Controlling Instruments with HP VEE

Notice

The information contained in this document is subject to change without notice.

Hewlett-Packard Company (HP) shall not be liable for any errors contained in this document. *HP makes no warranties of any kind with regard to this document, whether express or implied. HP specifically disclaims the implied warranties of merchantability and fitness for a particular purpose.* HP shall not be liable for any direct, indirect, special, incidental, or consequential damages, whether based on contract, tort, or any other legal theory, in connection with the furnishing of this document or the use of the information in this document.

Warranty Information

A copy of the specific warranty terms applicable to your Hewlett-Packard product and replacement parts can be obtained from your local Sales and Service Office.

U.S. Government Restricted Rights

The Software and Documentation have been developed entirely at private expense. They are delivered and licensed as "commercial computer software" as defined in DFARS 252.227-7013 (Oct 1988), DFARS 252.211-7015 (May 1991) or DFARS 252.227-7014 (Jun 1995), as a "commercial item" as defined in FAR 2.101(a), or as "Restricted computer software" as defined in FAR 52.227-19 (Jun 1987)(or any equivalent agency regulation or contract clause), whichever is applicable. You have only those rights provided for such Software and Documentation by the applicable FAR or DFARS clause or the HP standard software agreement for the product involved.

Printing History

Edition 1 - March 1997

Conventions Used in this Manual

This manual uses the following typographical conventions:

Example	Represents
Getting Started with HP VEE	Italicized words are used for book titles and for emphasis.
`File`	Computer font represents text you will see on the screen, including menu names, features, buttons, toolbar button names, or text you have to enter.
`dir` *filename*	In this context, the word in computer font represents text you type exactly as shown, and the italicized word represents an argument that you must replace with an actual value.
`File` ⟹ `Open`	The "⟹" is used in a shorthand notation to show the location of HP VEE features in the menu. For example, "`File` ⟹ `Open`" means to select the `File` menu and then select `Open`.
`Flat` | `Sunken` | `Raised`	Choices in computer font, separated with a bar (|), indicate that you should choose one of the options.
Return	The keycap font graphically represents a key on the keyboard.
Press Ctrl+O	Represents a combination of keys on the keyboard that you should press at the same time.
Dialog Box	Bold font indicates the first instance of a key term.

Contents

5. Using VXIplug&play Drivers

Figures

Tables

Contents

1

Introduction

Introduction

This manual describes how to configure and control instruments from HP VEE. The chapters in this manual are described briefly below:

- Chapter 1, "Introduction," (this chapter) provides an overview of this manual, a table listing the supported I/O interfaces, and a list of related reading material.
- Chapter 2, "Instrument Control Fundamentals," provides an overview of the fundamentals of instrument control using HP VEE.
- Chapter 3, "Configuring Instruments," describes how to use the `Instrument Manager` and the configuration dialog boxes to configure instruments in HP VEE.
- Chapter 4, "Using Transactions in Direct I/O and Interface Operations," describes using transaction I/O to send commands, read data, and control interface operations.
- Chapter 5, "Using VXI*plug&play* Drivers," describes techniques for using VXI*plug&play* drivers in HP VEE.
- Chapter 6, "Using Panel Drivers and Component Drivers," describes techniques for using `Panel Driver` and `Component Driver` objects.
- Chapter 7, "Advanced Topics," provides additional information about I/O configuration and I/O control techniques.
- Appendix A, "Select Codes and I/O Addressing," provides reference information about the HP VEE I/O addressing scheme.
- Appendix B, "Troubleshooting," suggests some techniques for solving problems that you may encounter.
- Appendix C, "Instrument I/O Data Type Conversions," describes the automatic data type conversions that HP VEE performs on incoming data from instruments.

Supported I/O Interfaces

The following table lists the supported I/O interfaces for each platform.

NOTE

Before HP VEE can communicate with instruments, the computer running HP VEE must be properly configured and the I/O libraries must be installed as described in *Installing the HP I/O Libraries - HP VEE*. Also, refer to Appendix A in this manual for select code and I/O addressing information.

Table 1-1. Instrument I/O Support

Platform	Supported I/O Interfaces
Windows 95 (PC, HP 6232, HP 6233, EPC7/8)	• HP-IB or GPIB[1] • Serial • GPIO • VXI[2]
Windows NT (PC, HP 6232, HP 6233, EPC7/8)	• HP-IB or GPIB[1] • Serial • GPIO • VXI[2]
HP-UX (HP 9000 Series 700, V/743)	• HP-IB[1] • Serial • GPIO • VXI[3]

1 Can address VXI devices using HP E1406 Command Module.

2 Direct backplane access for embedded controllers: HP 6232 or HP 6233 VXI Pentium® Controller, HP RADI-EPC7/8 VXI Controller, or RadiSys EPC7/8 VXI Controller. Direct backplane access for external PCs using VXLink.

3 Direct backplane access for HP V/743 VXI Embedded Controller. Direct backplane access for external Series 700 using HP E1489C EISA/ISA-to-MXIbus interface.

Related Reading

For more detailed information about instrument control topics discussed in this manual, refer to the following publications.

- *Tutorial Description of the Hewlett-Packard Interface Bus* (Hewlett-Packard Company, 1987), part number 5021-1927.

  This document provides a condensed description of the important concepts contained in IEEE 488.1 and IEEE 488.2. If you are unfamiliar with the IEEE 488.1 interface, this is the best place to start.

- *IEEE Standard 488.1-1987, IEEE Standard Digital Interface for Programmable Instrumentation* (The Institute of Electrical and Electronics Engineers, 1987).

  This standard defines the technical details required to design and build an HP-IB (IEEE 488.1) interface. This standard contains electrical specifications and information on protocol that is beyond the needs of most programmers.

- *IEEE Standard 488.2-1987, IEEE Standard Codes, Formats, Protocols, and Common Commands For Use with ANSI/IEEE Std 488.1-1987* (The Institute of Electrical and Electronics Engineers, 1987).

  This document describes the underlying message formats and data types used by instruments that implement the Standard Commands for Programmable Instruments (SCPI).

- *IEEE Standard 728-1982, IEEE Recommended Practice For Code and Format Conventions For Use with ANSI/IEEE Std 488-1978, etc.* (The Institute of Electrical and Electronics Engineers, 1983).

- *VMEbus Extensions for Instrumentation*, including: "VXI-0, Rev. 1.0: Overview of VXIbus Specifications" and "VXI-1, Rev. 1.4: System Specification," VXIbus Consortium, Inc., 1992.

- *HP VISA User's Guide* (Hewlett-Packard Company, 1996), part number E2090-90105.

  This document is useful for those who create their own VXI*plug&play* drivers, and provides additional information about addressing and using VXI*plug&play* drivers.

2

Instrument Control Fundamentals

Instrument Control Fundamentals

Overview

HP VEE supports four types of objects for controlling instruments. Figure 2-1 shows one of each of these objects in its open view. (Each of these examples communicates with an HP E1410A VXI Multimeter.)

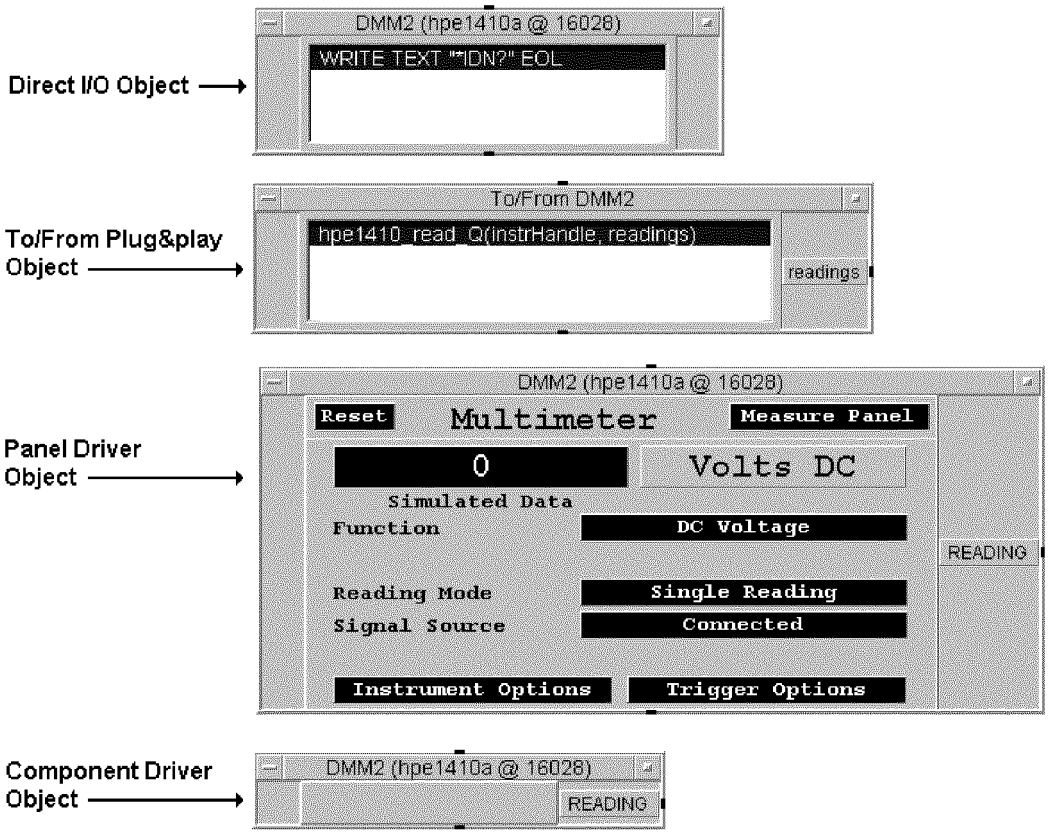

Figure 2-1. HP VEE Instrument Control Objects

Table 2-1 gives an overview of the differences between these instrument control objects.

Table 2-1. Comparing Instrument Control Objects in HP VEE

HP VEE Object	Instrument Access	Main Benefits	Supported Interfaces[1]
`Direct I/O`	Communicates directly with any instrument.	Fast I/O. Can control any instrument.	HP-IB or GPIB, Serial, GPIO, VXI, and LAN.
`To/From VXIplug&play`	Requires a VXI*plug&play* driver supplied from the instrument manufacturer specific to each platform. Requires VISA to be installed.	Fast I/O. Drivers can be used by multiple software applications.	HP-IB or GPIB, and VXI.
`Panel Driver`	Requires an HP Instrument Driver supplied with HP VEE.[2]	Easy to use.	HP-IB or GPIB, and VXI.
`Component Driver`	Requires an HP Instrument Driver supplied with HP VEE.	Faster I/O than `Panel Driver`.	HP-IB or GPIB, and VXI.

1 HP-IB is Hewlett-Packard's implementation of the IEEE-488 interface bus standard. Other implementations are called GPIB.

2 HP Instrument Drivers are also sometimes called "HP VEE drivers."

The `To/From VXIplug&play`, `Panel Driver` and `Component Driver` objects allow you to control instruments without learning the details of the instrument's programming mnemonics and syntax. If you prefer to communicate with your instruments by sending low-level mnemonics, or if a driver is not available for your instrument, you can use `Direct I/O`.

NOTE

You can use all four methods to communicate with different instruments within an HP VEE program. However, don't use VXI*plug&play* drivers along with any of the other methods to communicate with the same instrument in the same program—unexpected results may occur.

Introduction to Direct I/O

`Direct I/O` objects allow you to read and write arbitrary data to instruments in much the same way you read from and write to files. This allows you full access to any programmable feature of any instrument. No instrument driver file is required, but you must have a detailed understanding of your instrument's programming commands to use `Direct I/O`. Also, in order to use `Direct I/O` to communicate with HP-IB, GPIB, or VXI devices, the I/O libraries must be installed as described in *Installing the HP I/O Libraries - HP VEE*.

`Direct I/O` objects also provide convenient support for learn strings. A learn string is a special feature supported by some instruments that allows you to set up measurement states from the front panel of the physical instrument. Once the instrument is configured, you simply select `Upload` from the `Direct I/O` object menu to upload the entire measurement state of the instrument to HP VEE. You can recall the measurement state from within your program by using the `Direct I/O` object.

An Example of Direct I/O

Let's look at a simple example using the `Direct I/O` object. Figure 2-2 shows a `Direct I/O` object set up to obtain the idenfication string from an HP 34401A Multimeter:

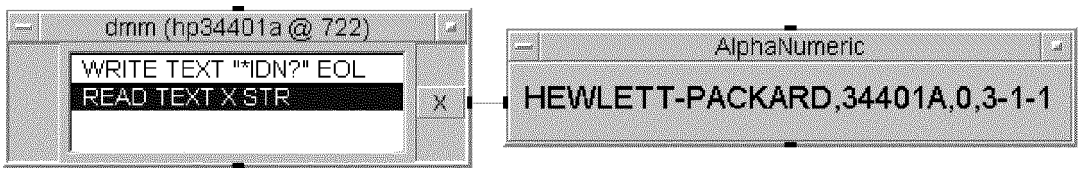

Figure 2-2. Using Direct I/O to Identify an Instrument

The first transaction in the `Direct I/O` object writes the text string `*IDN?` to the HP 34401A at HP-IB address `722`. This causes the HP 34401A to send the identification string, which is read by the second transaction and output to the `Alphanumeric` object.

For information about how to *configure* HP VEE to use `Direct I/O`, refer to Chapter 3. For details about how to *use* the `Direct I/O` object, refer to Chapter 4.

MultiDevice Direct I/O

The `MultiDevice Direct I/O` object lets you control several instruments from a single object using direct I/O transactions. This object appears the same as the `Direct I/O` object, except each transaction in the `MultiDevice Direct I/O` object can address a separate instrument. The object is a standard transaction object, and works with all interfaces that HP VEE supports. Since the `MultiDevice Direct I/O` object does not necessarily control a single instrument, the title does not list an instrument name, address, or live mode condition.

By using the `MultiDevice Direct I/O`, you can reduce the number of instrument-specific `Direct I/O` objects in your program. The resulting performance increase is especially important for the VXI interface, which is faster than HP-IB (GPIB) at instrument control. The following figure shows the `MultiDevice Direct I/O` object and its `I/O Transaction` dialog box. The object is being set up to communicate with an HP E1413B, HP E1328, and HP 3325.

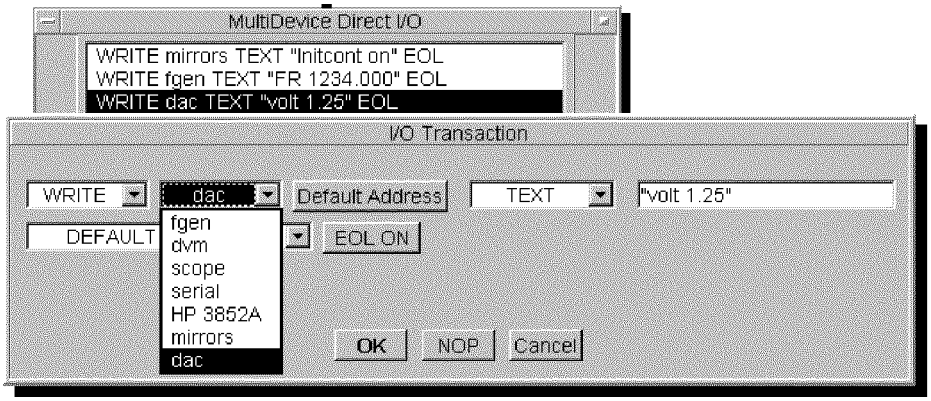

Figure 2-3. MultiDevice Direct I/O Controlling Several Instruments

For further information about using the `MultiDevice Direct I/O` object, refer to "Using the MultiDevice Direct I/O Object" in Chapter 4.

Introduction to VXIplug&play

VXI*plug&play* is an interface specification that allows multiple vendors to supply compatible hardware and software. A VXI*plug&play* driver is a library of functions for controlling a specific instrument. The driver is written by the hardware vendor and shipped with the instrument.

HP VEE version 4.0 supports drivers that comply with the WIN95, WINNT, or HP-UX framework, VXI*plug&play* specification version 3.0 or later. The HP-UX framework supports HP-UX version 10.x and above.

Getting Started

Before you can get started with VXI*plug&play*, you must have already completed these steps:

1. Install the interface (HP-IB, GPIB, or VXI).
2. Install VISA. If you're using an HP interface card, use VISA as supplied with HP VEE. Refer to *Installing the HP I/O Libraries - HP VEE* for details. Otherwise, you must install VISA as supplied with the interface card.
3. Configure VISA for each hardware interface. If you're using an HP interface card, follow the instructions in *Installing the HP I/O Libraries - HP VEE*. Otherwise, you must configure VISA as specified by the interface manufacturer.

NOTE

VISA (Virtual Instrument Software Architecture) is an I/O library that VXI*plug&play* drivers use to control instruments. VISA is required for VXI*plug&play* and provides VISA function calls, which are used by the VXI*plug&play* drivers.

What You Need

HP VEE needs these four files for each VXI*plug&play* driver.

- The library file
- The function panel file
- The header file
- The help file

The files installed with each VXI*plug&play* driver always include these files. Other files are also installed.

Note that not all VXI*plug&play* drivers support all frameworks. Also certain versions of VISA may not be supported on all frameworks. Please check with the appropriate vendor.

Installing the VXI*plug&play* Driver Software

To install the set of files needed for each driver, follow the instructions included with the driver by the instrument manufacturer.

Location of Files (WIN95 and WINNT Frameworks)

The VXI*plug&play* files are located under the `WIN95\` directory or the `WINNT\` directory. This location is relative to the root drive and directory value stored in the registry by the VISA installation. The default value for the root drive and directory is `C:\VXIPNP`.

These are the VXI*plug&play* driver files needed by HP VEE:

Table 2-2. Location of WIN95 and WINNT Framework Driver Files

Filename[1]	Location	Purpose
PREFIX`_32.DLL`	`BIN`	Instrument driver library
PREFIX`.FP`	*PREFIX*	Instrument driver function panel file
PREFIX`.H`	`INCLUDE`	Instrument driver header file
PREFIX`.HLP`	*PREFIX*	Instrument driver help file

1 *PREFIX* refers to the name of the instrument such as `HPE1410`.

Location of Files (HP-UX Framework)

The VXI*plug&play* files are located under the `vxipnp/hpux/` directory. This location is relative to the root directory represented by the environment variable `VXIPNPPATH`. This environment variable is set to `/opt` by default, so the directory is normally `/opt/vxipnp/hpux/`.

These are the VXI*plug&play* driver files needed by HP VEE:

Table 2-3. Location of HP-UX Framework Driver Files

Filename[1]	Location	Purpose
PREFIX`.sl`	`bin`	Instrument driver library
PREFIX`.fp`	*PREFIX*	Instrument driver function panel file
PREFIX`.h`	`include`	Instrument driver header file
PREFIX`.hlp`	*PREFIX*	Instrument driver help file

1 *PREFIX* refers to the name of the instrument such as HPE1410.

Summary of Terminology

Working with VXI*plug&play* drivers is different than using other types of I/O with HP VEE. Here is a summary of how the different pieces fit together.

- The HP VEE program calls VXI*plug&play* functions.
- The functions (that have parameters that may be set via function panels) are part of the VXI*plug&play* driver. The functions talk to the instrument through the VISA software.
- The instrument passes data back through VISA and into the function parameters.

A VXI*plug&play* Example Program

The following is a simple example program that uses the `To/From VXIplug&play` object to initiate a voltage measurement, and to obtain a reading from the HP E1410A Multimeter.

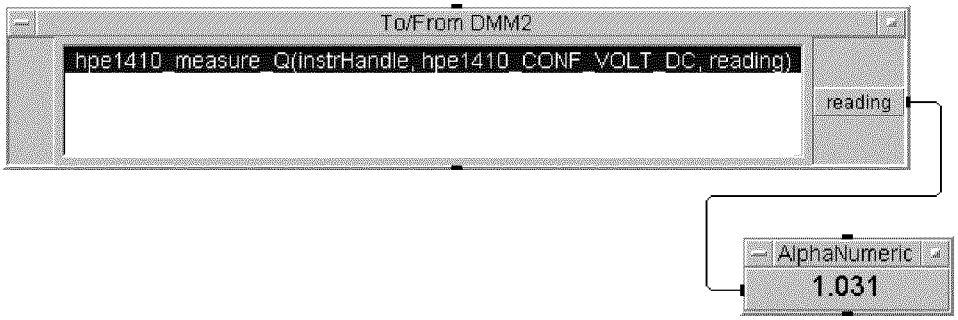

Figure 2-4. Using the To/From VXIplug&play Driver Object

Further Information

For information about how to *configure* HP VEE to use VXI*plug&play*, refer to Chapter 3. For further information about how to *use* VXI*plug&play* in HP VEE, refer to Chapter 5.

Introduction to Panel Drivers and Component Drivers

`Panel Driver` and `Component Driver` objects can be used for a particular instrument only if there is a **driver file** to support that instrument. The installation procedure for HP VEE for HP-UX automatically copies all of the available driver files onto your system disk. The installation procedure for HP VEE for Windows 95 and Windows NT allows you to select which drivers you want to install. Chapter 3 describes how to select and configure the proper driver files for your instruments. Also, the I/O libraries must be installed as described in *Installing the HP I/O Libraries - HP VEE*.

Panel Drivers

`Panel Drivers` serve two purposes in HP VEE:

- They allow you to define a measurement state that specifies all the instrument function settings. When a `Panel Driver` operates, the corresponding physical instrument is automatically programmed to match the settings defined in the `Panel Driver`.
- They act as instrument control panels for interactively controlling instruments. This is useful during development and debugging of your programs. It is also useful when your instruments do not have a physical front panel.

As shown in Figure 2-1, the open-view of a `Panel Driver` contains a graphical control panel for the associated physical instrument. If the physical instrument is properly connected to your computer, you can control the instrument by clicking on the fields in the graphical control panel. You can also make measurements and display the results by clicking on numeric and XY displays.

Even if the instrument is not connected to your computer, you can still use the graphical panel to define a measurement state. In fact, this can be a great benefit if you wish to develop programs before instruments are purchased or while they are being used elsewhere.

For example, suppose you want to program the HP 3325B function generator to provide two different output signals:

1. A square wave with a frequency of 20kHz and an amplitude of 20mV rms
2. A sine wave with a frequency of 50kHz and an amplitude of 50mV rms

Figure 2-5 shows the two **Panel Drivers** that provide the desired signals.

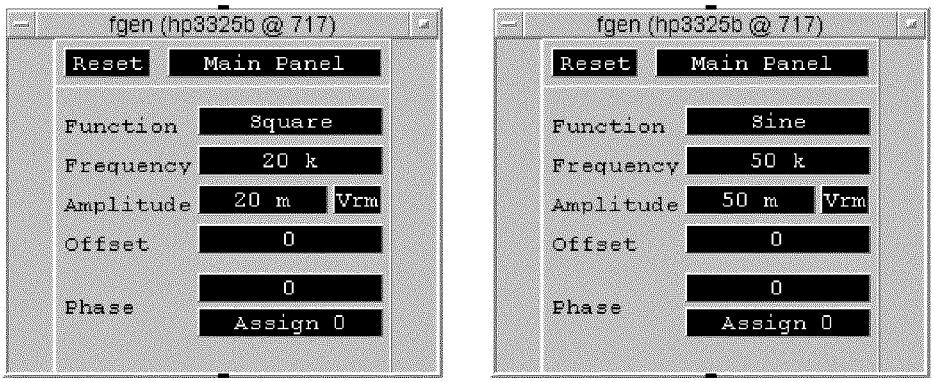

Figure 2-5. Two Panel Drivers

Component Drivers

In an HP instrument driver, each instrument function and measured value is called a **component**. A component is like a variable inside the driver that records the function setting or measured value. Thus, a **Component Driver** is an object that reads or writes only the components you specify as input and output terminals. This is in contrast to a **Panel Driver**, which automatically writes values for many or all components.

Component Drivers are provided to help you improve the execution speed of your program; speed is the only advantage they provide over **Panel Drivers.** The execution speed of a program is generally impacted most when an instrument control object is attached to an iterator object where it must operate many times. In these cases, it is common for only one or two components to be changing; this is exactly the situation **Component Drivers** are designed to handle.

The increase in execution speed provided by a **Component Driver** will vary considerably from one situation to another. The increase depends primarily on the particular driver file used. There is no easy way to predict the exact increase in execution speed.

For example, suppose you want to program the HP 3325B Function Generator to do the following:

1. Output a sine wave with an initial frequency of 10 kHz and an amplitude determined by operator input.
2. Sweep the frequency output from 10 kHz to 1 MHz using 5 steps per decade.

In this case, it makes sense to use a `Panel Driver` to perform the initial setup and a `Component Driver` to repeatedly set the output frequency. Figure 2-6 shows a program that does this.

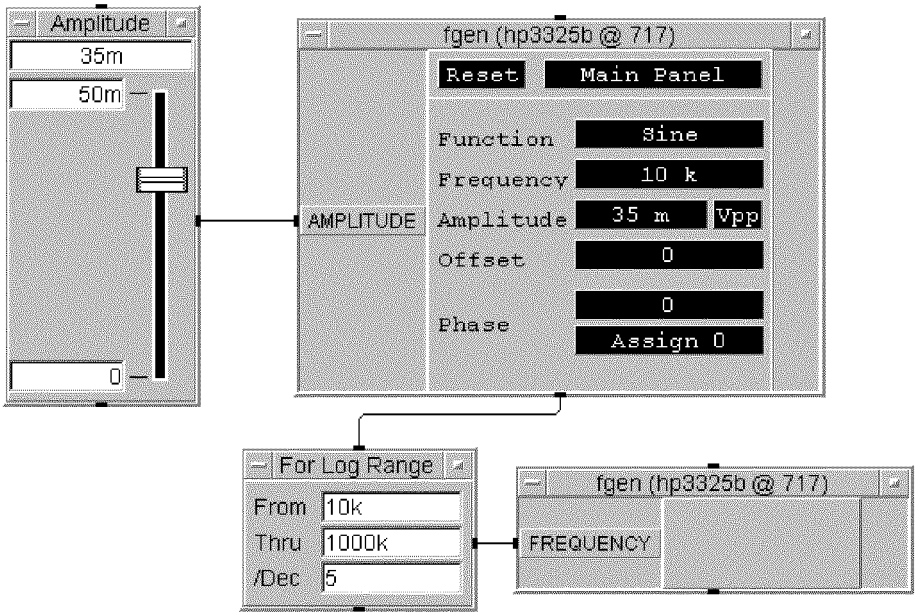

Figure 2-6. Combining Panel Drivers and Component Drivers

Further Information

For information about how to *configure* HP VEE, refer to Chapter 3. For further information about how to *use* the `Panel Driver` and `Component Driver` objects, refer to Chapter 6.

