB devices
`SendLLO (board%)`	Send the local lockout message to all

	devices
`SendSetUp (board%, addresslist%(0))`	Prepare particular devices to receive data bytes
`SetRWLS (board%, addresslist%)`	Place particular devices in the Remote with Lockout state
`TestSRQ (board%, result%)`	Determine the current state of the SRQ line
`TestSys (board%, addresslist%, resultlist%(0))`	Cause devices to conduct self-tests
`Trigger (board%, address%)`	Trigger a single device
`Triggerlist (board%, addresslist%(0))`	Trigger multiple devices
`WaitSRQ (board%, result%)`	Wait until a device asserts Service Request

Performance

With NI-488.2 software, a separate routine or function call uniquely identifies the subroutine. You can use the `ibfind` function to return a descriptor the operating system defines for each GPIB interface board and device in a system. By using NI-488 functions or NI-488.2 routines to directly access a particular board or device, you can bypass the operating system and reduce overhead. The following figure compares the performance of a character-I/O driver, our NI-488 driver running on the NEC µPD7210 Controller chip, our NI-488.2 driver running on the NAT4882™ Controller chip, and our NI-488.2 driver running on the TNT4882C Controller chip.

Figure 1. NI-488 Throughput Comparison



GPIB hardware interfaces equipped with the NAT4882 and Turbo488™ ASICs can transfer data at rates exceeding 1 Mbytes/s for reads and writes. TNT4882C-based interfaces can attain IEEE 488.1 transfer rates of 1.5 Mbytes/s and HS488 transfer rates up to 8 Mbytes/s. These chips increase efficiency by moving time-consuming software driver functions into hardware. The NAT4882 and Turbo488 are featured on boards for ISA, MIcro Channel, Macintosh LC bus, and DEC TURBOchannel. The TNT4882C is featured on boards for PCI, ISA, EISA, PC Card (PCMCIA), Macintosh NuBus, Sun SBus, and NEC bus.

Buffered Transfers

Buffered data transfers are easy to accomplish with a subroutine-structured driver. A buffered data transfer is a transfer of many data values between a device and computer memory. This capability is important for instruments that deal with large arrays of data, such as digitizers and spectrum analyzers. Digitizers are the fastest growing instruments, particularly with the high-powered analysis capabilities of modern PCs and software.

The NI-488 software makes buffered data transfers easy and transparent. The software can accomplish these transfers with only one command. This is done by specifying one of the parameters of an NI-488 function or NI-488.2 routine as an array, string, or file. Then data is read into or written from that particular array, string, or file. Buffered data transfers are much easier to program in instruments with a subroutine-structured driver than with a character-I/O driver. The following table illustrates this point.

Table 3. Comparison of the NI-488.2 Driver Code and Character-I/O Driver Code

NI-488.2 Driver	Character-I/O Driver

`CALL ibrd (scope%,wvfrm$)`
 	
`or`
 	
`CALL Receive (0,1,wvfrm$,STOPend)`
	`DEF FNgetwd(addr)=PEEK(addr) +`
	`  256 * PEEK(addr + 1)`
	`ds% = VARSEG(wvfrm$)`
	`rdesc = 0`
	`rdesc = VARPTR(wvfrm$)`
	`PRINT #1, "ENTER 01 #1024 BUFFER";`
	`  ds%;":";FNgetwd(rdesc + 2);"DMA"`

NI-488.2 Program Examples

The example programs shown below use high-level (device) functions and routines that automatically handle the details of GPIB protocol. These programs first configure the voltage type, voltage range, and speed of a multimeter, then perform a serial poll, take a voltage reading, and print the reading.