Support For Register-Based VXI Devices

When using the instrument control objects to directly address VXI devices on the VXI backplane, you need to know whether devices are **message-based** or **register-based**. HP VEE communicates with message-based devices by means of SCPI (Standard Commands for Programmable Instruments) messages. HP VEE also provides Interpreted SCPI (I-SCPI) support for most Hewlett-Packard register-based devices. I-SCPI drivers let you communicate with register-based devices as though they are message-based. This means that an HP VEE program can communicate with a register-based device using standard SCPI messages, provided there is an I-SCPI driver for that particular device. If no I-SCPI driver is available for a register-based device, HP VEE must communicate with that device by directly accessing its registers.

The I-SCPI drivers give you the flexibility to use any of the instrument control objects you prefer. You can use the `Panel Driver` for easier programming, or use SCPI commands in `Direct I/O` for faster execution speed. When you program HP VEE to communicate with a register-based device using SCPI messages, HP VEE will inform you if the required I-SCPI driver is not available. In that case, you need to access the device registers directly using `Direct I/O` or `MultiDevice Direct I/O`.

NOTE

I-SCPI is supported only for HP-UX and Windows 95. I-SCPI is not supported for Windows NT.

HP VEE includes a number of examples that are copied to your system disk automatically when you install HP VEE. In addition, the first time you execute HP VEE, it copies a default instrument configuration file to your home directory. *You must have this I/O con figuration to open the examples involving instruments.*

You can always configure additional instruments, but do not delete the entries in Figure 2-7 from the I/O configuration if you want to open the instrument examples:

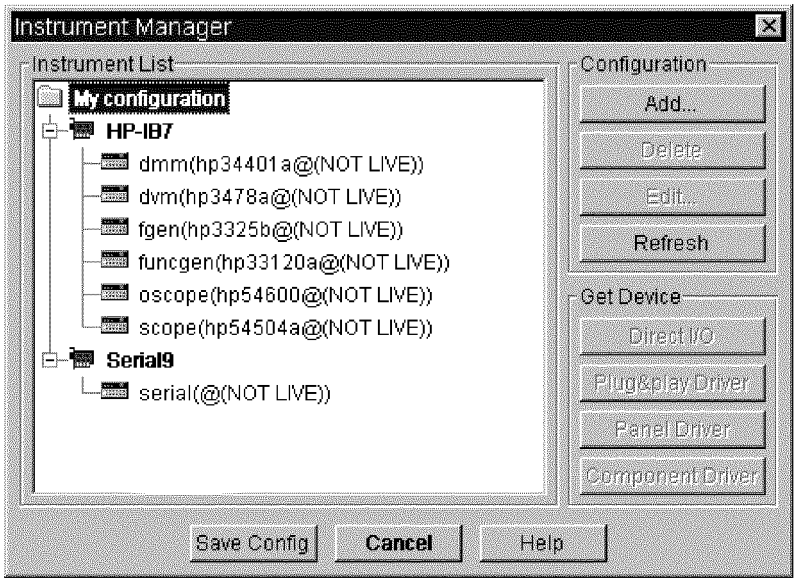

Figure 2-7. Default I/O Configuration

If HP VEE reports errors when you attempt to load the example programs referenced in this manual, please refer to the section "Running Example Programs" in Chapter 7.

3

Configuring Instruments

Configuring Instruments

This chapter shows how to configure HP VEE to communicate with your instruments using the following methods:

1. By means of `Direct I/O` objects (no instrument driver is required).
2. By means of VXI*plug&play* drivers using `To/From VXIplug&play` objects.
3. By means of HP Instrument Drivers ("IDs") using either `Panel Driver` or `Component Driver` objects.

The HP VEE `Instrument Manager` dialog provides a unified method to select and configure all of these instrument-control objects.

NOTE

In order for HP VEE to communicate with instruments, you must first install the HP I/O Libraries as described in *Installing the HP I/O Libraries - HP VEE*. You must install the HP SICL libraries in order to use `Panel Driver`, `Component Driver`, or `Direct I/O` objects. You must install the VISA libraries in order to use `To/From VXIplug&play` objects.

Also, in order to use `Panel Driver` or `Component Driver` objects, you must install the appropriate HP Instrument Drivers. For HP VEE for HP-UX, the drivers are automatically installed as part of the HP VEE installation. For HP VEE for Windows, you can install any desired selection of HP Instrument Drivers during the HP VEE installation. (No instrument drivers are required for `Direct I/O` objects.)

VXI*plug&play* drivers are supplied by the instrument manufacturer with many VXI instruments. In order to use a `To/From VXIplug&play` object, you must install the appropriate VXI*plug&play* driver files, following the instructions provided with the driver. For further information about VXI*plug&play* drivers, refer to Chapter 5.

Using the Instrument Manager

This section provides an overview of how to use the `Instrument Manager` and the configuration dialog boxes to select and configure instruments in HP VEE. Some simple examples are given, and for many applications you can use the default values for most parameters. However, please refer to "Details of the Configuration Dialog Boxes", later in this chapter, for the technical details of the configuration fields in these dialog boxes.

Overview

To configure an instrument, select `I/O` ⟹ `Instrument Manager`:

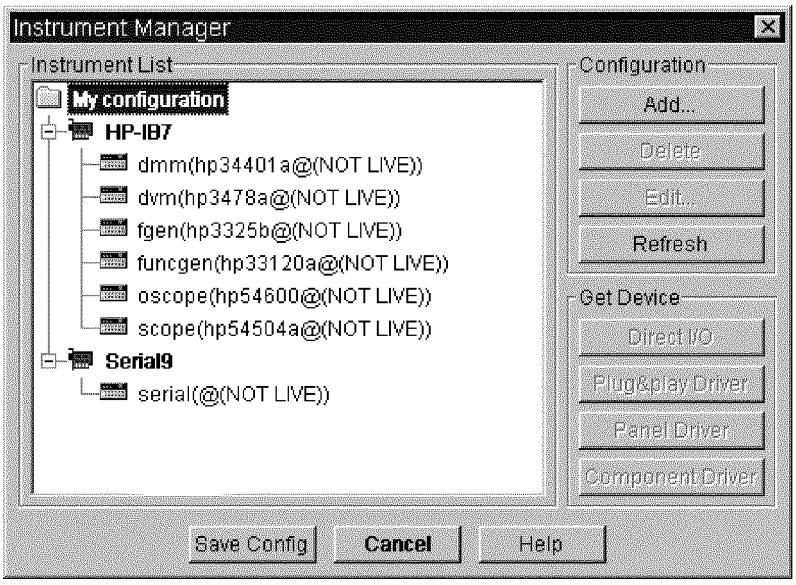

Figure 3-1. The Instrument Manager

The `Instrument Manager` displays three sections:

1. The `Instrument List` displays the instruments that are currently configured. This configuration is defined by the I/O configuration file (refer to "The I/O Configuration File" in Chapter 7 for further information). The default configuration, contained in the default I/O configuration file, is displayed in Figure 3-1.
2. The `Configuration` buttons (`Add ...` , `Delete`, `Edit ...` , and `Refresh`) allow you to modify the instrument configuration.
3. The `Get Device` buttons allow you to select `Direct I/O`, `To/From VXIplug&play`, `Panel Driver`, and `Component Driver` objects, and place them in your program.

Let's take a quick look at how to use the `Instrument Manager`. If you click on the `HP-IB7` selection, it becomes highlighted and the `Edit ...` button becomes active (it was "grayed out" before). This means you can now edit the configuration of the HP-IB interface at select code 7. Click on the `[-]` icon in front of `HP-IB7` to "collapse" the selections under it:

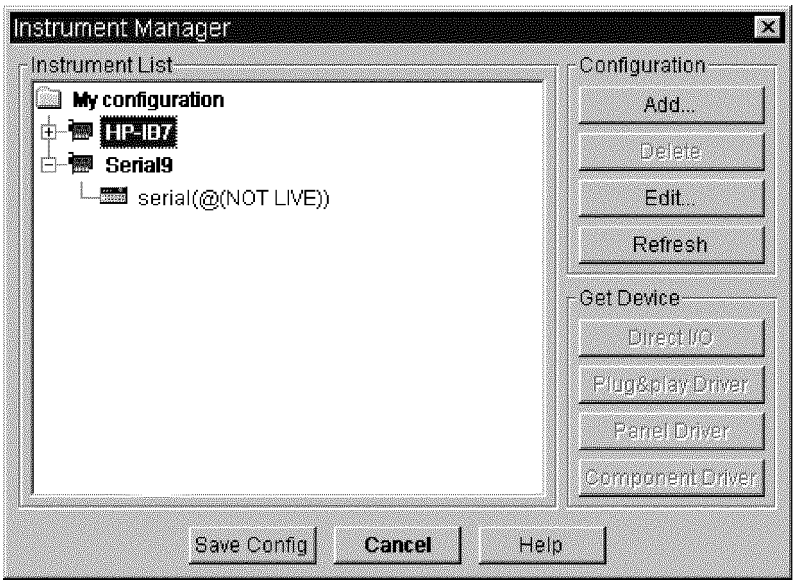

Figure 3-2. Collapsing the HP-IB7 Configuration

To "expand" the selections again, click on the `[+]` icon in front of `HP-IB7`.* Now click on the selection `dmm(hp34401@(NOT LIVE))`, or the "instrument" icon in front of it, to highlight it:

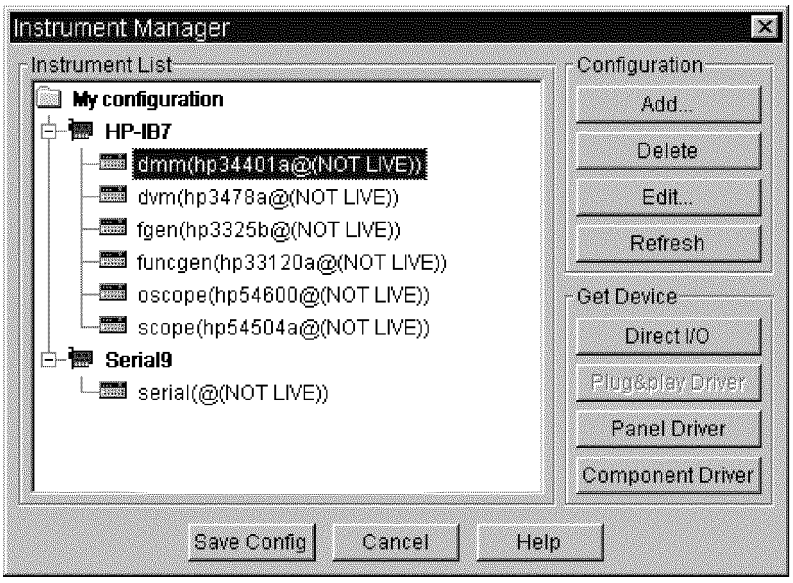

Figure 3-3. Expanding the HP-IB7 Configuration

Note that all of the buttons under `Configuration` are now active, including `Delete`. This means that you can delete, edit, or refresh (update) the configuration of the HP 34401A Digital Multimeter, or add a new instrument to the list. Also, note that the `Direct I/O`, `Panel Driver`, and `Component Driver` buttons under `Get Device` are now active. This means that you can select and place any of these types of objects for the HP 34401A. The `Plug&play Driver` button remains grayed out unless you have installed a VXI*plug&play* driver, if available, for the selected instrument.

---

* To expand the entire tree, select `My configuration` and press the (*) key.

Now press the `Refresh` button to update the instrument. You'll be asked whether you want to send the `*IDN?` (identification) message to the instrument. Click on `OK`. If an HP 34401A is actually connected to the HP-IB and turned on, it will respond and the instrument list will be as follows:

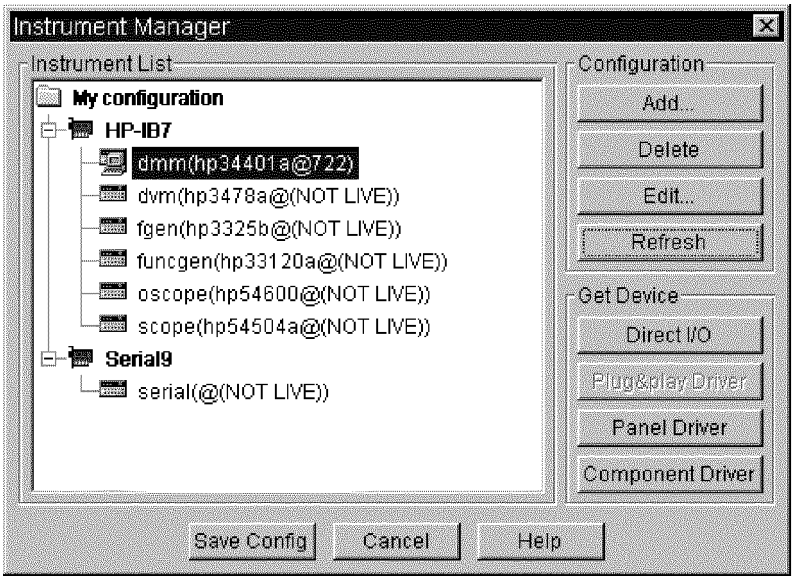

Figure 3-4. Updating the HP 34401A Configuration

Note that two changes have occurred:

1. The instrument identification has changed to `dmm(hp34401a@722)` indicating that the instrument is now in "live mode" and that the address of the HP 34401A is `722`.
2. The "instrument" icon in front of `dmm(hp34401a@722)` has changed to show that the instrument is connected to the computer.

(If you don't have an HP 34401A connected, or if it is not powered up, the configuration will remain in `NOT LIVE` mode and the icon won't change.)

You can use the `Refresh` button to automatically find and configure all of the instruments connected to your computer. Refer to "Using Refresh to Find and Configure Instruments" for further information.

Now let's add a `Panel Driver` object for the HP 34401A to the work area. With `dmm(hp34401a@722)` still highlighted in the instrument list, click on the `Panel Driver` button, and then place the object and click again. The following object appears:

Automatic Save Config

When you click on one of the `Get Device` buttons (`Direct I/O`, `Plug&play Driver`, `Panel Driver`, or `Component Driver`), the specified type of object is created. In addition, the instrument configuration is automatically saved, exactly as if you pressed the `Save Config` button.

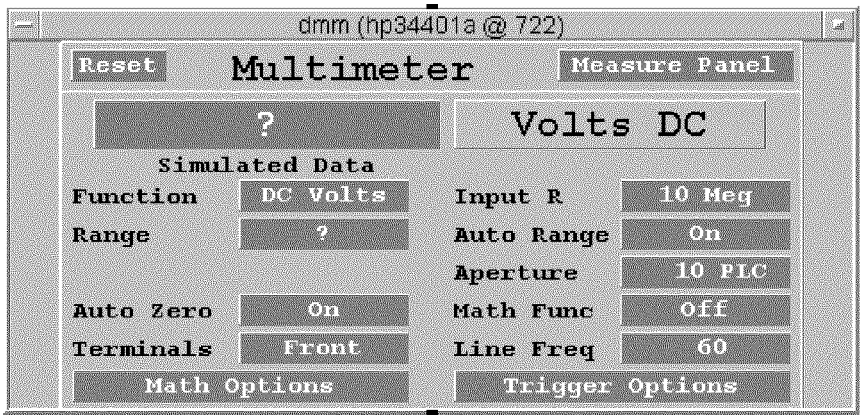

Figure 3-5. The Panel Driver Object

(If no HP 34401A is connected, the object will show that the instrument is in `NOT LIVE` mode.)

Configuring for a Panel Driver or Component Driver

In general, you'll use the same procedure to create a configuration for either a `Panel Driver` or a `Component Driver` object. In either case, you'll need to configure the appropriate instrument address, interface, and HP Instrument Driver ("ID") file using the `Instrument Manager`. (The HP Instrument Driver file is required for `Panel Driver` and `Component Driver` objects.)

Let's begin by adding a configuration for another HP 34401A Digital Multimeter.

Adding an Instrument Configuration

To add an instrument, open the `Instrument Manager` (`I/O` ⟹ `Instrument Manager`) and click on the `Add ...` button. The `Device Configuration` dialog box appears:

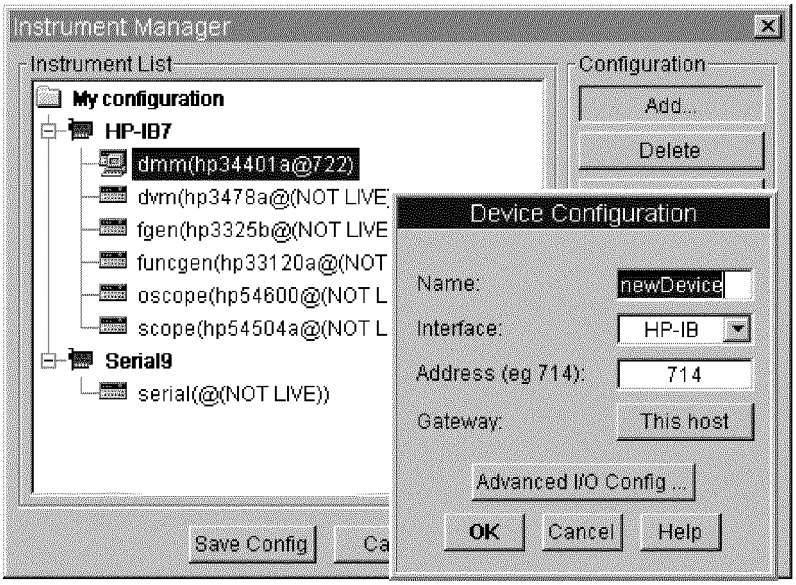

Figure 3-6. The Device Configuration Dialog Box

By default, the new configuration displays the name `NewDevice`. You can type in a new name, for example: `dmm2`. Leave the `Interface` field with `HP-IB` selected. (If you want to change the type of interface, click on the arrow to the right of `HP-IB` to display the drop-down list.) Now, click on the address field and change the address to `723`:

NOTE

To move from field to field in the dialog box, click on the desired field, or use the Tab key. If you press Enter or Return, the dialog box will exit.

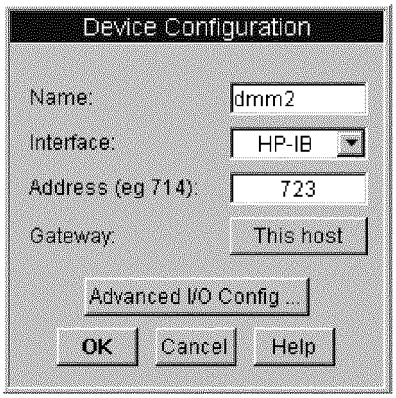

Figure 3-7. Changing the Name and Address Fields

Now click on the `Advanced I/O Config ...` button to display the `Advanced Device Configuration` dialog box:

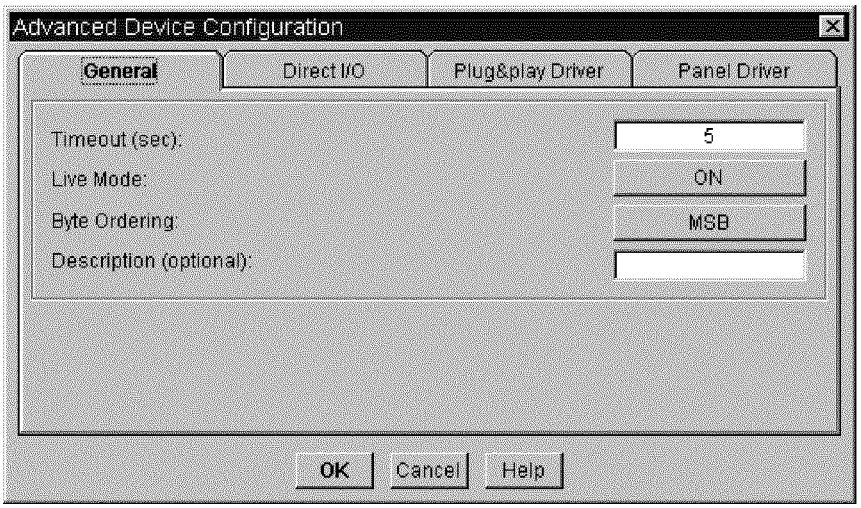

Figure 3-8. The Advanced Device Configuration Dialog Box

The `General` tab of this dialog box allows you to specify a timeout value, to turn live mode on or off, to select byte ordering, and to add a description. Click on the `Description` field and enter `hp34401a`.

NOTE

For further information about the individual fields in the `Device Configuration` and `Advanced Device Configuration` dialog boxes, refer to "Details of the Configuration Dialog Boxes" later in this chapter.

The tabs and fields displayed in the `Advanced Device Configuration` dialog box depend on the interface that you have selected.

Now select the `Panel Driver` tab.

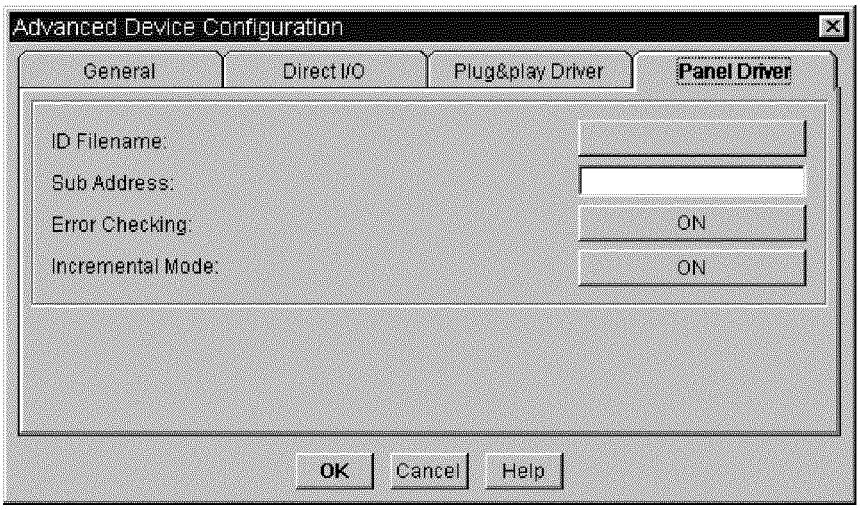

Figure 3-9. The Panel Driver Tab

Click on the `ID Filename` field. You are prompted to select an HP Instrument Driver file. (The Windows dialog is shown below. The UNIX® dialog is different, but also allows you to select a file.)

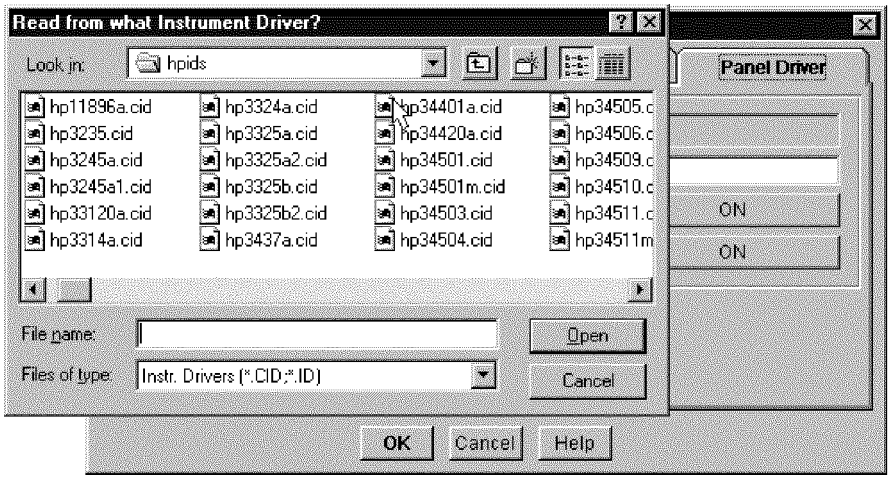

Figure 3-10. Selecting an Instrument Driver File

Double-click on `hp34401a.cid` to select that file.

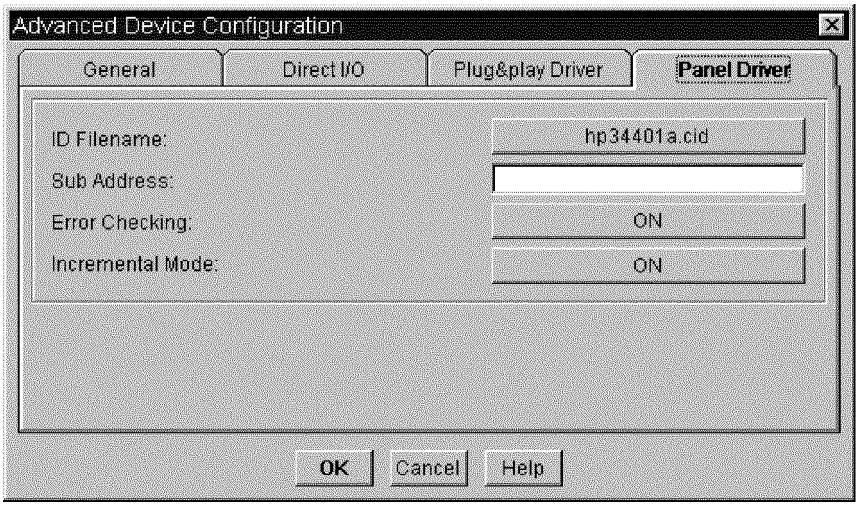

Figure 3-11. The Selected ID Filename

Now click on `OK` on each dialog box to return to the `Instrument Manager`.

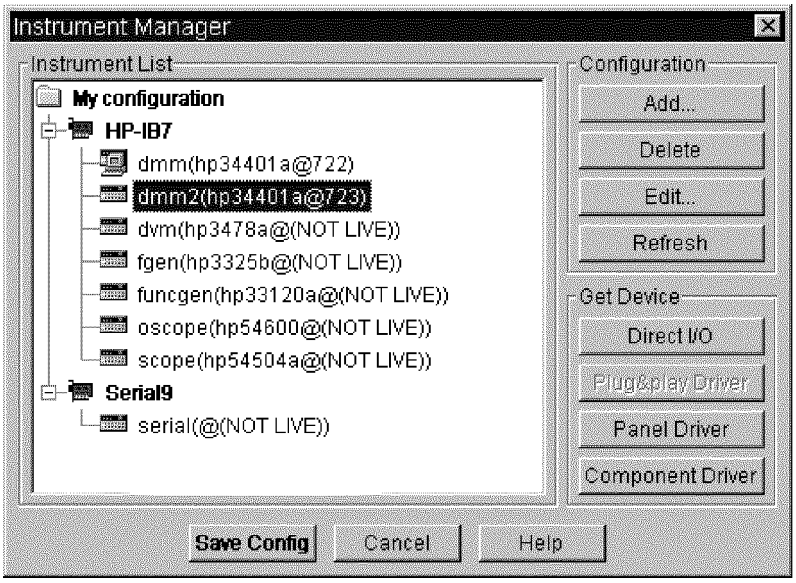

Figure 3-12. The New Configuration

At this point you can save the new configuration by clicking on the `Save Config` button.

Adding a Panel Driver or Component Driver to the Work Area

Now that you have saved your new configuration, you can add either a `Panel Driver` object or a `Component Driver` object for `dmm2`. Select `I/O` ⟹ `Instrument Manager` to redisplay the `Instrument Manager`, as shown in Figure 3-12. Click on `dmm2(hp34401a@723)` if it is not already highlighted, and then click on the `Component Driver` button. Move the outline to the desired position in the work area, and click the mouse button to place the `Component Driver` object. The object appears as an icon:

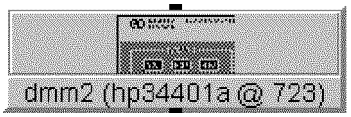

Figure 3-13. The Component Driver Object

In the same manner, if you had clicked on the `Panel Driver` button, a `Panel Driver` object (see Figure 3-5) would have appeared.

Editing an Instrument Configuration

You can edit an existing instrument configuration, also using the `Device Configuration` and `Advanced Device Configuration` dialog boxes. To edit the configuration for the HP 34401A Digital Multimeter, select `dmm(hp34401a@722)` in the `Instrument List`, and then click on the `Edit ...` button. The `Device Configuration` dialog appears:

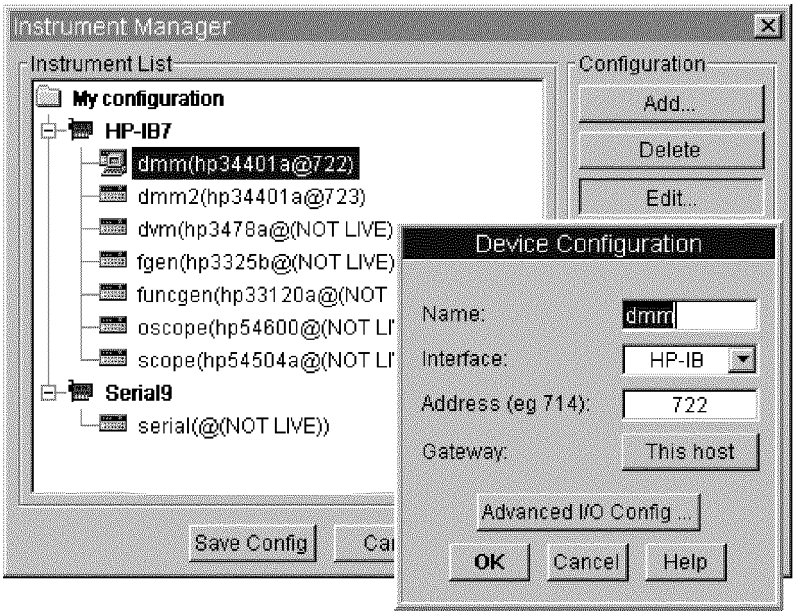

Figure 3-14. Editing the dmm Configuration

To change the configuration, modify the fields in the configuration dialog boxes as described previously in "Adding an Instrument Configuration".

Editing an Interface Configuration

You can also edit an entire interface configuration, affecting multiple instruments. To do this, select the *interface* in the instrument list, and then click on the `Edit ...` button. For example, select `HP-IB7` and click on the `Edit ...` button:

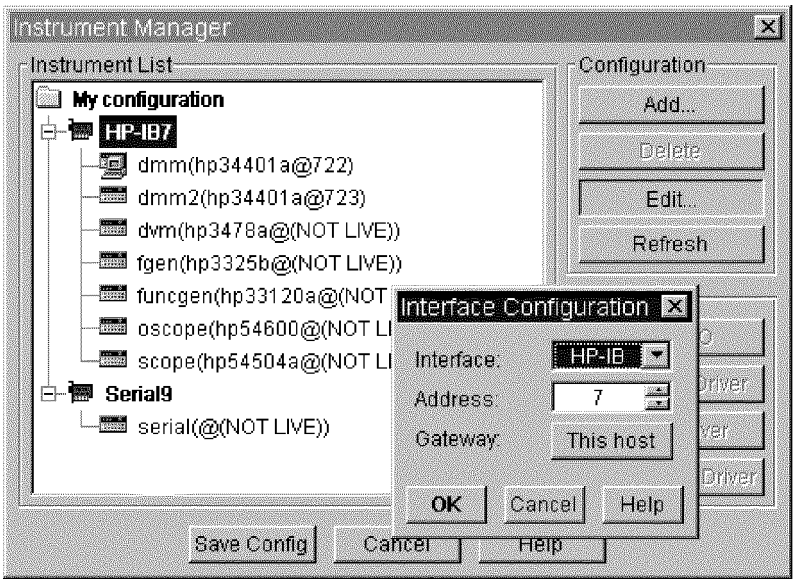

Figure 3-15. Editing the HP-IB7 Configuration

Press `Cancel` to make no changes, retaining the `HP-IB7` configuration for use in examples.

NOTE

From the `Interface Configuration` dialog box, you can change the interface type from `HP-IB` to `VXI`, the address from `7` to some other unused select code, and you can configure a LAN gateway. Any changes will affect all of the instruments (`dmm`, `dmm2`, and so forth) currently under `HP-IB7`. For further information, refer to "Details of the Configuration Dialog Boxes".

Configuring for a Direct I/O Object

Now let's look at an example of configuring for a `Direct I/O` object. In our example, we'll configure the serial interface at select code 9 (COM1) for direct I/O.

Select `I/O` ⟹ `Instrument Manager`, highlight `serial(@(NOT LIVE))`, and click on `Edit ...`

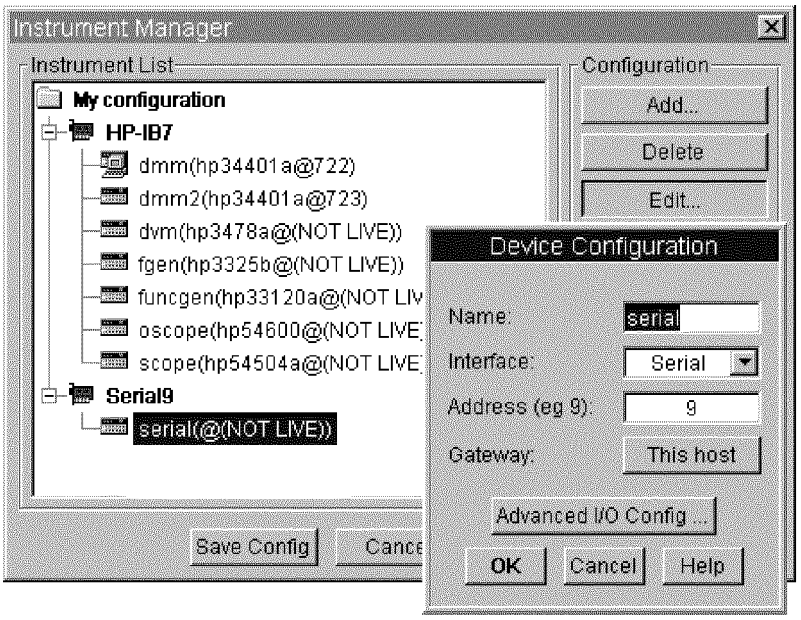

Figure 3-16. Configuring a Serial Device

The `Device Configuration` dialog box allows you to select the name and address of the interface, as usual. You can use the default parameters.

Now click on `Advanced I/O Config ...` to display the `Advanced Device Configuration` dialog box. There are two tabs of interest. The `Serial` tab allows you to specify the serial parameters such as baud rate. Refer to "Details of the Configuration Dialog Boxes" for further information about the individual parameters and fields. You can use the defaults for most applications.

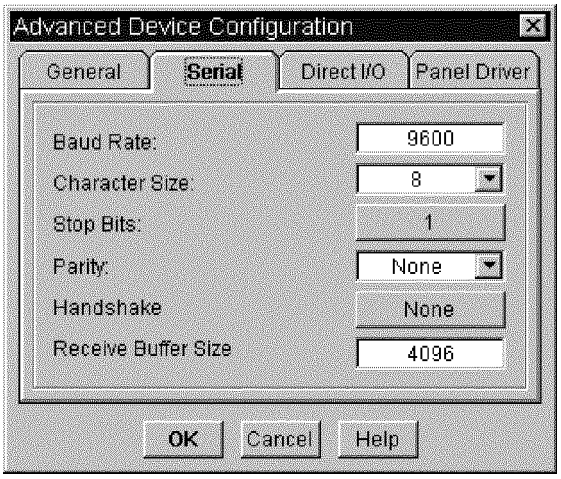

Figure 3-17. The Serial Tab

The `Direct I/O` tab allows you to specify a number of parameters for direct I/O, including the EOL sequence. You can use the defaults for most applications.

NOTE

The selection of fields displayed by the `Direct I/O` tab depends on the interface that you have selected. In addition, for VXI only there are two additional tabs—`A16 Space` and `A24/A32 Space`. These tabs allow you to configure a VXI device's registers for `WRITE` or `READ` transactions in a `Direct I/O` object. Refer to "Details of the Configuration Dialog Boxes" for further information about the parameters and fields displayed by each tab.

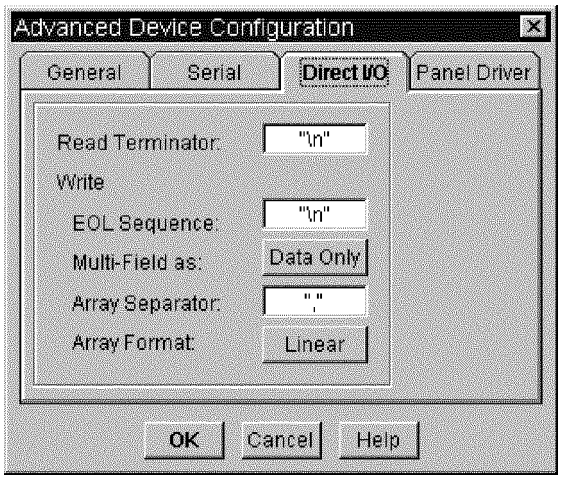

Figure 3-18. The Direct I/O Tab

Click on `OK` (or `Cancel` to make no changes) on each dialog box to return to the `Instrument Manager`. To add a `Direct I/O` object to the work area, click on the `Direct I/O` button, place the object, and click again.

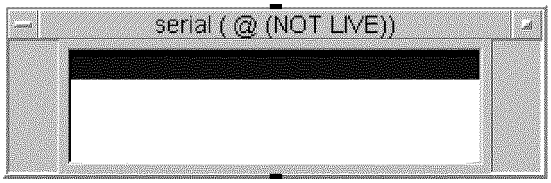

Figure 3-19. The Direct I/O Object

NOTE

`Direct I/O` objects use transaction-based I/O to communicate with instruments, without using an instrument driver. Refer to Chapter 4 for further information.

Configuring for a VXIplug&play Driver

The procedure to configure for a `To/From VXIplug&play` object is very similar to the procedures for `Panel Driver`, `Component Driver`, and `Direct I/O` objects. However, you must first install the appropriate VXI*plug&play* driver files as described in "Installing the VXI*plug&play* Driver Software" in Chapter 2.

For example, let's add a VXI*plug&play* configuration for the HP E1410A 6.5-Digit VXI Multimeter. Select `I/O` ⟹ `Instrument Manager`, and click on `Add ...`. The `Device Configuration` dialog box appears. Change the name to `vxiDevice`, and select `VXI` for the interface type, as shown below:

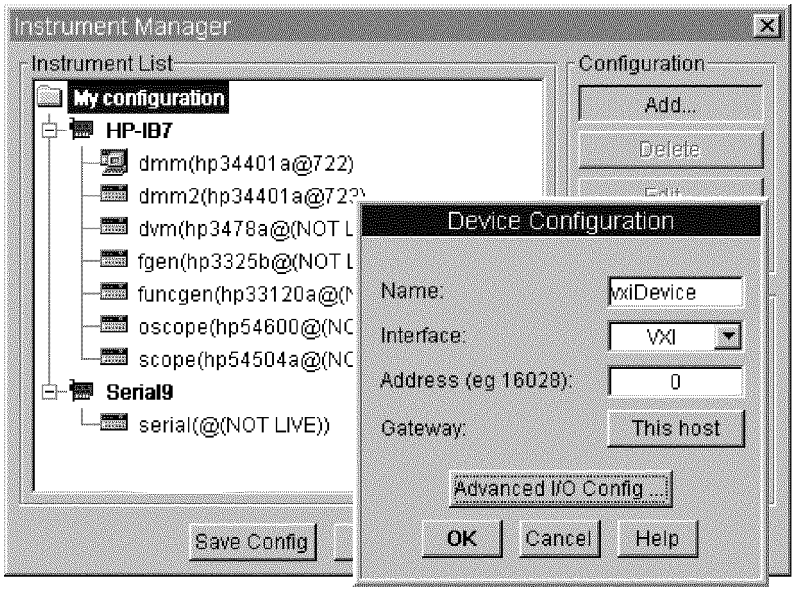

Figure 3-20. Adding a VXI Device

The `Address` field is not used for VXI*plug&play* drivers. Click on `Advanced I/O Config ...` to display the `Advanced Device Configuration` dialog box, and then select the `Plug&play Driver` tab.

Now select the driver named `HPE1410` from the `Plug&play Driver Name` drop-down list, as shown below. (You won't be able to select the VXI*plug&play* driver unless you have previously installed it as described in "Installing the VXI*plug&play* Driver Software" in Chapter 2.)

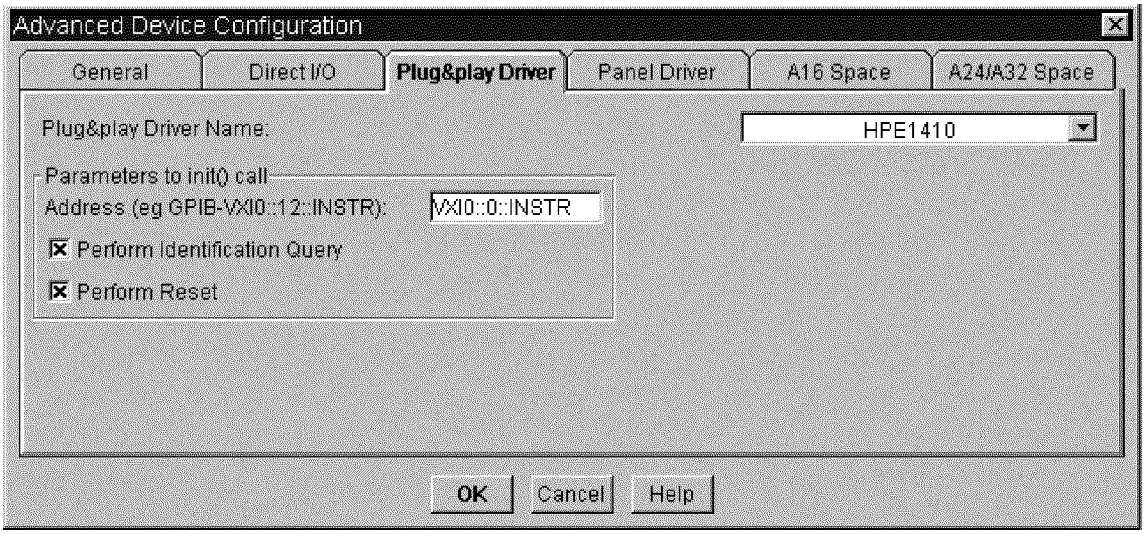

Figure 3-21. The Plug&play Driver Tab

By default, the `Address` field displays `VXI0::0::INSTR`, which assumes a VXI logical address of `0` for the instrument. Generally, you'll need to supply the correct logical address. For example, if the logical address of the HP E1410A is `24`, change the `Address` field to `VXI0::24::INSTR`. For further information about the fields in the `Plug&play Driver` tab, refer to "Details of the Configuration Dialog Boxes".

NOTE

Only the `Plug&play Driver` tab applies to configuring VXI*plug&play* drivers. The `General`, `Direct I/O`, `Panel Driver`, `A16 Space`, and `A24/A32 Space` tabs have no effect on a VXI*plug&play* configuration.

Once you have configured the instrument, click on `OK` on each dialog box to return to the `Instrument Manager`, which will show the added instrument:

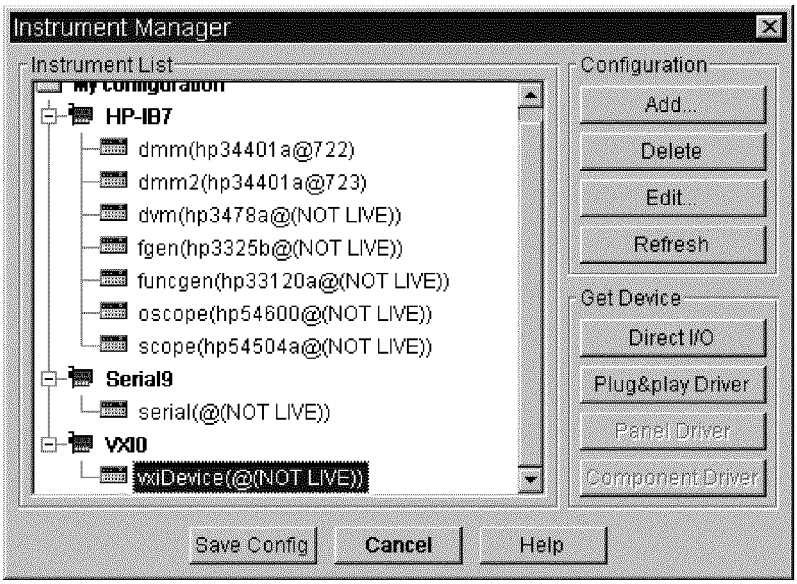

Figure 3-22. The VXI Configuration

Click on the `Plug&play Driver` button to add a `To/From VXIplug&play` object:

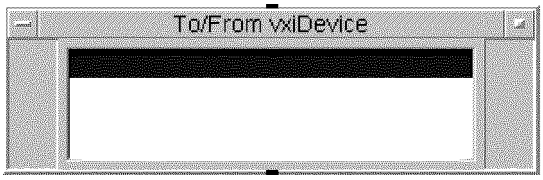

Figure 3-23. The To/From VXIplug&play Object

Refer to "Using the To/From VXIplug&play Object" in Chapter 5 for information about using the `To/From VXIplug&play` object.

Using Refresh to Find and Configure Instruments

You can use the `Refresh` button in the `Instrument Manager` to automatically find and configure the instruments that are connected to your computer:

- If you click on `Refresh` with `My Configuration` highlighted in the `Instrument List`, all of the existing instrument configurations will be updated (live mode will be turned on or off depending on whether the instrument is present and powered up), and any unconfigured instruments will be added to the list.
- If you click on `Refresh` with an interface (for example, `HP-IB7`) highlighted, all instrument configurations under that interface will be updated, and any unconfigured instruments will be added to the list.
- If you click on `Refresh` with a particular HP-IB or VXI instrument highlighted, the `*IDN?` message will be sent to the physical instrument, and the instrument will be updated in the `Instrument List` to include its idenfication.

Let's look at an example to see how this works. Figure 3-24 shows the `Instrument Manager` with an "empty" `Instrument List`. We've deleted all of the instruments from the list, but don't do this yourself unless you are sure you don't need them.

NOTE

Don't delete the instruments from your `Instrument List` unless you first "back up" your I/O configuration file by renaming it, as described in "Running Example Programs" in Chapter 7.

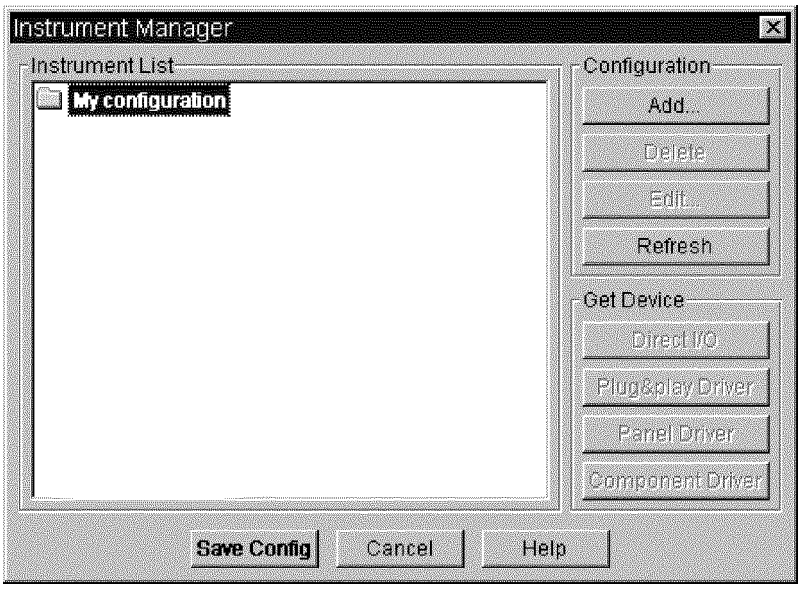

Figure 3-24. An "Empty" Instrument List

If we now click on the `Refresh` button, all of the interfaces present, and all of the HP-IB and VXI instruments connected and powered up, will be added to the `Instrument List`.

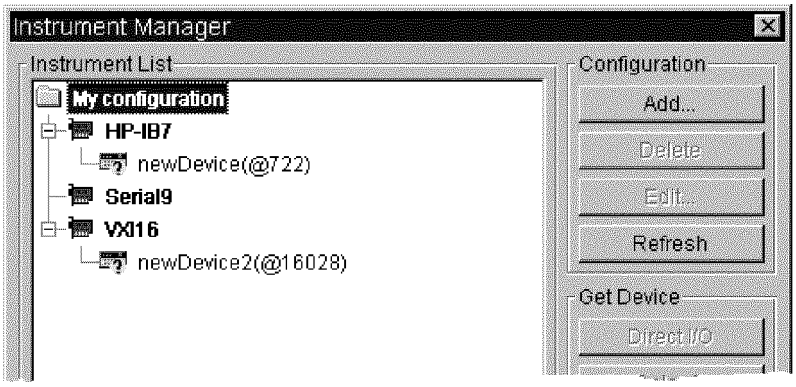

Figure 3-25. Finding the Interfaces and Instruments

In this case, the interfaces are `HP-IB7`, `Serial9`, and `VXI16`. Also, an HP-IB device identified as `newDevice` is present at address `722`, and a VXI device identified as `newDevice2` is present at address `16028`. (Note that these instruments will only be found if they are powered up.)

To indentify and configure the HP-IB device, select `newDevice(@722)` and click on `Refresh` again. You'll be asked whether you want to send the `*IDN?` message to the device.

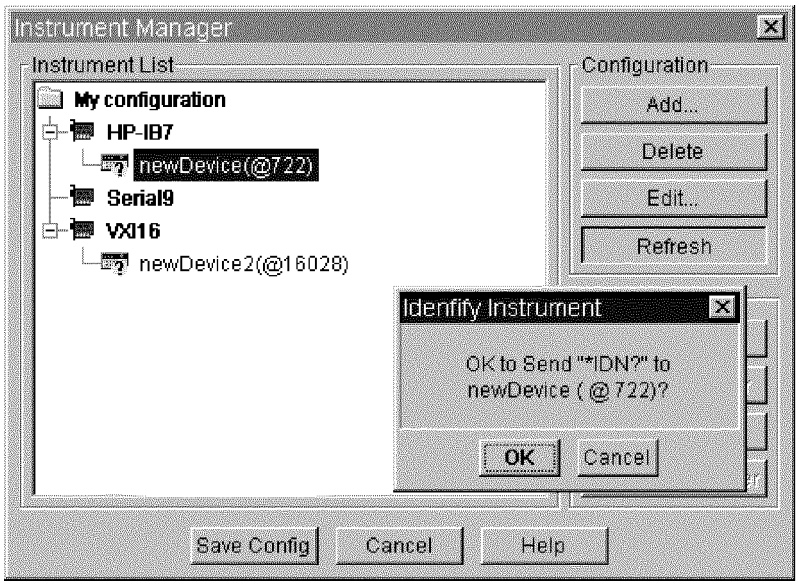

Figure 3-26. Sending the *IDN? Message to an Instrument

Click on `OK` to send the message. The device, now idenfified as an HP 34401A Multimeter, is now configured as `newDevice(hp34401@722)`.