Using NI-488 Functions	Using NI-488.2 Routines

Visual Basic
 	
`Private Sub Command1_Click()`
`VOLT$ = SPACE$ (13)`
`CALL IBFIND ("DMM", DMM%)`
`CALL IBWRT (DMM%, "*RST; VDC; RATE F",`
`17)`
`CALL IBRSP (DMM%, SPR%)`
`CALL IBRD (DMM%, VOLT$, 100)`
`TextDisplay = VOLT$`
`End`
	Visual Basic
	
	`Private Sub Command1_Click()`
	`VOLT$ = SPACE$ (13)`
	`CALL Send (0,1, "*RST; VDC; RATE F,`
	`  MEAS1?", NLend)`
	`CALL ReadStatusByte (0, 1, spr%)`
	`CALL Receive (0, 1, VOLT$, STOPend)`
	`TextDisplay = VOLT$`
	`End`

C
 	
`#include <stdio.h>`
`#include "decl-32.h"`
`main () {`
	C
	
	`#include <stdio.h>`
	`#include "decl-32.h"`
	`main () {`

Using NI-488 Functions	Using NI-488.2 Routines

`char int dmm; spr; volt [13];`
`dmm = ibfind ("DMM");`
`ibwrt (dmm, "F0R0S2", 6);`
`ibrsp (dmm, &spr);`
`ibrd (dmm, volt, 13);`
`printf ("%s", volt);`
`}`
	`int spr;`
	`char volt [13];`
	`Send (0, 1, "FOR0S2", 6, NLEND);`
	`ReadStatusByte (0, 1, &spr);`
	`Receive (0, 1, volt, 13,`
	`STOPend);`
	`printf ('%s", volt);`
	`}`

After compiling these programs, you must link the object code with the appropriate NI-488 language interface to create your stand-alone executable program.

Utilities

The NI-488 software includes several useful utilities designed to shorten development time. These utilities can help you quickly develop and debug your application programs.

Interactive Control Program

With the Interface Bus Interactive Control (IBIC) program, you can communicate with GPIB devices through NI-488 functions and NI-488.2 routines. This program is a powerful development and debugging tool. You can use it to learn the NI-488 functions and routines, learn the device-specific messages of an instrument, debug an application program one step at a time, or locate a malfunctioning device on the GPIB.

NI-488.2 Communicator

You can use the NI-488.2 Communicator to verify that you can establish simple communication with your GPIB instrument. This is an interactive utility that allows you to write commands to your instrument and read responses back from your instrument. It provides detailed information about the status of the NI-488.2 calls and you can use it to print sample C source code that performs a simple query to a GPIB instrument.

Configuration Utility

The configuration utility is a menu-driven program in which you can customize software parameters such as the device name, address, message termination mode, and timeout limit for use with functions and routines. With a configuration utility, you do not have to specify these parameters in each program and you can use names and mnemonics in your program to reference actual GPIB devices.

NI I/O Trace

Using NI I/O Trace, the user can “trace” driver calls. This feature is extremely useful for application debugging. The NI I/O Trace records all device and board level calls with a time stamp. Developers can easily and efficiently detect errors and timing issues in their applications. This utility further differentiates National Instruments GPIB device functionality and usability above all others.

GPIB Analyzer

With GPIB Analyzer, you can analyze the physical bus activity by observing all of the GPIB handshake signals, interface management signals, and data signals. This capability is useful for advanced debugging where NI I/O Trace alone cannot resolve the problems. In addition, you can analyze bus timing issues in much more detail. The GPIB Analyzer software is available with only NI GPIB Analyzer boards (PCIe/PCI-GPIB+). No other GPIB supplier provides similar functionality.

Diagnostic Programs

All NI products include hardware and software diagnostic programs to help you check for hardware conflicts and verify proper software installation.

Summary

The NI-488.2 software is designed for high performance and maximum flexibility. Many options are available with the NI-488.2 software. You do not need to learn the programming details of the GPIB interface board or IEEE 488.2 protocol. If you are already familiar with the IEEE 488.2 protocol, NI-488.2 software gives you low-level functionality, thereby giving you maximum flexibility and performance. This document has described the NI-488.2 software features you can use to achieve performance and minimize development time.

Related Links

Products and Services: GPIB, Serial, and Ethernet