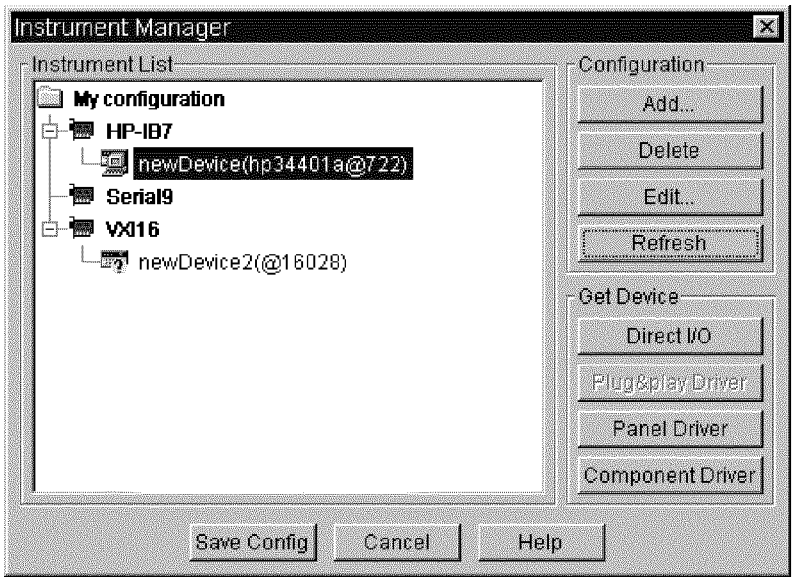

Figure 3-27. The Configured Instrument

You can now repeat the process to configure the VXI device:

1. Select `newDevice2@(16028)` in the `Instrument List`.
2. Click on `Refresh`.
3. Click on `OK` to send the `*IDN?` message to the instrument.
4. The instrument will be identified and configured, provided it is properly connected and powered up.

NOTE

`Refresh` will automatically find and configure VXI and HP-IB instruments with VXI*plug&play* driver configurations, provided the corresponding VXI*plug&play* driver files have been installed.

Details of the Configuration Dialog Boxes

The remainder of this chapter provides a detailed description of the `Device Configuration` dialog box, each tab of the `Advanced Device Configuration` dialog box, and the `Interface Configuration` dialog box. In each case, the individual fields are described in detail. For an overview of how to use the `Instrument Manager` and these dialog boxes, refer to "Using the Instrument Manager" earlier in this chapter.

Device Configuration

The `Device Configuration` dialog box appears when you select an instrument and click on either the `Add ...` button or the `Edit ...` button in the `Instrument Manager`. Here is an example of this dialog box:

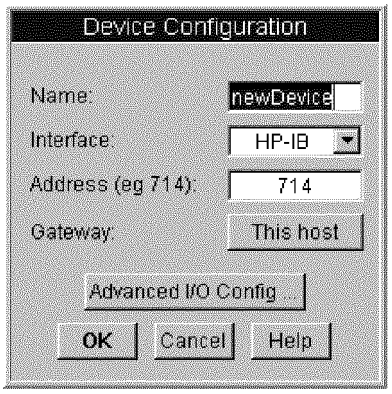

Figure 3-28. The Device Configuration Dialog Box

The following sections describe the individual fields.

Name

The **Name** field uniquely identifies a particular instrument configuration. The instrument **Name** is a symbolic link between each instance of an instrument control object and all the configuration information corresponding to that **Name**. Usually, this field is used to give a descriptive name to the instrument, such as **Oscilloscope** or **Power Supply**.

Names must be unique; you cannot configure two instruments with a **Name** of **Scope**. While it is possible to create two different **Name**s that refer to the same physical instrument, it can cause problems if you use both **Name**s with **Panel Driver**s or VXI*plug&play* drivers in the same program.

Do not confuse the **Name** of an instrument with the text that appears as the title in an instrument control object. The default title of an instrument control object is the name, but you can change the title and it has no effect on the **Name**. If you need to determine the **Name** of a particular instance of an instrument control object, select **Show Config** in the object menu.

> **NOTE**
>
> It is very important that you use **Name**s correctly. This section discusses only the more common situations. For more details about how HP VEE uses **Name**s please refer to "The Importance of Names" in Chapter 6.

Interface

The **Interface** field specifies the type of hardware interface used to communicate with the instrument: **HP-IB**, **VXI**, **GPIO**, or **Serial**.

Address

The **Address** field specifies the address of the instrument. For instruments using GPIO or serial interfaces, the address is the same as the interface select code. An interface select code is a number used by the computer to identify a particular interface.

For instruments using HP-IB interfaces, the address is of the form *xxyyzz*, where:

- *xx* is the one- or two-digit interface select code. The factory default select code for most HP-IB interfaces is 7.

- *yy* is the two-digit bus address of the instrument. Use a leading zero for bus addresses less than `10`; for example, use `09` not `9`.
- *zz* is the secondary address of the instrument. Secondary addresses are typically used by cardcage-type instruments that use multiple plug-in modules. Secondary addresses are used to access devices through a command module in a C size VXI mainframe, and to address devices in a B size VXI mainframe.

NOTE

The secondary address is the secondary address as defined in IEEE 488.1; it is part of the interface specification of the instrument hardware. The instrument *hardware* design determines whether or not a secondary address is required; secondary addresses are *not* related to *driver* configuration.

Do not confuse secondary addresses with the `Sub Address` field used in the `Advanced Device Configuration` dialog box. Subaddresses are a *driver-related* feature and are used *very rarely*.

For instruments using VXI interfaces (connected to embedded controllers or controllers with direct access to the VXI backplane), the address is of the form *xxyyy*, where:

- *xx* is the one- or two-digit select code of the VXI backplane interface of an embedded or external controller.
- *yyy* is the logical address of the VXI device. Use leading zeros for logical addresses less than `100`. (For example, use `008` not `8`.)

NOTE

Setting the `Address` field to `0` has special meaning. Setting the `Address` field to `0` (for any interface) means that there is no physical instrument matching this device description connected to the computer. An address of `0` automatically sets `Live Mode` to `OFF`.

HP-IB and GPIB Address Examples. Suppose you want to control an HP-IB instrument at bus address 9 using an HP-IB interface card that has been configured with select code 7. (Refer to Appendix A for information about the recommended select codes.) The proper `Address` field setting for the instrument is `709`.

If you want to control an instrument at bus address 12 using a GPIB interface card that has been configured with select code 14, the proper `Address` field setting is `1412`.

VXI Address Examples. Suppose you want to control a VXI instrument, which has logical address 28, using an HP V/743 Embedded VXI Controller, which is configured with select code 16. (Refer to Appendix A for information about the recommended select codes.) The proper `Address` field setting is `16028`. (Logical addresses for VXI instruments are in the range 1–255, inclusive.)

Suppose you want to address a VXI instrument, which has logical address 24, using an HP E1406 HP-IB Command Module, which has bus address 9, by means of the HP-IB interface at select code 7. For the HP E1406 HP-IB Command Module, use a secondary address for the VXI instrument equal to its logical address divided by 8. That is, for logical address 24, the secondary address is 3. Thus, the complete address is `70903`.

Serial Address Example. Suppose you want to control an instrument using the COM1 serial port, and that COM1 has been configured with select code 9. (Refer to Appendix A for information about the recommended select codes.) The proper `Address` field setting for the instrument is `9`.

GPIO Address Example. Suppose you want to control a custom-built instrument using an HP E2075 GPIO Interface that has been configured with select code 13. (Refer to Appendix A for information about the recommended select codes.) The proper `Address` field setting for the instrument is `13`.

Gateway

Use the `Gateway` field set to the name of the LAN gateway used during a remote process. Refer to "LAN Gateways" in Chapter 7 for further information.

Advanced I/O Config ... Button

Click on the `Advanced I/O Config ...` button to go to the `Advanced Device Configuration` dialog box.

Advanced Device Configuration: General

Here is an example of the `General` tab of the `Advanced Device Configuration` dialog box:

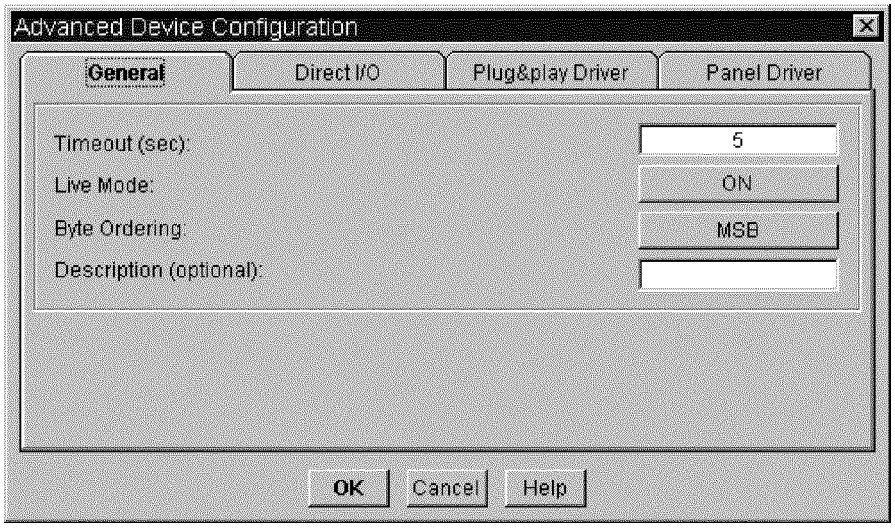

Figure 3-29. The General Tab

The following sections describe the individual fields.

NOTE

The parameters specified in the `General` tab apply to `Direct I/O`, `Panel Driver`, and `Component Driver` objects, but not to `To/From VXIplug&play` objects.

Timeout

The `Timeout` field specifies how many seconds HP VEE will wait for an instrument to respond to a request for communication before generating an error. The default value of five seconds works well for most applications. In general, you should *not* set this field to `0`. If you do, HP VEE will *never* detect a timeout. Certain `Direct I/O` transactions for register or memory access of VXI devices do not support a timeout.

Live Mode

The `Live Mode` field determines whether or not HP VEE will attempt to communicate with an instrument at the specified address. To actually communicate with a physical instrument connected to your computer, you *must* set `Live Mode` to `ON`.

Note that if `Live Mode` is `OFF` for a particular instrument, you can run programs containing `Panel Drivers`, `Component Drivers`, or `Direct I/O` objects that would otherwise read and write to that instrument. However, no instrument communication actually takes place. This behavior can be useful if you want to develop or debug portions of a program while instruments are not available.

Byte Ordering

Use this field to specify the order the device uses for reading and writing binary data. HP VEE uses the value in this field to determine if byte swapping is necessary. Click on this field to choose between `MSB` (send Most-Significant Byte first) and `LSB` (send Least-Significant Byte first). All IEEE 488.2-compliant devices *must* default to MSB order. Please refer to your device manual for more specific information.

Description

The `Description` field is typically used to record the manufacturer's model number. For example, the `Description` for the HP 54504A oscilloscope could be `hp54504a`. This field is provided for your convenience—HP VEE does not use it.

Advanced Device Configuration: Direct I/O

Here is an example of the `Direct I/O` tab of the `Advanced Device Configuration` dialog box (shown for the HP-IB interface):

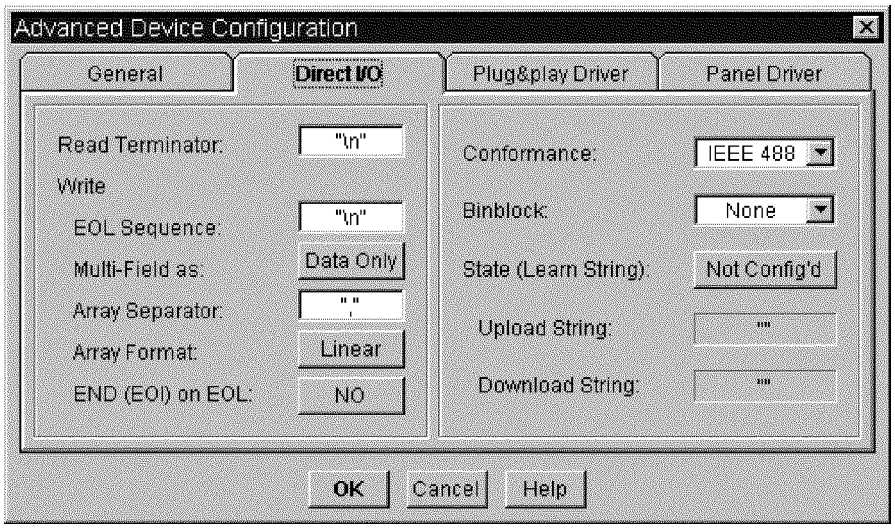

Figure 3-30. The Direct I/O Tab

The following sections describe the individual fields.

NOTE

When addressing VXI devices directly on the VXI backplane, you can use SCPI messages to control register-based devices if I-SCPI drivers exist for them. HP VEE will inform you if required I-SCPI drivers are not available. If I-SCPI drivers are not available, you must then control register-based devices by direct read/write access to device registers or device memory. Refer to "Advanced Device Configuration: A16 Space (VXI Only)" or "Advanced Device Configuration: A24/A32 Space (VXI Only)" for details.

Read Terminator

The `Read Terminator` field specifies the character that terminates `READ` transactions. The entry in this field must be a a single character surrounded by double quotes. "Double quote" means ASCII 34 decimal. HP VEE recognizes any ASCII character as a `Read Terminator` as well as the escape characters shown in Table 3-1.

The character you should specify is determined by the design of your instrument. Most HP-IB instruments send newline after sending data to the computer. Consult your instrument programming manual for details.

Table 3-1. Escape Characters

Escape Character	ASCII Code (decimal)	Meaning
`\n`	10	Newline
`\t`	9	Horizontal Tab
`\v`	11	Vertical Tab
`\b`	8	Backspace
`\r`	13	Carriage Return
`\f`	12	Form Feed
`\"`	34	Double Quote
`\'`	39	Single Quote
`\\`	92	Backslash
`\`*ddd*		The ASCII character corresponding to the three-digit octal value *ddd*.

EOL Sequence

The `EOL Sequence` field specifies the characters that are sent at the end of `WRITE` transactions that use `EOL ON`. The entry in this field must be zero or more characters surrounded by double quotes. "Double quote" means ASCII 34 decimal. HP VEE recognizes any ASCII characters within `EOL Sequence` including the escape characters shown previously in Table 3-1.

Multi-field As

The `Multi-field As` field specifies the formatting style for multi-field data types for `WRITE TEXT` transactions. The multi-field data types in HP VEE are Coord, Complex, PComplex, and Spectrum. Other data types and other formats are unaffected by this setting.

Specifying a multi-field format of `( ... ) Syntax` surrounds each multi-field item with parentheses. Specifying `Data Only` omits the parentheses, but retains the separating comma. For example, the complex number `2+2`*j* could be written as `(2,2)` using `( ... ) Syntax` or as `2,2` using `Data Only` syntax.

Array Separator

The `Array Separator` field specifies the character string used to separate elements of an array written by `WRITE TEXT` transactions. The entry in this field must be a a single character surrounded by double quotes. "Double quote" means ASCII 34 decimal. HP VEE recognizes any ASCII character as an `Array Separator` as well as the escape characters shown previously in Table 3-1.

`WRITE TEXT STR` transactions in `Direct I/O` objects that write arrays are a special case. In this case, the value in the `Array Separator` field is ignored and the linefeed character (ASCII 10 decimal) is used to separate the elements of an array. This behavior is consistent with the needs of most instruments.

Note that HP VEE allows arrays of multi-field data types; for example you can create an array of Complex data. In such a case, if `Multi-Field Format` is set to `( ... ) Syntax`, the array will be written as:

`(1,1)`*array_sep*`(2,2)`*array_sep* `...`

where *array_sep* is the character specified in the `Array Separator` field.

Array Format

The `Array Format` determines the manner in which multidimensional arrays are written. For example, mathematicians write a matrix like this:

```
1  2  3
4  5  6
7  8  9
```

HP VEE writes the same matrix in one of two ways, depending on the setting of `Array Format`. In the two examples that follow, `EOL Sequence` is set to `"\n"` (newline) and `Array Separator` is set to `" "` (space).

```
1 2 3    Block Array Format
4 5 6
7 8 9
```

```
1 2 3 4 5 6 7 8 9   Linear Array Format
```

Either array format separates each element of the array with the `Array Separator` character. `Block Array Format` takes the additional step of separating each row in the array using the `EOL Sequence` character.

In the more general case (arrays greater than two dimensions), `Block Array Format` outputs an `EOL Sequence` character each time a subscript other than the right-most subscript changes. For example, if you write the three-dimensional array `A[x,y,z]` using `Block` array format with this transaction:

```
WRITE TEXT A
```

an `EOL Sequence` will be output each time `x` or `y` changes value. If the size of each dimension in `A` is two, the elements will be written in this order:

```
A[0,0,0]  A[0,0,1]<EOL Sequence>
A[0,1,0]  A[0,1,1]<EOL Sequence>
<EOL Sequence>
A[1,0,0]  A[1,0,1]<EOL Sequence>
A[1,1,0]  A[1,1,1]<EOL Sequence>
```

Notice that after `A[0,1,1]]` is written, `x` and `y` change simultaneously and consequently two `<EOL Sequence>`s are written.

Writing Arrays with Direct I/O. `WRITE TEXT STR` transactions that write arrays to direct I/O paths ignore the `Array Separator` setting for the `Direct I/O` object. These transactions always use linefeed (ASCII decimal 10) to separate each element of an array as it is written. This behavior is consistent with the needs of most instruments. *(This special behavior for arrays does not apply to any other type of transaction.)*

END On EOL (HP-IB Only)

`END on EOL` controls the behavior of EOI (End Or Identify). If `END on EOL` is `YES`, the EOI line is asserted on the bus at the time the last data byte is written under one of the following circumstances:

1. A `WRITE` transaction with `EOL ON` executes.
2. A `WRITE` transaction executes as the last transaction listed in the `Direct I/O` object.
3. One or more `WRITE` transactions execute without asserting EOI and are followed by a non-`WRITE` transaction, such as `READ`.

Many instruments accept *either* EOI or a newline as valid message terminators. Some block transfers may require EOI. Consult your instrument's programming manual for details.

Conformance

`Conformance` specifies whether an instrument conforms to the IEEE 488.1 or IEEE 488.2 standard. Refer to your instrument programming manual to determine the standard to which your instrument conforms, and then set the `Conformance` field accordingly.

Each of these standards defines communication protocols for the HP-IB interface. However, IEEE 488.2 specifies rules for block headers and learn strings that are left undefined in IEEE 488.1. All message-based VXI instruments are IEEE 488.2 compliant, as well as register-based VXI instruments supported by I-SCPI drivers.

If you set `Conformance` to `IEEE 488` (which denotes IEEE 488.1), you may optionally specify additional settings to handle block headers and learn strings, as described on the following page.

Binblock

The `Binblock` field specifies the block data format used for `WRITE BINBLOCK` transactions. `Binblock` may specify IEEE 728 `#A`, `#T`, or `#I` block headers. If `Binblock` is `None`, `WRITE BINBLOCK` writes an IEEE 488.2 Definite Length Arbitrary Block Response Data block.

IEEE 728 block headers are of the following forms:

```
#A<Byte_Count><Data>
#T<Byte_Count><Data>
#I<Data><END>
```

where:

`<Byte_Count>` is a 16-bit unsigned integer that specifies the number of bytes that follow in `<Data>`.

`<Data>` is a stream of arbitrary bytes.

`<END>` indicates that EOI is asserted with the last data byte transmitted.

State

The `State` field indicates whether or not the instrument has been configured for uploading and downloading learn strings. If the `State` entry is `Not Config'd` and you want to configure the instrument for use with learn strings, click on the `State` field and the `Upload String` and `Download` fields will appear. If the `State` entry is `Not Config'd`, the `Upload String` and `Download String` fields are set to the null string.

Upload String

The `Upload String` field specifies the command that is sent to the instrument when you select `Upload State` from the `Direct I/O` object menu. Specify the command that causes the instrument to output its learn string; consult your instrument programming manual for details. Note that you must surround the command with double quotes.

Download String

The `Download String` field specifies the string that is sent to the instrument immediately before the learn string as the result of a `WRITE STATE` transaction in a `Direct I/O` object. This field is provided to support instruments that require a command prefix when downloading a learn string; consult your instrument programming manual for details.

Advanced Device Configuration: Plug&play Driver

Here is an example of the `Plug&play Driver` tab of the `Advanced Device Configuration` dialog box (shown for the HP-IB interface):

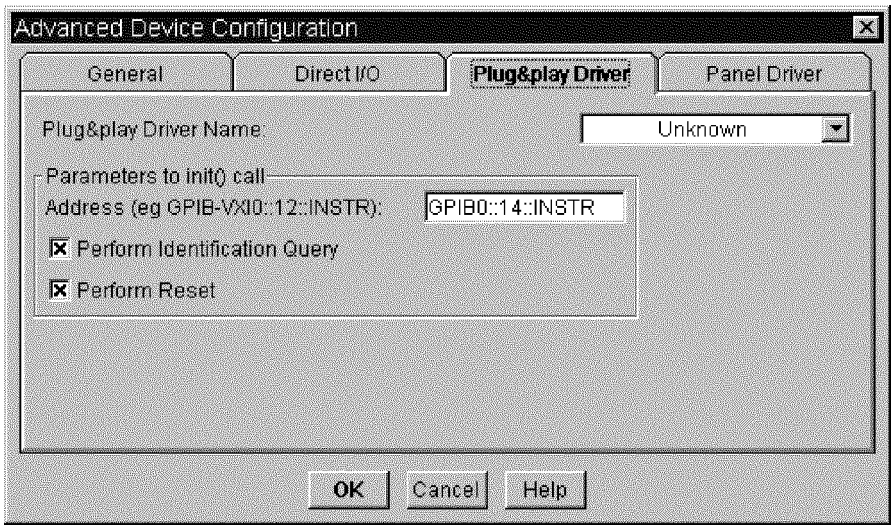

Figure 3-31. The Plug&play Driver Tab

The `Plug&play Driver` tab is the *only* tab of the `Advanced Device Configuration` dialog box that applies to VXI*plug&play* driver configurations.

The following sections describe the individual fields.

Plug&play Driver Name

This field specifies the name of the VXI*plug&play* driver. You must select a driver name—this parameter is required. The drop-down list displays all of the VXI*plug&play* drivers that are installed. If there are no entries in the list, either you do not have any VXI*plug&play* drivers installed, or your registry entry or environment variable may not be set correctly. (Refer to "Introduction to VXI*plug"&play*" in Chapter 2 for further information.)

Parameters to init() call

Address. Enter the address that identifies the instrument. The address format depends on the interface to which the instrument is connected:

- VXI address string (*embedded VXI, VXLink, or MXIbus controller*).

  For a VXI instrument with an embedded, VXLink, or MXIbus controller, the address string takes the form `VXI[`*board*`]::`*VXI logical address*`[::INSTR]`. An example is `VXI::24::INSTR` for an instrument at logical address 24.

  The *board* number is optional for the first board (`VXI::24::INSTR` is equivalent to `VXI0::24::INSTR`). However, the *board* number is required for subsequent boards (`VXI1`, `VXI2`, and so forth).

- GPIB-VXI address string (*command module*).

  For a VXI instrument that is being controlled from a GPIB card connected to a command module, the address string takes the form `GPIB-VXI[`*board*`]::`*VXI logical address* `[::INSTR]` . An example is `GPIB-VXI::24::INSTR` (or `GPIB-VXI0::24::INSTR`) for an instrument at VXI logical address 24.

- GPIB address string (*GPIB instruments*).

  For a non-VXI instrument being controlled from a GPIB card, the address string takes the form `GPIB[`*board*`]::`*GPIB primary address*`::[`*GPIB secondary address* `][::INSTR]`. An example is `GPIB::23::INSTR` (or `GPIB0::23::INSTR`) for a GPIB instrument at primary address 23. (The optional secondary address is rarely used.)

Perform Identification Query. Select this check box if you want the driver to query the instrument for its identification the first time a function panel for this driver is executed. You generally want to select the check box, except in the rare case that your instrument does not support this operation.

Perform Reset. Select this check box if you want a reset sent to the instrument the first time a function panel for this driver is executed. You generally want to select the check box, except in the rare case that your instrument does not support this operation. Note that all VXI instruments support this operation.

Advanced Device Configuration: Panel Driver

Here is an example of the `Panel Driver` tab of the `Advanced Device Configuration` dialog box:

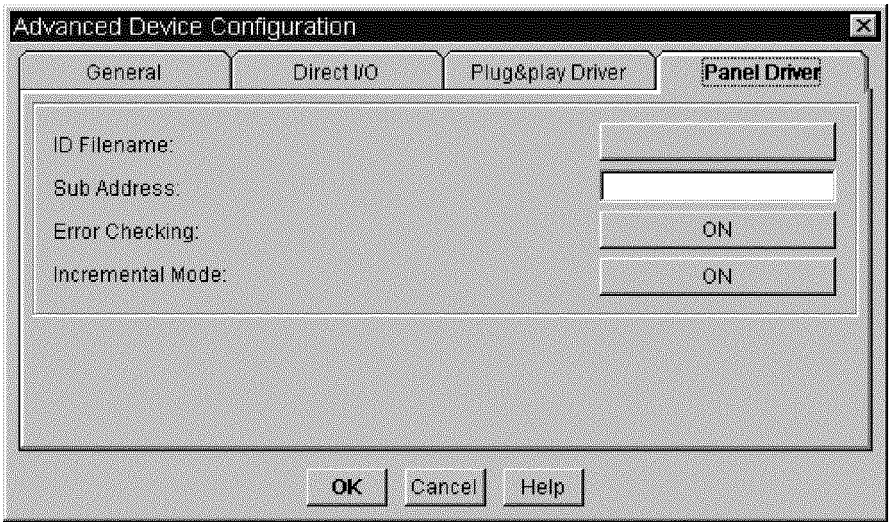

Figure 3-32. The Panel Driver Tab

NOTE

You can configure register-based VXI devices as message-based only if they are supported by I-SCPI drivers.

This tab is used to configure both `Panel Driver` and `Component Driver` objects. The following sections describe the individual fields.

ID Filename

The `ID Filename` field specifies the file that contains the desired HP Instrument Driver. Click on the field to display the `Read from what Instrument Driver?` dialog box, and choose a file. Note that files are named according to instrument model number. Be certain to choose the name corresponding to the exact model number you are using; there are similar file names, such as `hp3325a.cid` and `hp3325b.cid`.

If you are unsure which driver to use, please refer to the on-line information in `Help` ⟹ `Instruments`.

Sub Address

The `Sub Address` field specifies the subaddress used by certain drivers to identify plug-in modules in cardcage-type instruments, such as data acquisition systems and switches. If you are *not* configuring a driver for one of these plug-ins, set this field to `""` (the `NULL` string).

NOTE

Since *very* few drivers use subaddresses, the default setting of `""` (the `NULL` string) is the proper setting in 99% of all situations.

If you *are* configuring a driver for one of these plug-ins, refer to the on-line help for the instrument driver to determine if and how subaddresses are used. To get help on instrument drivers, click on `Help` ⟹ `Instruments`.

NOTE

Do not confuse the `Sub Address` field with a secondary address for HP-IB instruments. Subaddresses are part of the *driver* configuration; they are *not* part of the hardware address.

Error Checking

The `Error Checking` field determines whether or not HP VEE queries the instrument for errors after setting component values. Set this field to `ON` unless execution speed is not acceptable.

Incremental Mode

The `Incremental Mode` field specifies whether or not incremental state recall is used with `Panel Driver` objects.

> **NOTE**
>
> The proper setting for `Incremental Mode` is `ON` in 99% of all situations.

When `Incremental Mode` is set to `ON`, HP VEE automatically minimizes the number of commands sent to the instrument to change its state. To do this, HP VEE compares its record of the current state the physical instrument to the new state specified in the `Panel Driver`. HP VEE determines which component settings are different, then sends only those commands needed to change components that do not match the desired state. In most cases, you should set `Incremental Mode` to `ON`; it provides the best execution speed.

When `Incremental Mode` is set to `OFF`, HP VEE explicitly sets the values of *every* component when a corresponding `Panel Driver` operates. This is generally used only when there is a chance that HP VEE's record of the instrument state does not match the true state of the physical instrument.

Note that the `Incremental Mode` setting affects the operation of `Panel Driver` objects, but not `Component Driver` objects.

These things *do* suggest setting `Incremental Mode` to `OFF`:

- Allowing front panel operation of an instrument at any time
- Changing instrument settings outside of the HP VEE environment through C programs, HP BASIC programs, or shell commands

Using combinations of `Component Drivers`, `Panel Drivers`, and `Direct I/O` objects in a program does *not* imply that you need to set `Incremental Mode` to `OFF`.

Advanced Device Configuration: Serial

Here is an example of the `Serial` tab of the `Advanced Device Configuration` dialog box (serial interface only):

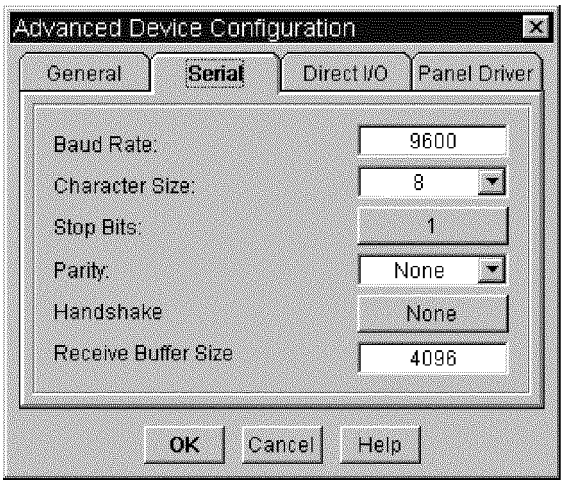

Figure 3-33. The Serial Tab

You can set the following fields for the serial (RS-232) interface:

- `Baud Rate`—The default is `9600` (bits per second).
- `Character Size`—The default is `8` (bits). Allowed values are `5`, `6`, `7`, `8`, and `None`.
- `Stop Bits`—The default is `1`. Allowed values are `1` and `2`.
- `Parity`—The default is `None`. Allowed values are `None`, `Odd`, `Even`, `Mark`, and `Space`.
- `Handshake`—The default is `None`. Allowed values are `None` and `Xon/Xoff`.
- `Receive Buffer Size`—The default is `4096` (bytes).

Advanced Device Configuration: GPIO

Here is an example of the `GPIO` tab of the `Advanced Device Configuration` dialog box (GPIO interface only):

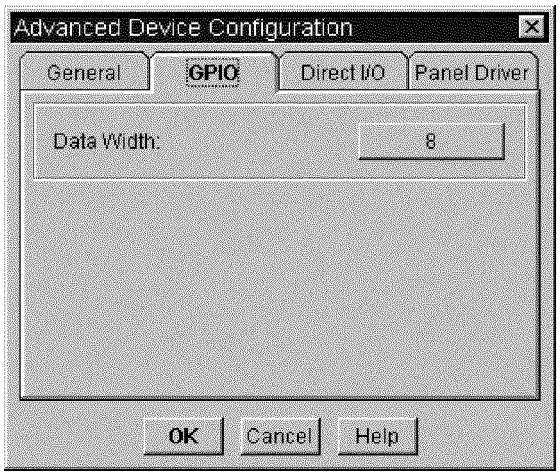

Figure 3-34. The GPIO Tab

The `GPIO` tab has only one field, `Data Width`.

The `Data Width` field specifies the number of bits of parallel data transmitted as a unit across the GPIO interface. This field configures the interface to read and write data eight or sixteen bits wide. No hardware switches need to be set in conjunction with this field.

Advanced Device Configuration: A16 Space (VXI Only)

Here is an example of the `A16 Space` tab of the `Advanced Device Configuration` dialog box. This tab appears only for the VXI interface, and is used only for register-based `Direct I/O` transactions.

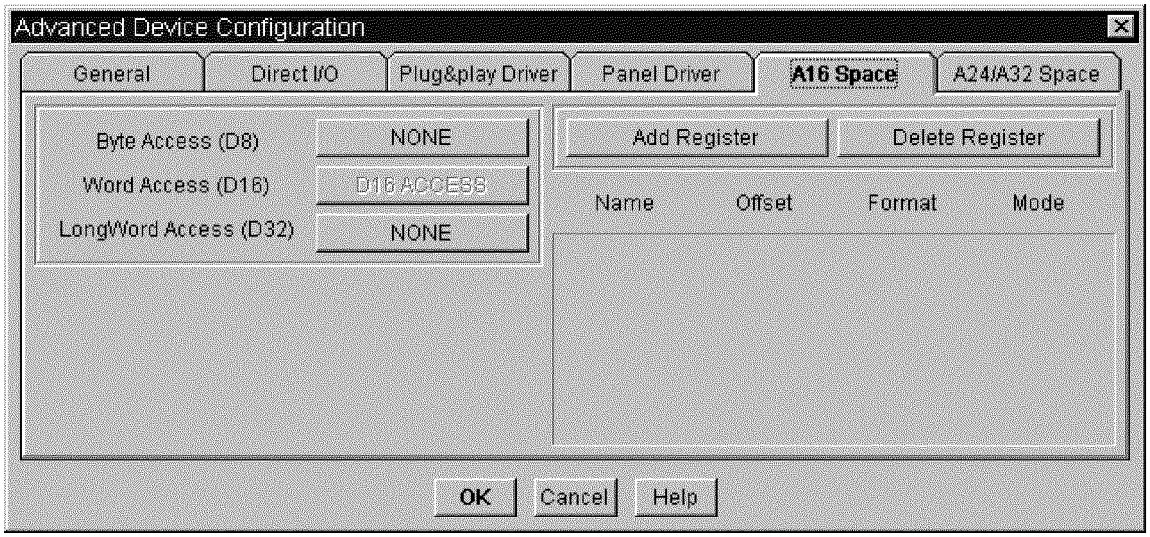

Figure 3-35. The A16 Space Tab

The following sections describe the individual fields.

Byte Access

The `Byte Access` field specifies whether the VXI device supports 8-bit A16 memory accesses. The possible choices for this field are:

- `NONE` - Device does not support byte access.
- `ODD ACCESS` - Device supports byte access, but only on odd byte boundaries (D08(O)).
- `ODD/EVEN ACCESS` - Device supports byte access on all boundaries (D08(EO)).

Word Access

The `Word Access` field is not editable. All VXI devices must support 16-bit access (D16).

LongWord Access

The `LongWord Access` field specifies whether the VXI device supports 32-bit A16 memory accesses. The possible choices are:

- `NONE` - Device does not support 32-bit access.
- `D32 ACCESS` - Device supports 32-bit A16 memory access.

Add Register

When you click on the `Add Register` field, it adds a row of fields to the dialog box. These fields allow you to configure access to a device's A16 memory. The four fields are:

- `Name` - The symbolic name of the register, which is used to refer to the particular register in a `Direct I/O` object using `READ REGISTER` or `WRITE REGISTER` transactions.
- `Offset` - The offset in *bytes* from the *relative* base of a device's A16 memory for the register being configured.
- `Format` - The data format that will be read from, or written to, the register being configured. The read or write access will take place at the byte specified in the `Offset` field. The possible formats are:
  - `BYTE` - Read or write a byte. The device must support and be configured correctly for 8-bit access by using the `BYTE` field discussed above. If the `BYTE` field is `ODD`, the byte location specified in the `Offset` field must be an odd number.
  - `WORD16` - Read or write a 16-bit word. The 16-bits are represented as a two's complement integer. All VXI devices explicitly support this format.
  - `WORD32` - Read or write a 32-bit word. The 32-bits are represented as a two's complement integer. HP VEE supports this format even if the `LongWord Access` field is specified as `NONE` (by using two D16 accesses to read or write all 32 bits). If the `LongWord Access` field is specified as `D32 ACCESS`, all 32 bits are accessed.
  - `REAL32` - Read or write a 32-bit word. The 32-bits are represented as a IEEE 754 32-bit floating-point number. HP VEE supports this format even if the `LongWord Access` field is specified as `NONE` (by using two D16 accesses to read or write all 32 bits). If the `LongWord Access` field is specified as `D32 ACCESS`, all 32 bits are accessed.

- `Mode` - Specify what I/O mode the register will support. The choices are:
  - `READ` - This register will appear as a choice in a `READ REGISTER` transaction only.
  - `WRITE` - This register will appear as a choice in a `WRITE REGISTER` transaction only.
  - `READ/WRITE` - This register will appear as a choice in both a `READ REGISTER` and `WRITE REGISTER` transaction.

Delete Register

When you click on the `Delete Register` field, it will display a list of the symbolic names of the currently configured registers. The selected register will be removed from the dialog box.

An Example

Figure 3-36 shows the `A16 Space` tab with the register configuration of an HP E1411B VXI Multimeter. Note that the list of registers scrolls as additional registers are added using `Add Register`.

NOTE

An extended (`A24/A32 Space`) memory configuration would be similar, but would consist of memory "locations," rather than "registers."

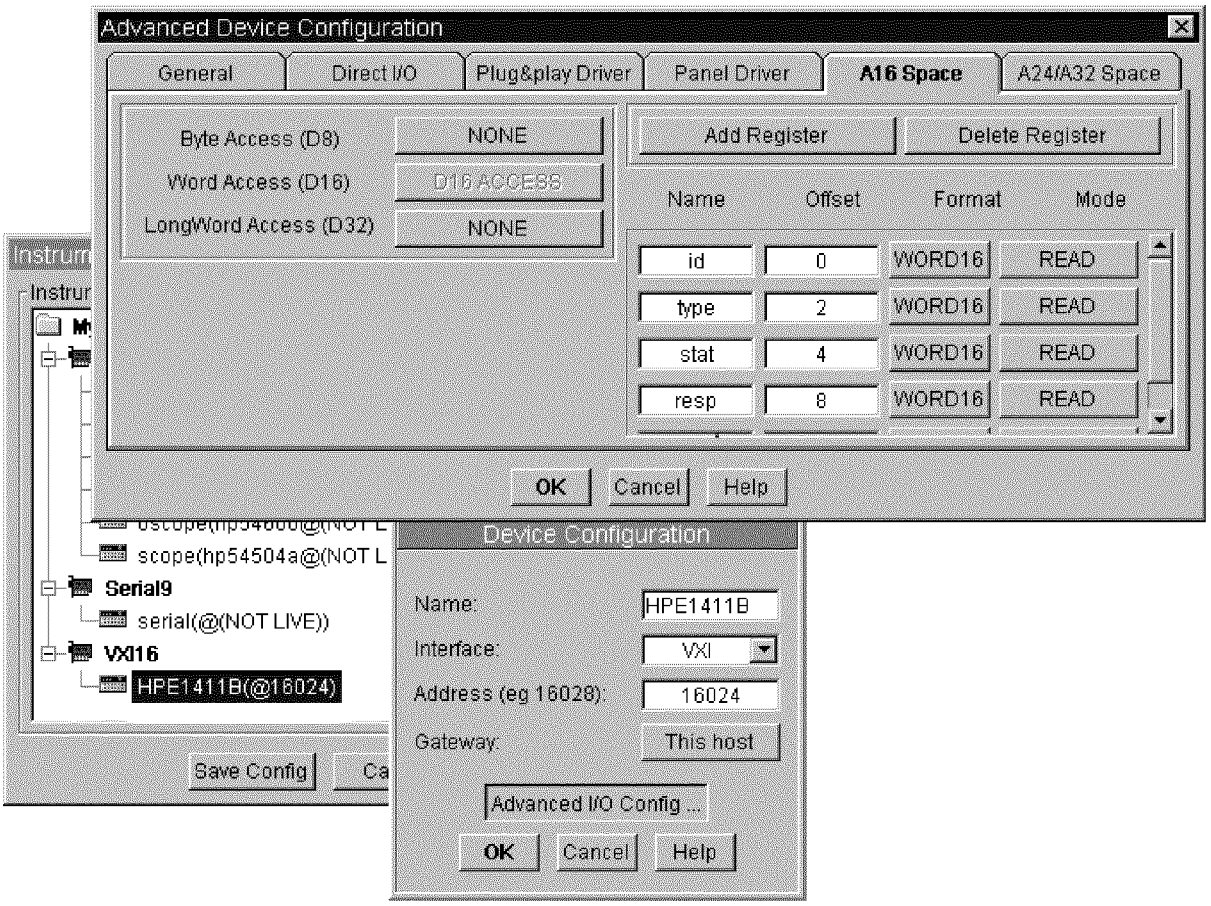

Figure 3-36. The A16 Configuration for the HP E1411B Multimeter

The `Offset` field is configured with the offset in bytes of each register from the *relative* base of the device's A16 space. The `status` register (`Name: = stat` in the figure) is configured with a 4-byte offset, and is configured for `READ` mode. The `control` register is not shown in the figure, but typically would be configured for a 4-byte offset in `WRITE` mode. While two separate register locations could have the same mode, the `Name` field must be unique. However, it would be possible for the register at byte location 4 to be named `statuscontrol` with a mode of `READ/WRITE`.

Advanced Device Configuration: A24/A32 Space (VXI Only)

Here is an example of the `A24/A32 Space` tab of the `Advanced Device Configuration` dialog box. This tab appears only for the VXI interface, and is used only for register-based `Direct I/O` transactions.

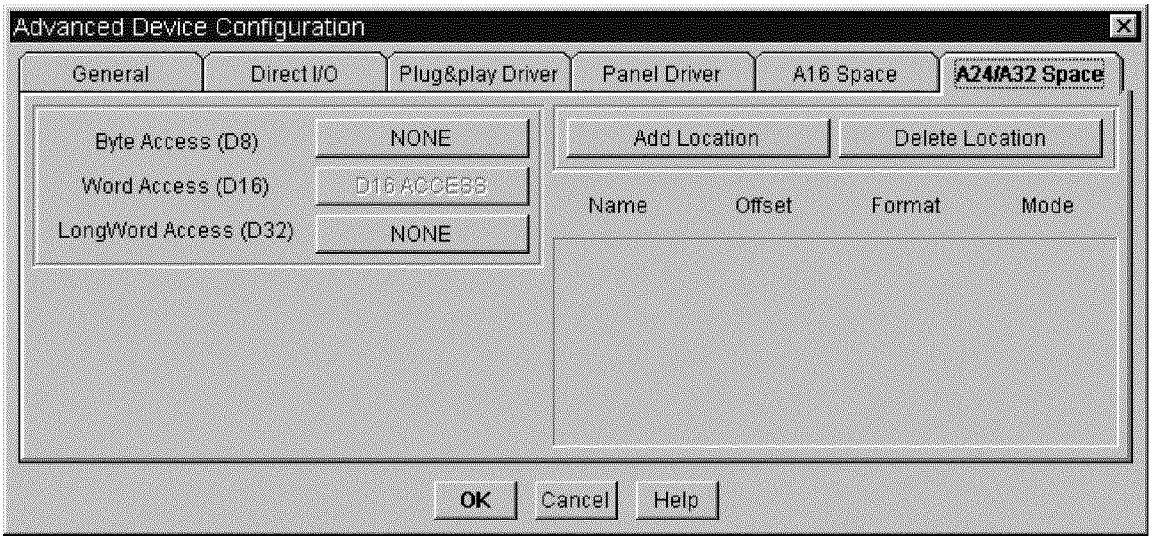

Figure 3-37. The A24/A32 Space Tab

The following sections describe the individual fields.

NOTE

The term "extended memory" indicates either A24 or A32 memory in a VXI device. (A VXI device can implement either A24 or A32 memory, but not both.)

Byte Access

The `Byte Access` field specifies whether the VXI device supports 8-bit extended memory accesses. The possible choices for this field are:

- `NONE` - Device does not support byte access.
- `ODD ACCESS` - Device supports byte access, but only on odd byte boundaries (D08(O)).
- `ODD/EVEN ACCESS` - Device supports byte access on all boundaries (D08(EO)).

Word Access

The `Word Access` field is not editable. All VXI devices must support 16-bit access (D16) for all memory spaces.

LongWord Access

The `LongWord Access` field is specifies whether the VXI device supports 32-bit extended memory accesses. The possible choices are:

- `NONE` - Device does not support 32-bit access.
- `D32 ACCESS` - Device supports 32-bit extended memory access.

Add Location

When you click on the `Add Location` field, it adds a row of fields to the dialog box. These fields allow you to configure access to a device's extended memory. The four fields are:

- `Name` - The symbolic name of the location, which is used to refer to the particular memory location in a `Direct I/O` object using `READ MEMORY` or `WRITE MEMORY` transactions.
- `Offset` - The offset in *bytes* from the *relative* base of a device's extended memory for the location being configured.
- `Format` - The data format that will be read from, or written to, the location being configured. The read or write access will take place at the byte specified in the `Offset` field. The possible formats are:
  - `BYTE` - Read or write a byte. The device must support and be configured correctly for 8-bit access by using the `BYTE` field discussed above. If the `BYTE` field is `ODD`, the byte location specified in the `Offset` field must be an odd number.
  - `WORD16` - Read or write a 16-bit word. The 16-bits are represented as a two's complement integer. All VXI devices explicitly support this format.

- WORD32 - Read or write a 32-bit word. The 32-bits are represented as a two's complement integer. HP VEE supports this format even if the LongWord Access field is specified as NONE (by using two D16 accesses to read or write all 32 bits). If the LongWord Access field is specified as D32 ACCESS, all 32 bits are accessed.
- REAL32 - Read or write a 32-bit word. The 32-bits are represented as a IEEE 754 32-bit floating-point number. HP VEE supports this format even if the LongWord Access field is specified as NONE (by using two D16 accesses to read or write all 32 bits). If the LongWord Access field is specified as D32 ACCESS, all 32 bits are accessed.

- Mode - Specify what I/O mode the location will support. The choices are:
  - READ - This location will appear as a choice in a READ MEMORY transaction only.
  - WRITE - This location will appear as a choice in a WRITE MEMORY transaction only.
  - READ/WRITE - This location will appear as a choice in both a READ MEMORY and WRITE MEMORY transaction.

Delete Location

When you click on the Delete Location field, it will display a list of the symbolic names of the currently configured location. The selected location will be removed from the dialog box.

Interface Configuration

The `Interface Configuration` dialog box appears only when you select an *inter face* in the instrument list, and then click on the `Instrument Manager` `Edit ...` button. Here is an example of this dialog box:

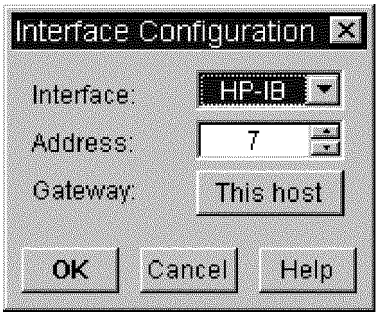

Figure 3-38. The Interface Configuration Dialog Box

The following sections describe the individual fields.

Interface

The `Interface` field specifies the type of hardware interface. You can interchange `HP-IB` with `VXI` (both are multiple-instrument buses), or `Serial` with `GPIO` (both are single-instrument interfaces).

Address

The `Address` field specifies the select code for the interface, affecting all instruments connected to it. Use the up and down arrows to change the `Address`—only the select codes without conflicts will appear.

Gateway

Use the `Gateway` field set to the name of the LAN gateway used during a remote process. Refer to "LAN Gateways" in Chapter 7 for further information.

4

Using Transactions in Direct I/O and Interface Operations

Using Transactions in Direct I/O and Interface Operations

Three HP VEE objects allow you to communicate with instruments using I/O transactions:

1. The `Direct I/O` object allows you to transmit data to and from instruments via the HP-IB, GPIB, VXI, serial, and GPIO interfaces, as well as via a LAN connection.
2. The `MultiDevice Direct I/O` object allows you to perform direct I/O transactions to multiple instruments from a single object.
3. The `Interface Operations` object allows you to send low-level HP-IB, GPIB, or VXI messages, commands, and data.

NOTE

Register-based VXI devices can be used as message-based only if supported by I-SCPI drivers.

For any of these objects, the messages are "constructed" and sent by means of I/O transactions. This chapter describes some techniques for using I/O transactions in the `Direct I/O`, `MultiDevice Direct I/O`, and `Interface Operations` objects. For further information about using I/O transactions, refer to the *HP VEE Advanced Programming Techniques* manual.

NOTE

You must properly configure HP VEE to communicate with instruments before you can use the `Direct I/O` , `MultiDevice Direct I/O`, and `Interface Operations` objects. Please refer to Chapter 3 for details.

Using the Direct I/O Object

The `Direct I/O` object allows you control an instrument directly using the instrument's built-in commands. You do not need an HP VEE instrument driver (ID) or VXI*plug&play* driver to use `Direct I/O` to control an instrument.

Sending Commands

To send commands to an instrument using `Direct I/O`, you can use `WRITE` transactions:

- The most important `WRITE` transactions for sending commands to HP-IB, GPIB, message-based VXI, register-based VXI supported by I-SCPI, and serial instruments are:
  - `WRITE TEXT`
  - `WRITE BINBLOCK`
  - `WRITE STATE`
- `Direct I/O` uses only `WRITE BINARY` and `WRITE IOCONTROL` transactions to send commands to GPIO instruments.
- `Direct I/O` uses `WRITE REGISTER` and `WRITE MEMORY` transactions to send commands to register-based and some message-based VXI instruments. These transactions are the *only* method of communicating with register-based VXI instruments not supported by I-SCPI drivers.

WRITE TEXT Transactions

Most HP-IB, message-based VXI, and serial instruments use human-readable text strings for programming commands. Such commands are easily sent to instruments using `WRITE TEXT` transactions. For example, all instruments conforming to IEEE 488.2 recognize `*RST` as a reset command. Here is the transaction used to reset such an instrument:

```
WRITE TEXT "*RST" EOL
```

Note that instruments often define very precise "punctuation" in their syntax. They may demand that you send specific characters after each command or at the end of a group of commands. In addition, HP-IB instruments vary in their use of the signal line End-Or-Identify (EOI). If you suspect that you are having problems in this area, examine the `END (EOI) on EOL` and `EOL Sequence` fields in the `Direct I/O` tab of the `Advanced Device Configuration` dialog box (refer to Chapter 3).

Please refer to your instrument programming manual to determine the proper command syntax for your instrument.

`WRITE TEXT` transactions are all that is needed to set up instruments for the majority of all situations where `Direct I/O` is required. `Direct I/O` allows you to use `WRITE` encodings other than `TEXT` when it is required by the instrument. The encodings other than `TEXT` that are most often useful are `BINBLOCK` and `STATE`.

WRITE BINBLOCK Transactions

`BINBLOCK` encoding writes data to instruments using IEEE-defined block formats. These block formats are typically used to transfer large amounts of related data, such as trace data from oscilloscopes and spectrum analyzers. Instruments usually require a significant number of commands before accepting `BINBLOCK` data. Refer to your instrument's programming manual for details.

To use `BINBLOCK` transactions, you *must* properly configure the `Conformance` field (and possibly `Binblock`) in the `Direct I/O` tab of the `Advanced Device Configuration` dialog box (refer to Chapter 3).

WRITE STATE Transactions

Some HP-IB and message-based VXI instruments support a learn string capability, which allows you to upload all of the instrument settings. Later, you can recall the measurement state of the instrument by downloading the learn string using a `WRITE STATE` transaction. Learn strings are particularly useful when you wish to download measurement states but an instrument driver is unavailable.

Note that `WRITE STATE` transactions are available for HP-IB and message-based VXI instruments only.

Here is the typical procedure for using learn strings:

1. Configure the instrument to the desired measurement state; typically this is done using the instrument front panel.
2. Click on `Upload State` in the object menu of a `Direct I/O` object configured for the instrument. The instrument state is now associated with this particular instance of the `Direct I/O` object.
3. Add a `WRITE STATE` transaction to the `Direct I/O` object.

When it is used, `WRITE STATE` is generally the first transaction in a `Direct I/O` object. `WRITE STATE` writes the `Upload`ed learn string to the instrument, thus setting all instrument functions simultaneously. Subsequent `WRITE` transactions can modify the instrument setup in an incremental fashion.

The behavior of `Upload` and `WRITE STATE` for HP-IB and message-based VXI instruments is affected by the `Direct I/O` tab settings for `Conformance` and `State (Learn String)`. If `Conformance` is `IEEE 488.2`, HP VEE will automatically handle learn strings using the IEEE 488.2

`*LRN?` definition. If `Conformance` is `IEEE 488`, `Upload String` specifies the command used to query the state, and the `Download String` specifies the command that precedes the string when it is downloaded. Note that message-based VXI instruments, and register-based VXI instruments supported by I-SCPI are `IEEE 488.2` compliant.

Clicking on `Upload State` in the `Direct I/O` object menu has these results:

- The learn string is uploaded *immediately*.
- The learn string remains with that particular `Direct I/O` object as long as it exists, until the next `Upload`. *The learn string is saved with the program*.
- If you clone a `Direct I/O` object, its associated learn string is included in the clone.

Learn String Example

Assume you wish to program the HP 54100A digitizing oscilloscope using learn strings. The oscilloscope's programming manual contains these important facts:

- The oscilloscope conforms to IEEE 488; it does not conform to IEEE 488.2.
- The command used to query the oscilloscope's learn string is `SETUP?`.

- The command that must precede a learn string that is downloaded to the instrument is `SETUP` . Note that a space must come between the `P` in `SETUP` and the first character in the downloaded learn string.

You must use the `Instrument Manager` (refer to Chapter 3) to specify the proper direct I/O configuration for the oscilloscope. The settings important to learn strings are shown in Figure 4-1.

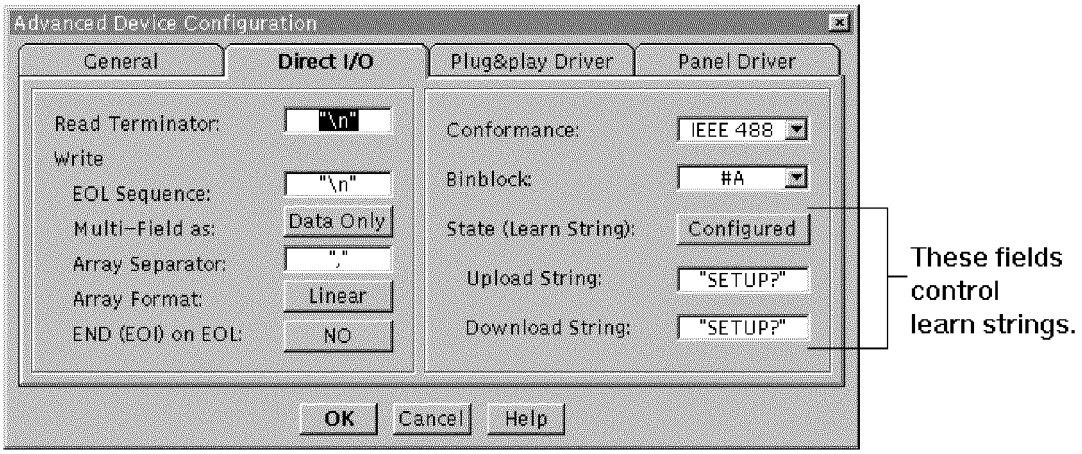

Figure 4-1. Configuring for Learn Strings

To upload a learn string from the oscilloscope, click on `Upload` in the object menu of a `Direct I/O` object that controls the oscilloscope. To download the learn string, use this transaction:

```
WRITE STATE
```

Reading Data

To read data from an instrument using `Direct I/O`, you can use `READ` transactions.

> **NOTE**
>
> Instruments return data in a variety of formats. In general, you must know *what kind* of data and *how much* data you want HP VEE to read from an instrument. The kind of data determines the encoding and format you must specify in the transaction. The amount of data being read determines the configuration you must use for the `SCALAR` or `ARRAY` fields in the transaction dialog box.

- The most important `READ` transactions for `Direct I/O` use with HP-IB, GPIB, message-based VXI, and serial instruments are:
  - `READ TEXT`
  - `READ BINBLOCK`
- `Direct I/O` uses only `READ BINARY` and `READ IOSTATUS` transactions to read data from GPIO instruments.
- `Direct I/O` uses `READ REGISTER` and `READ MEMORY` transactions to read data from register-based and some message-based VXI instruments. These transactions are the *only* method of communicating with register-based VXI instruments not supported by I-SCPI.

If you have difficulty reading data from instruments, try using the `Bus I/O Monitor` to examine the data to determine its format.

READ TEXT Transactions

Frequently, the data you read from an instrument as the result of a query is a single numeric value that is formatted as text. For example, a particular voltmeter returns each reading as a single number in exponential notation, such as `-1.234E+00`. Here is the transaction to read a value from the voltmeter:

```
READ TEXT a REAL
```

Some instruments respond to a query with alphabetic information combined with the numeric measurement data. In general, this not a problem; `READ TEXT REAL` transactions throw away preceding alphabetic characters and extract the numeric value.

NOTE

When reading numeric data from an instrument, the data type of the instrument data is automatically converted, if necessary, according to the rules listed in Appendix C.

Using the MultiDevice Direct I/O Object

The `MultiDevice Direct I/O` object (`I/O` ⟹ `Advanced I/O` ⟹ `MultiDevice Direct I/O`) lets you control several instruments from a single object using direct I/O transactions. The object is a standard transaction object, and works with all interfaces that HP VEE supports. It appears the same as the `Direct I/O` object, except each transaction in `MultiDevice Direct I/O` can address a separate instrument. Since the `MultiDevice Direct I/O` object does not necessarily control a particular instrument as the `Direct I/O` object does, the title does not list an instrument name, address, or live mode condition.

By using the `MultiDevice Direct I/O`, you can reduce the number of instrument-specific `Direct I/O` objects in your program, which optimizes icon-to-icon interpretation time. This performance increase is especially important for the VXI interface, which is faster than HP-IB (GPIB) at instrument control. The following figure shows the `MultiDevice Direct I/O` object and its `I/O Transaction` dialog box configured to communicate with an HP E1413B, HP E1328, and HP 3325.

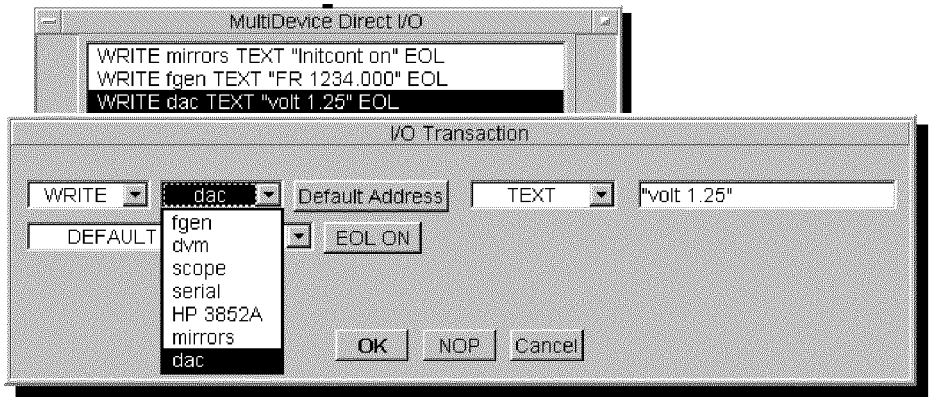

Figure 4-2. MultiDevice Direct I/O Controlling Several Instruments

Transaction Dialog Box

The `I/O Transaction` dialog box is similar to the one used by `Direct I/O`, except it contains two additional fields. The common fields work the same way. The following sections describe the two additional fields.

Device Field

The `Device Field` contains the name of any of the currently configured instruments. Clicking on the down arrow presents a list of available configured instruments. You can select a different instrument for each transaction.

Address Field

The `Address Field` specifies the address of the device showing in the `Device Field`. The `Address Field` has two modes—`Default Address` and `Address:`. `Default Address` sets HP VEE to use the address entered when the instrument was originally configured. `Address:` includes a text box that lets you enter a different address. You can enter a specific numeric value, a variable name, or an expression. The entry must evaluate to a valid address. The value entered for `Address:` will change the device's address when the object executes, which is like the address control pin action. The following figure shows the `I/O Transaction` dialog box using `Address:`.

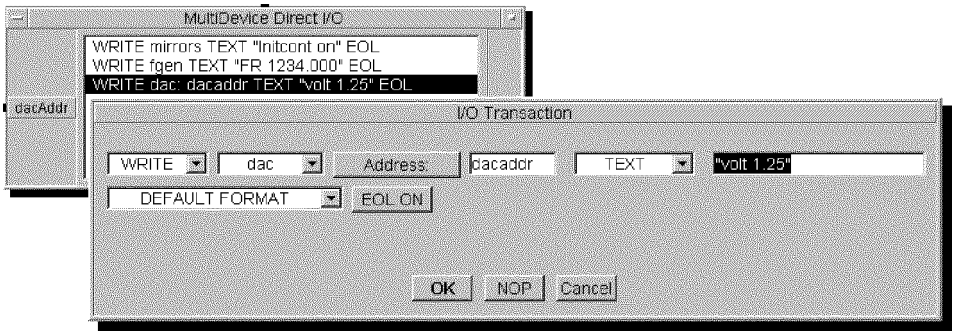

Figure 4-3. Entering an Instrument Address as a Variable

Editing Transactions

As you edit transactions using the `I/O Transaction` dialog box, only those transactions allowed by the type of instrument are accepted. For example, if the name showing in the `Device Field` is configured as a VXI device controlled via the VXI backplane, then you can configure a `REGISTER` or `MEMORY` access transaction.

If the `I/O Transaction` dialog box is configured for a particular type of transaction and you change the `Device Field` name, then the transaction must remain correct for the different instrument. If the transaction is incorrect, entries in the `I/O Transaction` dialog box will change to the last valid transaction for that instrument type. A `REGISTER` access transaction for a VXI device would be incorrect if you change the `Device Field` name to a non-VXI instrument.

Object Menu

The object menu for `MultiDevice Direct I/O` is similar to that of the the `Direct I/O` object. The `MultiDevice Direct I/O` menu does not include the `Show Config ...` or `Upload State` menu choices. These menu choices are for specific instrument configurations. Use the `Direct I/O` object to show an instrument's configuration or to upload a physical instrument's settings.

There is no live mode indicator for any of the possible devices in the transactions. To control live mode for an instrument, click on `I/O` ⟹ `Instrument Manager ...` , and then edit the selected instrument configuration.

Using the Interface Operations Object

The `Interface Operations` object (`I/O` ⟹ `Advanced I/O` ⟹ `Interface Operations`) allows you to control HP-IB, GPIB, VXI, and serial instruments using low-level commands. `Interface Operations` supports two types of transactions that provide this low-level control: `EXECUTE` and `SEND`.

The EXECUTE Transaction

`EXECUTE` transactions are of the form:

`EXECUTE` *Command*

where *Command* is one of the bus commands summarized in Table 4-1. While the commands listed in Table 4-1 have the same names as the `EXECUTE` commands in `Direct I/O`, there is an important difference.

- `Direct I/O EXECUTE` commands address an instrument to receive the command.
- `Interface Operations EXECUTE` commands may affect multiple instruments. For HP-IB, these instruments must be addressed to listen.

Table 4-1. Summary of EXECUTE Commands (Interface Operations)

Command	Description
`ABORT`	Clears the HP-IB interface by asserting the IFC (Interface Clear) line. To clear and reset the VXI interface use `CLEAR`.
`CLEAR`	Clears all HP-IB devices by sending DCL (Device Clear). For VXI, resets the interface and runs the Resource Manager.
`TRIGGER`	For HP-IB, triggers all devices addressed to listen by sending GET (Group Execute Trigger). For VXI, triggers TTL, ECL, or external triggers.
`REMOTE`	For HP-IB, asserts the REN (Remote Enable) line. There is no counterpart for VXI.
`LOCAL`	For HP-IB, releases the REN (Remote Enable) line. There is no counterpart for VXI.
`LOCAL LOCKOUT`	For HP-IB, sends the LLO (Local Lockout) message. Any device in remote mode at the time LLO is sent will lock out front panel operation. There is no counterpart for VXI.
`LOCK INTERFACE`	In a multi-process system with shared resources, lets one process lock the resources for its own use during a critical section to prevent another process from trying to use them.
`UNLOCK INTERFACE`	In a multi-process system where a process has locked shared resources for its own use, unlocks the resources to allow other processes access to them.
`PASS CONTROL`	Passes control to an HP-IB device at the specified address, provided the device is capable of becoming the active controller. There is no counterpart for VXI.

The SEND Transaction

`SEND` transactions are of this form:

`SEND` *BusCmd*

where *BusCmd* is one of the bus commands listed in Table 4-2. These messages are defined in detail in IEEE 488.1. *BusCmd* is HP-IB specific only. There are no counterparts for VXI.

Table 4-2. SEND Bus Commands

Command	Description
COMMAND	Sets ATN true and transmits the specified data bytes. ATN true indicates that the data represents a bus command.
DATA	Sets ATN false and transmits the specified data bytes. ATN false indicates that the data represents device dependent information.
TALK	Addresses a device at the specified primary bus address (0-30) to talk.
LISTEN	Addresses a device at the specified primary bus address (0-30) to listen.
SECONDARY	Specifies a secondary bus address following a TALK or LISTEN command. Secondary addresses are typically used by card cage instruments where the card cage is at a primary address and each plug-in module is at a secondary address.
UNLISTEN	Forces all devices to stop listening; sends UNL.
UNTALK	Forces all devices to stop talking; sends UNT.
MY LISTEN ADDR	Addresses the computer running HP VEE to listen; sends MLA.
MY TALK ADDR	Addresses the computer running HP VEE to talk; sends MTA.
MESSAGE	Sends a multi-line bus message. Consult IEEE 488.1 for details. The multi-line messages supported by HP VEE are: DCL Device Clear SDC Selected Device Clear GET Group Execute Trigger GTL Go To Local LLO Local Lockout SPE Serial Poll Enable SPD Serial Poll Disable TCT Take Control

5

Using VXIplug&play Drivers

Using VXIplug&play Drivers

This chapter provides further information about using VXI*plug&play* drivers with HP VEE. In order to use a VXI*plug&play* driver to communicate with an instrument, you must first install the appropriate VXI*plug&play* driver files and the VISA I/O library (refer to "Introduction to VXI*plug&play*" in Chapter 2). You must also configure HP VEE for the instrument as described in "Configuring for a VXI*plug&play* Driver" in Chapter 3.

The primary means of communicating with a VXI*plug&play* driver in HP VEE is the `To/From VXIplug&play` object, which is described in the following section. In addition, you can call VXI*plug&play* functions from HP VEE `Call` objects (refer to "Using VXI*plug&play* Functions from Call Objects"). The latter method is provided for backward compatibility with HP VEE version 3.1.

Program Compatibility

Two previous versions of HP VEE have supported VXI*plug&play* drivers:

- HP VEE version 3.2 provided the `To/From VXIplug&play` object. HP VEE 3.2 programs using this object are compatible with HP VEE version 4.0.
- HP VEE version 3.1 provided *only* direct `Call` access to VXI*plug&play* drivers. If you used `Call` objects to control VXI*plug&play* instruments in HP VEE version 3.1, your program will still work in HP VEE version 4.0 once you make certain changes. You must reinstall the Windows 95 version of VISA and the 32-bit version of the VXI*plug&play* driver, and you may have to change the `Import` objects to use the new location of the VXI*plug&play* driver files. For more information on using `Call` objects to access VXI*plug&play* drivers, refer to "Using VXI*plug&play* Functions from Call Objects".

Using the To/From VXIplug&play Object

After you have added VXI*plug&play* instruments to the HP VEE instrument configuration, you can use the VXI*plug&play* drivers in your program. You access the instruments by the functions contained in the drivers. The `To/From VXIplug&play` object provides access to the VXI*plug&play* function panels.

To get the `To/From VXIplug&play` object:

1. Select `I/O` ⟹ `Instrument Manager`. The `Instrument Manager` appears. It displays all currently configured VXI*plug&play* instruments (as well as any other instruments that are configured).
2. Select the instrument with which you want to communicate, and click on the `Plug&play Driver` button. The outline of the object appears.
3. Place the outline of the `To/From VXIplug&play` object where you want it in the work area and click the mouse button. The object appears as shown below:

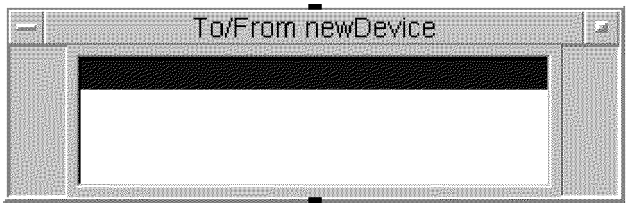

Figure 5-1. To/From VXIplug&play Object

Selecting a Function

You can select the VXI*plug&play* functions from the `To/From VXIplug&play` object.

1. Double-click on an empty transaction or select `Add Trans` or `Insert Trans` from the object menu. The `Select a Function Panel` dialog box appears. It displays function panels grouped into logical categories (such as `Measure` or `Configure`) as shown in Figure 5-2. Each driver has different categories.

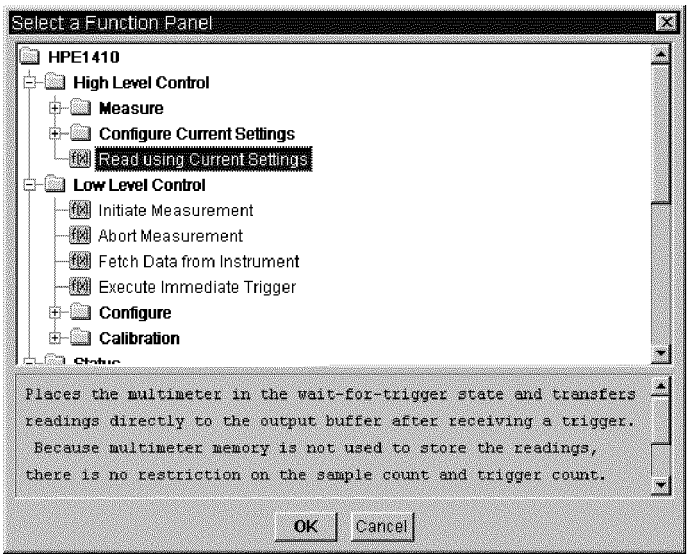

Figure 5-2. Select a Function Panel Dialog Box

- Click on the `[+]` icons to view the hierarchical structure of function panels.
- Click on the `[-]` icons to hide the function panels in the hierarchical structure.
- Click on the `[f(x)]` icons to select the function panel. You'll see a short description of the function panel in the lower part of the dialog box.

To completely expand a branch of the tree, select the item to expand and press the (*) key.

Generally, you'll see only function panels that adhere to the VXI*plug&play* version 3.x specification and are allowed by HP VEE.

NOTE

HP VEE automatically calls `init()` at the appropriate time, however, there may be other initialization functions such as `init_all()`, `init_next()`, or `init_first()` functions in the list. These functions are not defined in the VXI*plug&play* specification and therefore are not supported by HP VEE. Do not select these functions. If you must use these functions, you need to create your program differently and call the VXI*plug&play* driver from a `Call` object as described in "Using VXI*plug&play* Functions from Call Objects."

Note that there are no entries for *PREFIX*`_init()` or *PREFIX*`_close()`. These functions are performed automatically by HP VEE.

2. Click `OK` on the `Select a Function Panel` dialog box.
3. You'll see a tabbed dialog box called `Edit a Function Panel` that allows you to specify the parameters for the function panel.

Editing Function Panel Parameters

The `Edit a Function Panel` dialog box allows you to set controls and variables to pass to the selected VXI*plug&play* driver's function. There are two tabs, `Panel` and `Parameter`.

The Panel Tab. The `Panel` tab allows you to specify the constant (control) values to pass to the function.

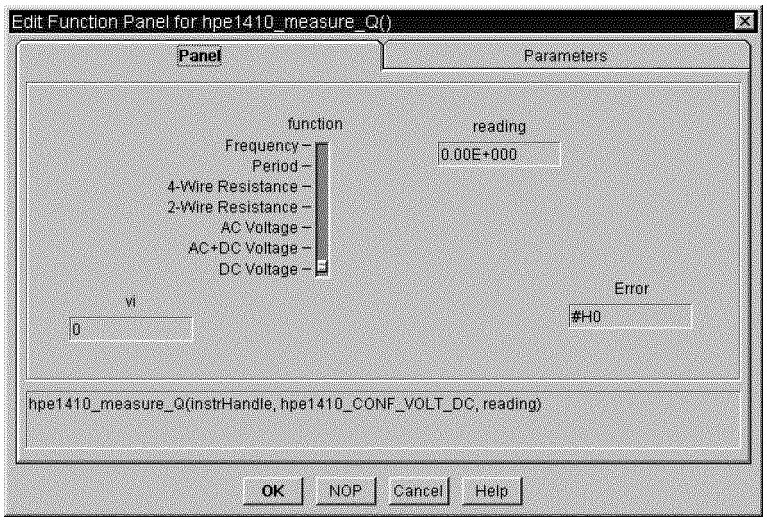

Figure 5-3. Panel Tab of Edit Function Panel Dialog Box

- Controls - The top part of this dialog box contains controls for you to specify constant parameters. The names of the controls are labels specified from the function panel file.
- `vi` - Displays the unique "virtual instrument" handle (also called the "session handle") of the instrument. Depending on the driver version, the name of this field may change, but the location will always be in the lower-left corner of the function panel.
- `Error` - Displays a non-zero value if an error occurred when executing this function panel. Depending on the driver version, the name of this field may change, but the location will always be in the lower-right corner of the function panel.
- Function call - At the bottom of the dialog box is the C function and the parameters that are sent to the driver when the object executes. This

command string is also shown as a transaction on the open view of the object.

Getting Help on a VXI*plug&play* Function Panel

When in the `Edit Function Panel` dialog box, click the right mouse button on the background of the `Panel` tab for help on the function panel. A dialog box containing a description of the function appears.

Click the right mouse button on a control (not the label) for an explanation of the parameter.

For complete help on the VXI*plug&play* driver, select `Instrument Help` from the object menu of the `To/From VXIplug&play` object.

The Parameters Tab. The `Parameters` tab allows you to specify the variables to pass to the function. This allows you to set the parameter values programmatically.

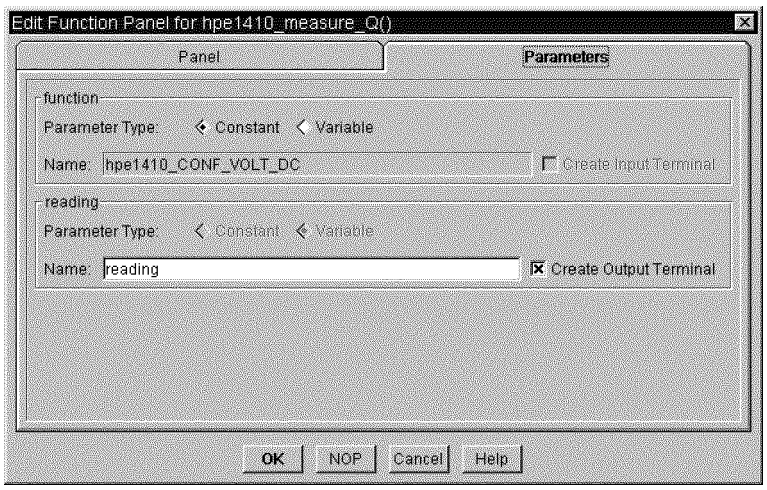

Figure 5-4. Parameter Tab of Edit Function Panel Dialog Box

- Group name - The name of each group is the label name of the parameter as specified in the VXI*plug&play* function panel. In Figure 5-4, `function` and `reading` are labels.
- `Parameter Type` - When `Constant` is selected, this parameter is passed as a constant value that is set on the `Panel` tab. When `Variable` is selected, this parameter is passed as a variable. That is, the value of the parameter may be changed programmatically. Some fields are always variables, such as the output for a reading.
- `Name` - When the `Parameter Type` is set to `Variable`, this field is editable. By default, the name of the variable is set to its label name (or a similar name to make it a valid HP VEE variable name). You can change this to any valid variable name in HP VEE. If the variable is an input variable, you can also put an expression, function call, or global variable in this edit field.
- `Create Terminal` - When the `Parameter Type` is set to `Variable`, this field is editable. When the check box is checked, and `Name` does not currently exist as a terminal name, when you press `OK`, the terminal (with

the name specified in `Name`) is created (as an input, output, or input/output terminal as indicated in the dialog box). To delete a terminal once it is created, you must use `Delete Terminal` from the object menu.

If the `Name` is changed, and `Create Terminal` is checked, a new terminal will be added.

If the `Name` is set to an invalid terminal name, `Create Terminal` is grayed out.

Press the `NOP` button to save the latest settings shown in this dialog box and make this transaction a "no operation". This is the same as commenting out a line of code in a text-based computer program.

Press the `Help` button for help about the `To/From VXIplug&play` object.

Press the `OK` button when you're finished editing.

The Auto-Allocate Feature (Passing Arrays and Strings). Some VXI*plug&play* functions desire to return data in an array or `Text` string. The VXI*plug&play* specification requires that the application (HP VEE) allocate the memory for the array or string—the VXI*plug&play* function cannot pass back allocated memory.

The `Auto-Allocate` feature lets you easily tell HP VEE how much memory to allocate. HP VEE takes care of allocating the correct data type and shape, in the size required.

If a parameter to a function is a variable that requires an array or a `Text` string, the `Parameters` tab displays an additional field: `Auto-Allocate input`. For example, in the dialog shown below, `readings` can input an array. The `Parameters` tab shows `Auto-Allocate input` selected:

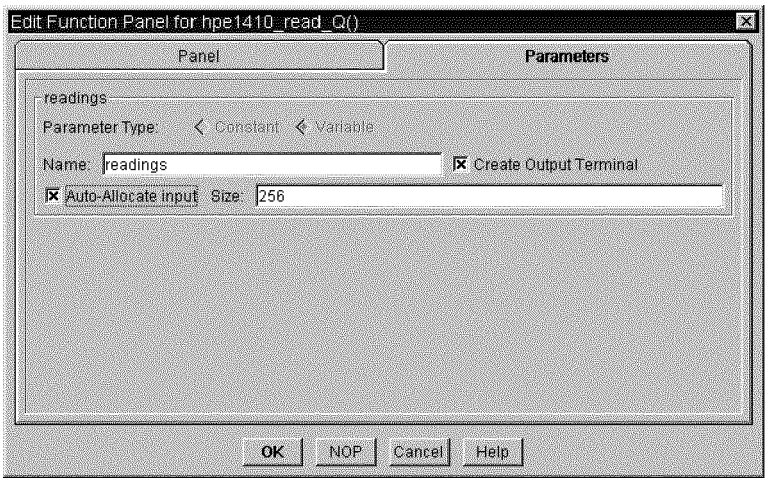

Figure 5-5. Selecting the Auto-Allocate Input Feature

When `Auto-Allocate input` is selected, the `Size` field becomes active. The default size is 256, but you can enter any appropriate size to allocate the input data. You must determine how large an array or string needs to be passed. An input terminal is not created for this parameter, and HP VEE automatically allocates the memory for the parameter. For an array, `Size` denotes the number of elements in the array. For a text string, `Size` denotes the number of characters (bytes). Refer to `Instrument Help`, or click the

right mouse button on the `Panel` background or on the parameter to obtain more information on the size of array or string the function requires.

NOTE

If you use the `Auto-Allocate input` feature, a data input terminal is *not* created for the function. If the data input terminal already exists, you should delete it from the `To/From VXIplug&play` object.

If you do not select ("check") `Auto-Allocate input`, both input and output terminals are created for the function by default. You must create an object to allocate the correct type, shape, and amount of memory, and connect it to the input terminal. Refer to "Passing Parameters", later in this chapter, for information on how to manually allocate the memory needed for inputs.

Getting Help on a VXIplug&play Driver

From the object menu of the `To/From VXIplug&play` object, select `Instrument Help`. This accesses the help file provided by the instrument manufacturer. This help topic contains information about using the VXI*plug&play* driver including the data types required for the parameters.

For help on each particular function, refer to the note in the previous section, "Getting Help on a VXI*plug&play* Function Panel".

Running an HP VEE Program

The transactions in the `To/From VXIplug&play` object execute from top to bottom. This section explains what happens when `To/From VXIplug&play` objects execute.

Initializing and Closing Drivers

The first time you run a program after you load or create it, there will be a delay to initialize each instrument controlled with `To/From VXIplug&play` objects. This initialization is done to set the instrument to a known initial state. Each successive time you run the program (after the first time), your program will execute normally, without performing the initialize actions.

Each instrument controlled by the program must be initialized once in a HP VEE session. The VXI*plug&play* Resource Manager does an "instrument find" to verify the instrument is connected to the address and to set the instrument to a known state. This will take an indeterminate amount of time, possibly up to 10 seconds per instrument. This delay happens the first time the `To/From VXIplug&play` object for each instrument is executed.

Because the initialization is only performed once per HP VEE session, you should execute functions (such as clear or reset) that set an instrument to a known state every time the program runs.

When you load another program or exit HP VEE, the VXI*plug&play* drivers are automatically closed.

In More Detail. This section explains some of the details behind some of the HP VEE internal implementation of VXI*plug&play* initialization. Understanding these concepts is not required to successfully write an HP VEE program that uses VXI*plug&play* drivers.

Each VXI*plug&play* driver is required to have a *PREFIX*`_init()` and a *PREFIX*`_close()` function. These functions are called automatically by HP VEE.

The purpose of the `init()` function is to set your instrument to a known state and to get a "session handle". Each instrument specified by an `HP VEE Name` (when configured) will have a unique session handle assigned to it the first time it is executed in a program. That session handle is used through the remainder of your program to uniquely identify that particular instrument. All `To/From VXIplug&play` objects communicating with the same instrument (with the same `HP VEE Name`) are identified by the same session handle. The session handle is shown in the `vi` field in the lower left corner of `Panel` tab of the function panel. HP VEE automatically takes care of passing this session handle between the various `To/From VXIplug&play` objects.

Because the `init()` call is usually a lengthy operation, it is only called when necessary. When the first `To/From VXIplug&play` object is executed in a program, the appropriate `init()` function is called. When `init()` is called, it may also perform an `Identification Query` and/or a `Reset` depending on how you configured the driver.

The purpose of the `close()` function is to close the session handle (there are a limited number of them), take the instrument off-line, clear any data associated with the instrument, and may perform instrument-specific actions. HP VEE calls the `close()` function at the following times:

- After `New`, `Open`, or `Exit` is selected.
- When all `To/From VXIplug&play` objects for a single HP VEE name (such as `dvm`) are deleted.
- When the `Address` or `init()` parameter values are changed in the `VXIplug&play Device Configuration` dialog box. In this case, `close()` is called so that `init()` will be called again with the new values.

Error and Caution Checking

After each transaction is executed, the function returns a status value to HP VEE. HP VEE automatically checks this value, and if it indicates the function executed successfully, the next transaction executes.

Error Checking. If the status value returned is an error, HP VEE stops the program and reports the error to you. If you have an error output pin to trap the error, the error does not stop the program. Use the `errorInfo()` object to get the details of the error message. HP VEE will automatically call the *PREFIX*`_error_message()` function to get as much error information from the VXI*plug&play* driver as the manufacturer includes. This information is output in the HP VEE error message or from `errorInfo()`. Note that after an error occurs, the instrument will be left in an unknown state. Unless you call specific reset or clear functions at the beginning of your program, you won't know the state of your instruments the next time you start the program.

Caution Checking. If the status value returned is a caution, the HP VEE program pauses and displays a caution dialog box. The caution dialog box contains information from the instrument manufacturer and lets you choose to continue running the program or to stop.

Caution messages cannot be trapped programmatically. However, if you are aware of the common caution messages from the driver, you can handle them in the HP VEE program. For example, if you get a caution message that the instrument is not ready to let you read data, you can use a `Delay` object or put the `To/From VXIplug&play` object in a loop to retry reading. If you handle a known caution condition in the HP VEE program, you may want to suppress the caution message dialog box. To do this, from the `To/From VXIplug&play` object's `Properties` dialog box, select the check box for `Ignore Cautions Returned`. Generally, ignoring caution messages (by checking the `Ignore Cautions Returned` check box) is not necessary and, unless you are sure of how to handle the caution condition in your program, it is discouraged.

Passing Parameters

According to the VXI*plug&play* specification, you must allocate memory and pass it to the driver before requesting data. Some VXI*plug&play* functions place the data read into an array. Most of these VXI*plug&play* functions also have a parameter which specifies the size of the array sent in, and will error if the array is not big enough. In this case, you may allocate an array of any size and tell the function how big it is. The function will then write data into the array only to the size specified.

CAUTION

Other VXI*plug&play* functions simply assume the array passed in is big enough for the data read, and will write to it regardless of its size. This is especially common for `Text` strings. If insufficient memory is allocated, this action will overwrite memory and cause a General Protection Fault or Segmentation Violation. Since the VXI*plug&play* shared library is linked directly into HP VEE, this situation can cause HP VEE to crash and exit.

The Easy Way

The most straightforward method to allocate memory for an array or string data input is to use the `Auto-Allocate` feature. Refer to "The Auto-Allocate Feature (Passing Arrays and Strings)" earlier in this chapter. You will still need to determine the size to allocate, but once you specify the size, the memory will be allocated automatically.

Find out how much memory you need for your data by reading the driver's help file. Select `Instrument Help` from the `To/From VXIplug&play` object's object menu. This help file will tell you how large the array must be.

If you do not use `Auto-Allocate`, you must create an object to allocate the memory, and connect it to the data input terminal of the `To/From VXIplug&play` object:

- For an array input, use an `Alloc Array` object of the appropriate type, and set the size appropriately.
- For a string input, use a `Formula` object. Delete the data input terminal from the `Formula` object and enter an expression like `256*"a"`. This will create a string that is 256 characters long (plus a null byte) filled with `a`'s. Most VXI*plug&play* functions will not write more than 256 characters into a `Text` parameter. However, it is best to check the help on each function panel that requires a `Text` input to be sure.

An Example Program

Figure 5-6 shows a simple program that uses `To/From VXIplug&play` objects to communicate with the HP E1410A VXI Multimeter:

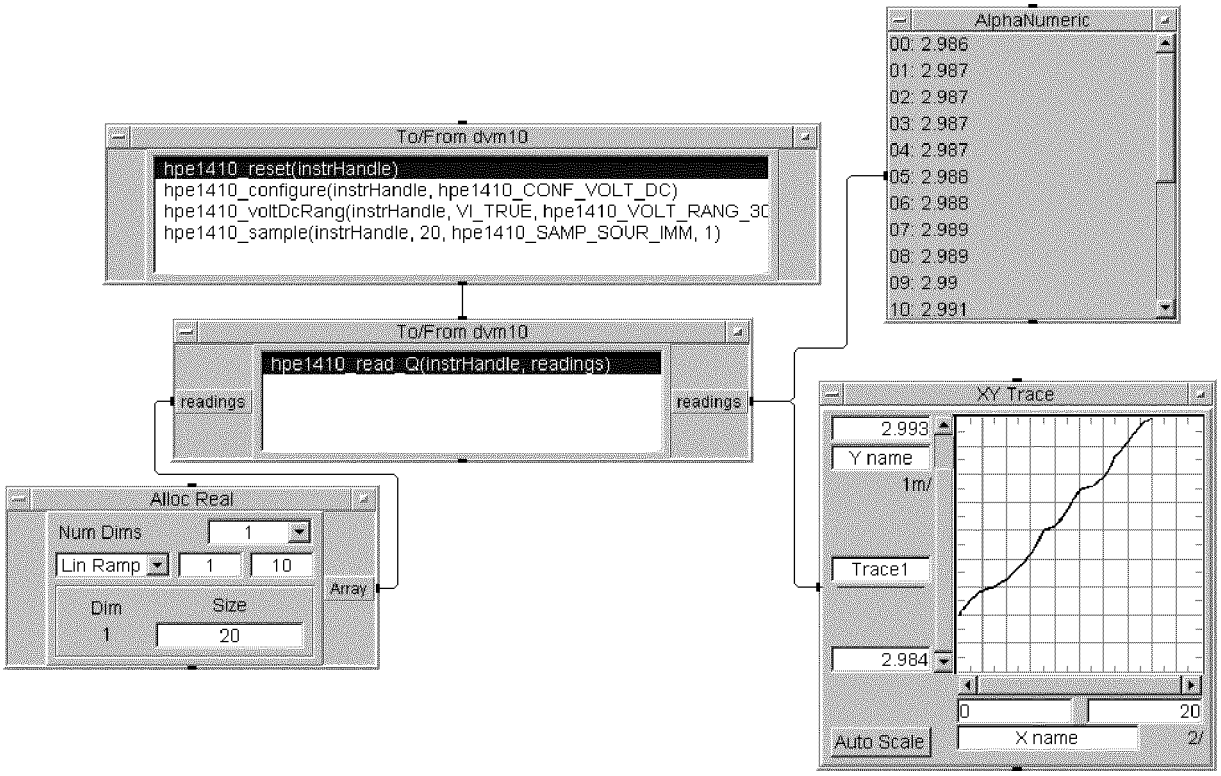

Figure 5-6. A Program Using To/From VXIplug&play Objects

Limitations to VXI*plug&play*

There are some limitations to using VXI*plug&play* drivers in HP VEE.

- Because the `Bus I/O Monitor` object only shows I/O to and from HP VEE itself, it does not show any I/O done from VXI*plug&play* drivers. VXI*plug&play* drivers are just C programs that are linked into HP VEE. If needed, it is recommended that you use a hardware bus monitor.
- Some optional features that are not required by the VXI*plug&play* specification, such as callbacks, are not supported by HP VEE.
- All `I/O` ⟹ `Advanced I/O` objects (including `Interface Operations`, `Device Event` (`SPOLL`), and `Interface Event`) are not supported for VXI*plug&play*.
- VXI*plug&play* does not support the concept of `LIVE MODE/NOT LIVE MODE`. When you *run* a program, all instruments used in your program must be connected to your computer. However, you can *open* a program without the instruments used in the program being connected. Also, you can *create* a program without having the instruments connected. You can use `To/From VXIplug&play` objects and specify the function calls as long as the VXI*plug&play* driver is installed.
- You *cannot* use VXI*plug&play* drivers and any of the other HP VEE instrument control methods (`Direct I/O`, `Panel Driver`, or `Component Driver` objects) to communicate with the same instrument in the same program. However, you can use VXI*plug&play* drivers for one instrument, and other instrument control methods for other instruments in the same program.

NOTE

The VXI*plug&play* specification is continually being updated and enhanced. New features may be voted into the specification by the VXI*plug&play* consortium between revisions of HP VEE. Because the VXI*plug&play* specification does not specify that revision information should be included in the driver library, HP VEE cannot check the driver for compatibility. Therefore, you need to check with the instrument manufacturer to make sure the driver conforms to the currently supported VXI*plug&play* specification.

Using VXIplug&play Functions from Call Objects

You may want to use VXI*plug&play* with an HP VEE `Call` object because of the following reasons:

- Existing Program Compatibility.

  If you have existing programs using VXI*plug&play* that were created using HP VEE version 3.1, you may want to continue to use them with minimal modifications as mentioned earlier in this chapter. However, if you plan to maintain these programs over the long term, it would be better to rewrite them using the standard function panel access in the `To/From VXIplug&play` object as described in "Using the To/From VXIplug&play Object."

- Access to Older Drivers.

  Some earlier versions of non-HP VXI*plug&play* drivers (1995 and earlier) were written to earlier versions of the VXI*plug&play* specification. You can still access these drivers through the HP VEE `Call` object.

Except for the reasons listed above, you should use VXI*plug&play* drivers using the methods described in "Using the To/From VXIplug&play Object."

Using a Dynamic Library in HP VEE

This section will show you the steps in loading a VXI*plug&play* driver into HP VEE once the required files are installed.

To use a VXI*plug&play* driver in a HP VEE program, there are three steps:

1. Import the library.
2. Run the routines which use the library.
3. Delete the library when the program is done.

The three HP VEE objects associated with these steps are `Import Library`, `Call`, and `Delete Library`.

Importing the Library

Before you can use a `Call` object (or `Formula` object) to execute the driver, you must import the function into the HP VEE environment via the `Import Library` object. On the `Import Library` object, under `Library Type`, select `Compiled Function`. Enter the path and name of *PREFIX*`.H` (*PREFIX*`.h` on HP-UX) using the `Definition File` button. See Table 2-2 and Table 2-3 for the location of these files. Finally, select the path and name of *PREFIX*`_32.DLL` (*PREFIX*`.sl` on HP-UX) using the `File Name` button. The `Library Name` button assigns a logical name to a set of functions. It is recommended that the name be *PREFIX*, where *PREFIX* refers to the name of the instrument such as `HPE1410`.

Before using a driver with the `Call` object, you must configure the `Call` object. The easiest way to do this is to select `Load Lib` from the `Import Library` object menu to load the driver file into the HP VEE environment. Bring up a `Call` object from the `Device` menu. Then select `Select Function` on the `Call` object menu. HP VEE will bring up a dialog box with a list of all the functions listed in the header file which are exported from driver file.

Calling a VXI*plug&play* Driver from HP VEE

Use a `Call` object to make the calls to a VXI*plug&play* driver.

Sequence of Calls. The sequence of calls for a VXI*plug&play* driver is very important. The sequence is:

1. Call the initialize function. (This function returns a session handle.)
2. Perform calls to the driver using the handle returned by the initialization function.
3. Call the close function.

Initialize Function. The initialize function *PREFIX*`_init` has three input pins and two output pins. The three input parameters are:

- *Instrument Address*

  Refer to "Advanced Device Configuration: Plug&play Driver" in Chapter 3 for information about VXI*plug&play* addressing.

- *Identification Verification Flag*

  If the verification flag is `1`, the initialize function checks the identity of the instrument. This is to be done by checking the manufacturer ID and model number, using the `"*IDN?"` query, or other means specified by the instrument manufacturer. Set the flag to `0` if the check should not be done.

- *Reset Flag*

  The reset flag should be `1` if the initialize function is to place the instrument in a pre-defined state. Set the flag to `0` if the reset should not be done.

The two output parameters are:

- *Return Value*

  VXI*plug&play* defines the return value from a VXI*plug&play* driver to be the status of the operation performed. The integer returned can be translated into a meaningful message by calling *PREFIX*`_error_query` from a separate `Call` object. If the return value is `0`, the `init()` call was successful.

- *Handle for VXIplug&play Functions*

  If the return value from the initialize function is `0`, then the output parameter contains an instrument handle. An instrument handle is simply a number which associates a function call with this initialization. Most VXI*plug&play* functions require this handle as an input parameter. Each initialization returns a unique handle in the output parameter `vi`. The parameter may be called by a different name, such as `session handle`, but it is always the last parameter returned from the `init()` function. When the `close()` function is called, the handle is returned to the system.

Calling VXI*plug&play* Functions. Other functions can be called using the `Call` object. For each function called, the handle from the *PREFIX*`_init` function must be provided to the `instrID` input pin of the `Call` object.

Using Other Common VXI*plug&play* Functions. Besides the *PREFIX*`_init` and *PREFIX*`_close` functions, there are other common driver functions which VXI*plug&play* drivers may implement. These functions are *PREFIX*`_reset`, *PREFIX*`_self_test`, *PREFIX*`_revision_query`, *PREFIX*`_error_query`, and *PREFIX*`_error_message`.

Using Arrays As Parameters. The VXI*plug&play* specification states that the caller must allocate space for an array or text parameter. Simply stated, this means that HP VEE must allocate the array before passing it as a parameter to the VXI*plug&play* function as shown in Figure 5-8.

Using the Close Function. The close function *PREFIX*`_close` has one input parameter and no output parameters. The input parameter is the handle returned from *PREFIX*`_init`. Executing *PREFIX*`_close` takes the instrument off-line and clears any data associated with the instrument handle.

There may also be some other driver-specific actions related to closing the instrument. The handle cannot be used again by instrument functions. The *PREFIX*`_init` routine must be called again to obtain a new handle.

Deleting the Library

After you are finished using the VXI*plug&play* driver, the `Delete Library` object needs to be invoked for each driver loaded. After the library is unloaded, the library must be loaded again using the `Import Library` object before any functions using that library can be called.

A Simple Example

The follow example is a simple program using a VXI*plug&play* driver in HP VEE. All this program does is import the library, initialize the device, close the device, and delete the library. (Each program thread is started independently with a `Start` button.)

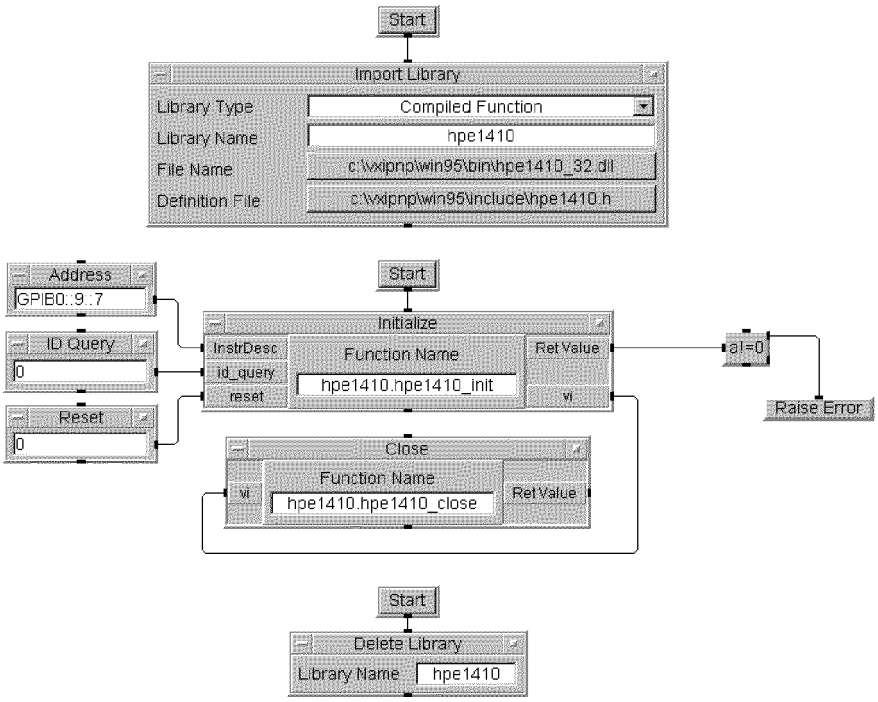

Figure 5-7. A Simple Example

A More Complete Example

The following example shows an HP VEE program that uses a VXI*plug&play* driver and allocates an array to be used as an output parameter.

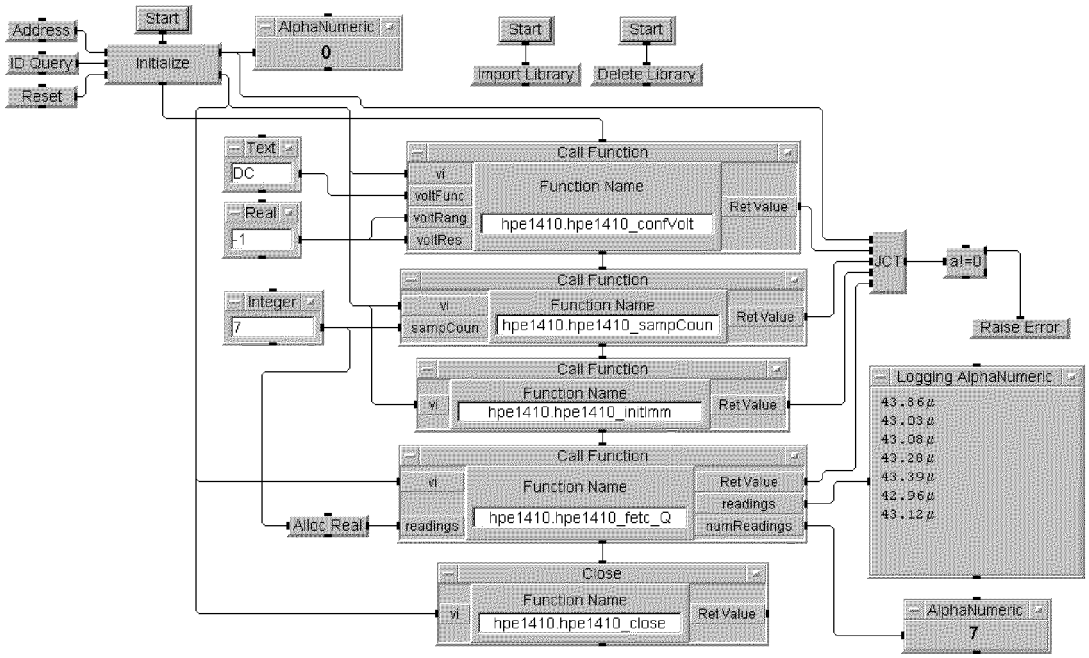

Figure 5-8. A More Complete Example

Some Helpful Hints

Keeping Track of Handles. The handle returned by *PREFIX*`_init` must be used by successive driver functions. There are two ways to accomplish this:

- Connecting Pins

  The value of a handle can be passed by connecting the *PREFIX*`_init` routine data output pin to the `vi` data input pins on each function.

- Keeping Track of Handles Globally

  The handle can be kept as a global variable. The handle from *PREFIX*`_init` routine is connected to a `Set Global` object. Each function that uses this handle, takes it from a `Get Global` object.

Control Flow. The driver needs to perform actions in a certain sequence (initialization, calling functions, and closing). The HP VEE program must be written to ensure that the handle is valid for all functions which require its usage.

Using VXI*plug&play* Functions from Call Objects

6

Using Panel Driver and Component Driver Objects

Using Panel Driver and Component Driver Objects

This chapter provides further information about using `Panel Driver` and `Component Driver` objects with HP VEE. Let's begin with a more detailed look at how these drivers work.

Understanding Panel Driver and Component Driver Objects

This section explains some background and details that will help you use `Panel Driver` and `Component Driver` objects more effectively.

Inside HP Instrument Drivers

The HP VEE `Panel Driver` and `Component Driver` objects both require that the appropriate HP **Instrument Driver** ("ID") be present, and that the instrument be configured to that driver. (These instrument drivers are sometimes called "HP VEE drivers.") The HP Instrument Driver file (the `.cid` file) must be present and configured in order to use `Panel Driver` and `Component Driver` objects. However, these files are not used for `Direct I/O` or VXI*plug&play* operations.

HP Instrument Driver Files

Key Idea

Each **HP Instrument Driver** ("ID") describes the unique personality of a particular test instrument. A driver file is required to control any instrument using a `Panel Driver` or `Component Driver` object.

HP Instrument Driver files (`.cid` files) are copied onto your system disk when HP VEE is installed. Each driver file contains two basic types of information:

1. A description of the instrument's functions and the commands used to set and query them.
2. A description of the appearance and behavior of the graphical control panel visible in the open view of a `Panel Driver` object.

Components

Key Idea

Internally, `Panel Driver` and `Component Driver` objects represent each instrument function as a **component.** Component names are analogous to variable names in programming languages; components are used to hold the value of instrument function settings or measured values.

For example, the HP 3478A voltmeter contains these and other components:

Table 6-1. Typical Voltmeter Driver Components

Component Name	Instrument Function
`ARANGE`	Autoranging is on or off.
`FUNCTION`	The measurement function is voltage, current, or resistance.
`TRIGGER`	The trigger source is internal, external, fast, or single.
`READING`	The most recent measured value.

Components can be accessed interactively or through a program. To access a component interactively, click on a labeled button or display in the open view of a `Panel Driver`. To access components using a graphical program, add them as input or output terminals. For detailed procedures on using components, refer to the sections "Selected Techniques" and "Using Component Driver Objects in a Program".

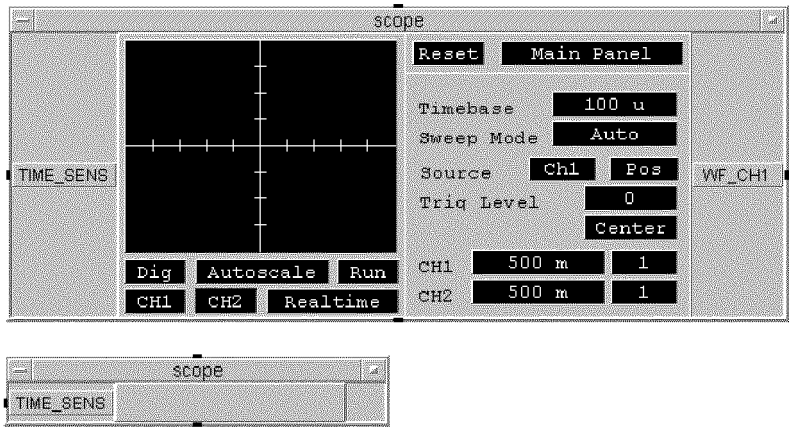

Figure 6-1. Accessing Driver Components

States

An instrument state is a specific set of values for all components in a particular driver. For example, you must set all the components in a voltmeter driver to particular values for AC voltage measurements. You must use a different set of component values to measure DC current. In other words, these two different measurements require two different states.

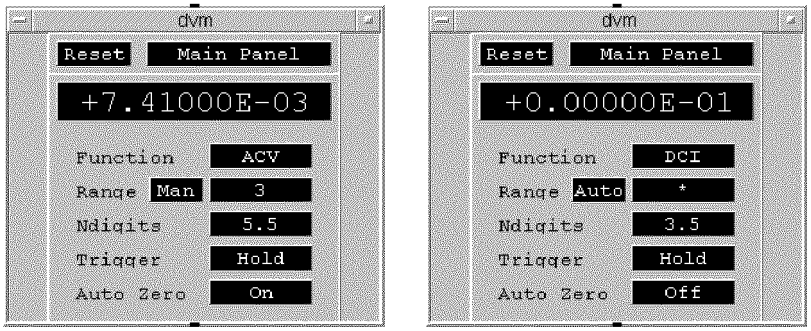

Figure 6-2. Two Voltmeter States

Key Idea

In HP VEE, each instance of a `Panel Driver` represents a separate measurement state. (`Panel Driver` objects are often called "state drivers".) It is common to have more than one `Panel Driver` in a program, where each `Panel Driver` programs the *same* physical instrument to a unique measurement state.

Each `Panel Driver` object you create using the same instrument `Name` will communicate with the same physical instrument.

How HP Instrument Driver-Based I/O Works

When you place a `Panel Driver` or `Component Driver` object in a program, HP VEE establishes a state record in memory. This state record is specific to a particular instrument `Name`. `Name`s are very important and are discussed in greater detail in the section "The Importance of Names" later in this chapter.

All the driver-based objects that reference a particular `Name` share a single state record. The state record reflects the *current* values of each of the instrument's components. When you write to components using `Panel Driver` or `Component Driver` objects, HP VEE updates both the physical instrument and the state record. If you write to the instrument using `Direct I/O`, HP VEE marks the state record as invalid because the state record no longer matches the true state of the physical instrument. However, subsequent use of a `Panel Driver` or `Component Driver` object causes HP VEE to recall the instrument's state, which resynchronizes the physical instrument state and state record.

Important differences occur when the `Panel Driver` and `Component Driver` objects operate.

Panel Driver Operation

> **Key Idea**
>
> When a `Panel Driver` operates, it sends only those commands necessary to make the state of the physical instrument match the state defined in the graphical control panel.

If necessary, a `Panel Driver` will send commands to reset and update all settings in the corresponding physical instrument. This behavior is affected by the `Incremental Mode` setting described in the section, "Advanced Device Configuration: Panel Driver" in Chapter 3.

If you set `Incremental Mode` to `ON`, HP VEE compares the current state record to the desired state defined in the `Panel Driver` and determines which components must be changed. HP VEE sends *only* those commands required to update the affected components.

If you set `Incremental Mode` to `OFF` or if the current state record is marked as invalid, HP VEE will explicitly send commands to update each and every component in order to guarantee synchronization between the desired state and the state of the physical instrument.

Note that a `Panel Driver` operates when its sequence input pin is activated *or* when you click on one of the control panel buttons visible in the open view.

Component Driver Operation

> **Key Idea**
>
> When a `Component Driver` operates, it writes *only* to those components that appear as input terminals and reads *only* from those components that appear as output terminals.

This is why `Component Driver` objects generally operate faster than `Panel Driver` objects. A `Panel Driver` potentially writes to *many* components to achieve a particular state; a `Component Driver` writes to only the components you specify.

Note that components are read and written in the order that they appear as terminals, from top to bottom. This order of operation is important in some cases where you want the instrument to change the value of one component, based on the value of another. This interaction is called **coupling**. With component drivers you must do this manually.

Multiple Driver Objects

This section discusses some situations that may be confusing when you are using multiple objects that:

- Use the same instrument `Name`.
- Use the same instrument address.
- Use the same driver file.

The Importance of Names. This section discusses some concepts related to configuring instruments. You may want to read Chapter 3 first.

Consider how HP VEE maps an instrument object to a specific instrument configuration created with the `Instrument Manager`.

> **Key Idea**
>
> It is the `Name` field in the `Device Configuration` dialog box that logically maps each instrument object to the address of a physical instrument and the other configuration information. To determine the `Name` of an instrument object, click on `Show Config` in the object menu; the text in the object title is *not* necessarily the same as the `Name`.

For example, the `Name`s of the instruments in the default I/O configuration are `dmm`, `dvm`, `fgen`, `funcgen`, `oscope`, and `scope`. `Name`s must be unique. There cannot be more than one configured instrument with the `Name` of `scope`.

In general, you should have only one configured `Name` referencing a particular physical instrument. While it is possible to have more than one `Name` referencing the same instrument address, it will cause unpredictable results in a program using `Panel Driver` objects. HP VEE's internal records of instrument states are organized by `Names`. Two `Panel Driver` objects with different names will blindly write to the same address, thus invalidating each other's state records.

In some cases involving `Direct I/O`, you may need to have more than one `Name` for the same physical instrument. This may be necessary if certain settings in the `Direct I/O` tab of the `Advanced Device Configuration` dialog box need to be varied depending on the direct I/O operation. For example, you may wish to send some commands to an oscilloscope with EOI asserted on the last character of data and some commands without EOI. In such a case, you can configure one instrument with the `Name Scope (EOI)` and another instrument with the `Name Scope`. Both `Scope` and `Scope (EOI)` have the same `Address` setting, but different settings for `END on EOL`.

Note that the configured `Name` appears as the default title in instrument objects at the time you select them from the menu. However, editing the title *in no way* affects the relationship to the `Name`.

`Names` are also important for saving and opening programs containing instruments. When you save a program, the `Name` of each instrument object in the program is saved. When you open a program, HP VEE looks in the current I/O configuration for the `Name` of each instrument being loaded. For example, if you saved a program containing an `Direct I/O` object with a name of `My Scope`, there must be an instrument named `My Scope` in the current I/O configuration. `Name`s must match *exactly*, including any spaces. However, `Name` is not case-sensitive. Furthermore, if the object under consideration is a `Panel Driver` or `Component Driver`, the `ID Filename` (driver file) in the current I/O configuration must match the one used in the saved program.

Reusing Driver Files. It is valid (and not uncommon) to have several objects with different names that use the same driver file. For example, you might have a test system that uses three programmable power supplies named `Supply1`, `Supply2`, and `Supply3` at three separate addresses that all use the `hp665x.cid` driver file. Since the `Names` are different, HP VEE maintains a separate state record for each name; a `Panel Driver` for `Supply1` will have no effect on anything related to `Supply2` or `Supply3`.

Selected Techniques

This section describes some techniques for using `Panel Driver` and `Component Driver` objects.

Using Panel Driver Objects Interactively

The open view of a `Panel Driver` object provides a graphical control panel that you can use to interactively construct a measurement state. If you connect the corresponding physical instrument to your computer and turn `Live Mode` on, you can control the physical instrument interactively as you build the measurement state. To change an individual setting, click on the corresponding field in the graphical control panel and complete the resulting dialog box. To make a measurement and view the result, click on the display region of a numeric or XY display. Note that XY displays may take a few seconds to update.

Using Panel Driver Objects Programmatically

To add a `Panel Driver` object to your program:

1. Click on `I/O` ⟹ `Instrument Manager ...` . The `Instrument Manager` dialog box appears.
2. Click on the desired instrument to highlight it, and then click on the `Panel Driver` button.

NOTE

The `Panel Driver` button will be inactive ("grayed out") if the instrument has not been configured with an HP Instrument Driver file. Refer to Chapter 3 for configuration procedures.

3. When the object outline appears, position the cursor and click once to place the object in the work area.

To use `Panel Driver` objects in a program, you will often use input or output terminals to set the values of components. Each input or output terminal actually corresponds to a component in the driver. There are two ways to add a terminal:

- Select `Add Terminal` ⟹ `Data Input` or `Add Terminal` ⟹ `Data Output` from the `Panel Driver` object menu. A list box appears that lists all the valid driver components not yet used as terminals. Double-click on the component in the list that you wish to add as a terminal.
- Select `Add Terminal by Component` ⟹ `Select Input Component` or `Add Terminal by Component` ⟹ `Select Output Component` from the `Panel Driver` object menu. After making this selection, click on one of the fields or display areas in the graphical control panel to add the corresponding component as a terminal.

In general, it is more convenient to use the first method listed above because you do not need to guess the name of the component you wish to use. However, some components are not visible on any part of the graphical control panel. You must access these using the second method.

Using Component Driver Objects in a Program

To add a `Component Driver` object to a program:

1. Click on `I/O` ⟹ `Instrument Manager . . .` . A list of configured instruments appears.
2. Click on the desired instrument to highlight it, and then click on the `Component Driver` button.

> **NOTE**
>
> The `Component Driver` button will be inactive ("grayed out") if the instrument has not been configured with an HP Instrument Driver file. Refer to Chapter 3 for configuration procedures.

3. When the object outline appears, position the pointer and click once to place the object in the work area.

`Component Driver` objects are generally used when you need to repeatedly execute an instrument control object while changing only a few components. `Component Driver` objects are preferred over `Panel Driver` objects in these situations because `Component Driver` objects write and read only the components you specify and, as result, they execute somewhat faster.

Figure 6-3 illustrates this type of situation. This program measures the frequency response of a filter by sweeping the input frequency sourced by `fgen` and measuring the response using `dvm`. Since the subthread attached to `For Log Range` executes repeatedly, component drivers are used to improve execution speed. Note that `Panel Driver` objects are still appropriate for the initial set up of `fgen` and `dvm`.

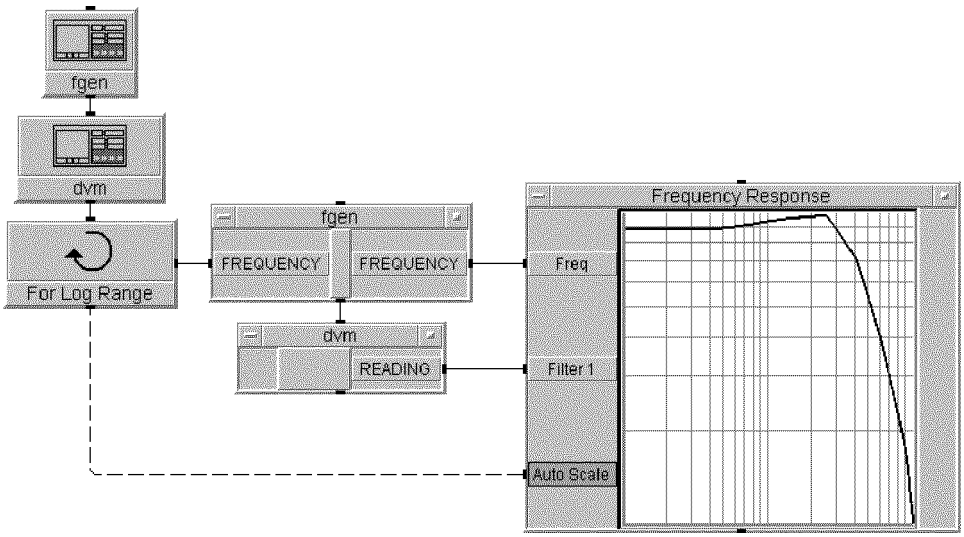

Figure 6-3. Using Panel Drivers and Component Drivers

The program shown in Figure 6-3 is stored in the file `manual15.vee` in your "manual examples" directory.

Getting Help on an HP Instrument Driver

To obtain help on an HP Instrument Driver:

- Select `Help` from the object menu of a `Panel Driver` or `Component Driver` object.
- Select `Help` ⟹ `Instrument` from the main menu, and then select the appropriate ID file name.

In either case, you can open the appropriate help topic from the resulting dialog box.

7

Advanced Topics

Advanced Topics

This chapter covers some advanced instrument I/O topics.

I/O Configuration Techniques

Let's begin by looking at some additional topics regarding the configuration of instruments with HP VEE.

The I/O Configuration File

This section discusses a special file that you may occasionally need to modify. This file (`VEE.IO` on a PC or `.veeio` on a UNIX system) is called the *I/O configuration file*. It is stored in the HP VEE installation directory (`C:\Program Files\Hewlett-Packard\VEE 4.0` by default) on a PC, or your `$HOME` directory on a UNIX system. You may want to ask your system administrator for help with making changes to this file.

When you configure instruments using the `Instrument Manager` (refer to Chapter 3) and you click on the `Save Config` button, the new settings are saved. The settings are saved not only in memory for the remainder of your work session, but also in the `VEE.IO` or `.veeio` file. That way, the next time you start HP VEE you can continue working with the same I/O configuration.

If you do not have a `VEE.IO` or `.veeio` file in your installation or `$HOME` directory when you run HP VEE, HP VEE creates a default `VEE.IO` or `.veeio` file for you. This default file is also created when you run HP VEE for the first time. The default configuration contains the instruments used in the examples included with HP VEE.

You cannot open any program containing an instrument control object unless your I/O configuration contains a device with a matching `Name`. In this discussion, `Name` means the entry in the `Name` field in the `Device Configuration` dialog box, not the text in the object's title bar. Furthermore, if the object is a `Panel Driver` or `Component Driver`, the `ID Filename` must also match your configuration. Settings other than `Name` and `ID Filename` do not affect your ability to *open* these programs, although other settings may affect how the programs *run*.

Most of the time, HP VEE takes care of the `VEE.IO` or `.veeio` file for you. But there may be times when you want to erase, update, or copy this file outside of the HP VEE environment. The rest of this section describes two situations for which you might want to do this.

Sharing Programs

Assume Susan develops an instrument control program that she wants to share with you. How can you get the same I/O configuration as Susan so you can run her program? You can either manually add all of Susan's instruments to your configuration using the `Instrument Manager` and configuration dialog boxes, or you can copy Susan's `VEE.IO` or `.veeio` file to your installation directory (for a PC) or `$HOME` directory (for UNIX). If you use the file copying method, save a copy of your original `VEE.IO` file to another name (such as `VEEIO.OLD`) in case you need it later. For UNIX systems, make sure that any `.veeio` file you place in your `$HOME` directory has write permissions set to allow HP VEE to write to it.

Running Example Programs

Assume that you want to open one of the example programs. Unfortunately, you have accidentally deleted the default instrument configuration. There are two ways to solve this problem:

1. Manually add the default instrument configuration to your current configuration using the `Instrument Manager` and the configuration dialog boxes.
2. Rename your `VEE.IO` or `.veeio` file and restart HP VEE. Then, simply load the program. If HP VEE finds any conflicts it will ask if you wish to add the device now. If you answer `yes`, a `Device Configuration` dialog box will appear with the correct name in the name field. Fill in any addition information needed and press `OK`. HP VEE will then continue loading the file.

If you choose method 1, configure the following instruments using the procedures outlined in Chapter 3:

Table 7-1. Default I/O Configuration

Name Field Entry	ID Filename Field Entry
`dmm`	`hp34401a.cid`
`dvm`	`hp3478a.cid`
`fgen`	`hp3325b.cid`
`funcgen`	`hp33120a.cid`
`oscope`	`hp54600.cid`
`scope`	`hp54504a.cid`

If you choose method 2, follow these procedures:

On PC systems:

1. Exit HP VEE.
2. Go to your installation directory.
3. Type `rename vee.io veeio.old` Enter. This renames your `VEE.IO` file.
4. Run HP VEE. It will look for `VEE.IO` and when it finds that it does not exist, it will create one for you using the default I/O configuration.

On UNIX systems:

1. Exit HP VEE.
2. Go to your `$HOME` directory (typically `/users/`*YourName*).
3. Type `mv .veeio .oldveeio` Return. This renames your `.veeio` file.
4. Execute `veetest`. HP VEE will look for `.veeio` and when it finds that it does not exist, it will create one for you using the default I/O configuration.

You can also run HP VEE with a different I/O configuration file by using the `-veeio` *configFile* command line option.

Programmatic I/O Configuration

You can configure device I/O programmatically. Control pins are available for the `Panel Driver`, `Component Driver`, and `Direct I/O` instrument control objects that let you input other values for device address and timeout. Control pins for setting timeout values are also available for the `Interface Operations`, `Device Event`, and `Interface Event` objects. When a new timeout or address pings one of the control pins, the new value is changed globally for that device. This means that *all* of the instrument control objects communicating with a particular device would begin using the new timeout or address value. The new value can be different than that entered in the Device Configuration dialog box and placed in the HP VEE configuration file. However, this new value is *never* written to the HP VEE configuration file.

The following example shows a `Direct I/O` object with an `Address` control pin. The HPE 1413B is originally configured for address 16032 as shown in the title bar. The input to the control pin is 16040, the new address. When the control pin is pinged that new address, 16040, is put in place for any other objects communicating with the HPE 1413B. The `Direct I/O` object's title bar will change to reflect the new address, then communicate with the device to perform any transactions it contains.

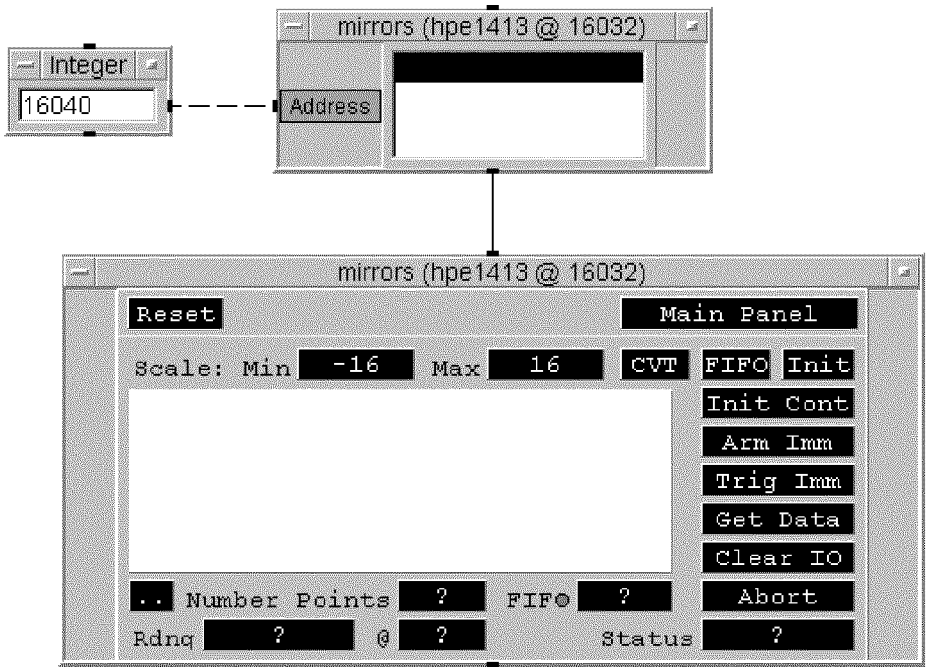

Figure 7-1. Programmatically Reconfiguring Device I/O

LAN Gateways

HP VEE can access LAN gateways to control instruments. A LAN gateway is a controller that allows access to its VXI, HP-IB, GPIO, and Serial interfaces and the instruments on these interfaces from a remote process.

The client-server model best represents the arrangement. An HP VEE process acts as the client when accessing a LAN gateway on a remote computer, the server. The server computer has a committed process, known as a daemon, which is part of the SICL process running on the server. The daemon communicates with the HP VEE client and allows access to its interfaces and their devices. The client process calls SICL in order to control devices on the interfaces which SICL supports. These interfaces are usually configured on the LAN gateway on which the SICL process is running. By using the LAN gateway, these interfaces can be on a remote computer. As far as the client is concerned, the fact that the interfaces and their devices are attached physically to a remote computer is invisible.

Configuration

You must complete configuration tasks in HP VEE and for the LAN hardware to use the LAN gateway.

HP VEE Configuration. Configuring HP VEE for gateway access is done during device configuration, described in Chapter 3. The following figure shows the `Device Configuration` dialog box. The `Gateway` field shows its default setting, `This host`:

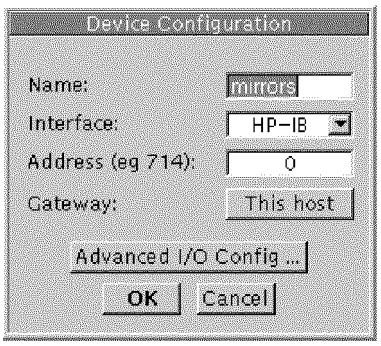

Figure 7-2. Gateway Configuration

You can select the gateway name by clicking on the `Gateway` field. A list box appears showing all of the gateways that have been configured previously. `This host` always points to the computer on which HP VEE is running. If there are no other choices for gateways, you may type in a name for a gateway. The name must be resolvable to an IP address either by a symbolic host name table or by a name-server. Alternatively, an IP address in dot-format may be entered as a name, such as `55.55.55.555`.

Beyond selecting a gateway, the configuration process remains the same. `Panel Driver` and `Direct I/O` objects are configured as before. The following figure shows various I/O devices configured for interfaces and devices on remote computers.

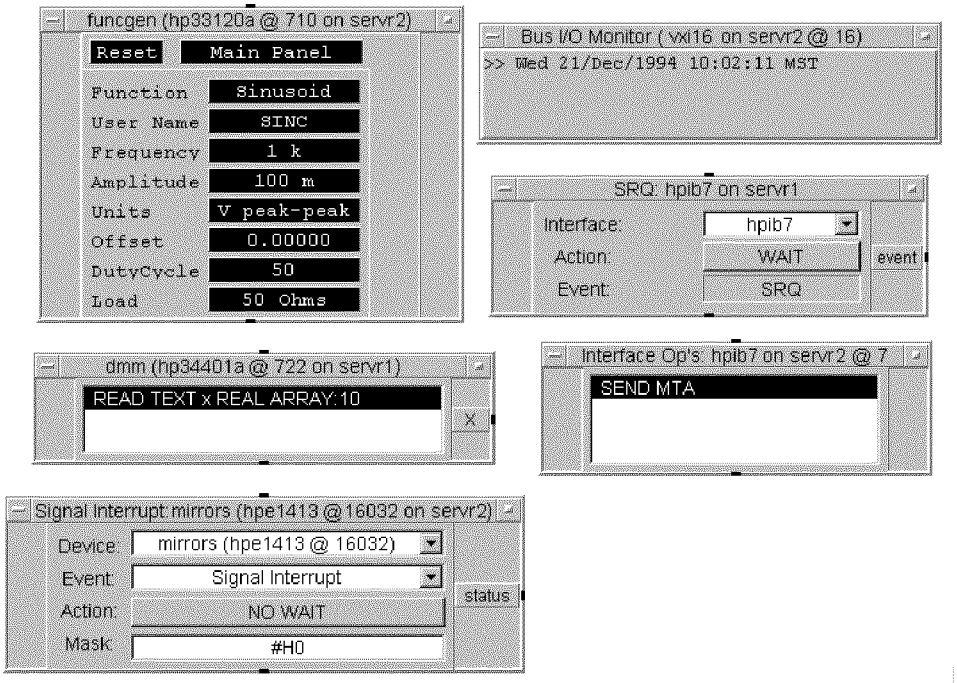

Figure 7-3. Examples of Devices Configured on Remote Machines

LAN Hardware Configuration. The SICL LAN gateway support is dependent on the configuration of the machine on which HP VEE is running, the machine on which the gateway daemon is running, and the overall configuration of the LAN. You should consult with your system administrator

to configure the LAN and ensure that names and IP addresses are resolvable. For the machine running the gateway daemon it is assumed that the daemon install procedures will configure the local networking files correctly. If you are using the HP E2050A LAN/HP-IB Gateway, it is self-contained and all internal configuration is done.

For networks using the HP-UX operating system, the client machine does not need any special network configuration files. However, the following line must be in the SICL configuration file `hwconfig.cf`.

```
#
# LAN Configuration
#     <lu> <symname> ilan <not used> <not used> <sicl_infinity> <lan_timeout_delta>
30 lan ilan 0 0 120 25
```

This entry contains the normal logical unit/symbolic name keys for SICL. The interface type is `ilan`. The `sicl_infinity` and `lan_timeout_delta` entries are special timeouts and will be discussed in the next section.

For the server machines, entries need to be made in two files, `/etc/rpc` and `/etc/inetd.conf`.

To `/etc/rpc` add the following line:

```
siclland      395180
```

To `/etc/inetd.conf` add one of the the following lines. For HP-UX 9.x use:

```
rpc stream tcp nowait root /usr/etc/siclland 395180 1 siclland -e -l /tmp/siclland.log
```

or, for HP-UX 10.x use:

```
rpc stream tcp nowait root /opt/sicl/bin/siclland 395180 1 siclland -e -l /tmp/siclland.log
```

On the server machine, the inet daemon must be made to reread the `inetd.conf` file by executing the following command with sys-admin (root) privileges:

```
/etc/inetd -c
```

If the LAN resource discovery is not managed by the local files but by Network Information Services (NIS, see Yellow Pages), then the same files must be modified on the database machine and the database recompiled.

Execution Behavior

Ideally, I/O operations through the gateway work as though the interfaces and devices are attached directly to the client computer. However, the

non-deterministic nature of the LAN can cause response times to vary. Response times vary depending on the LAN configuration including the number of connected hosts, LAN-to-LAN gateways, and current load. Sometimes, a connection is terminated by disconnected cables or computer failures on the LAN. These events must be accommodated by configuring timeout periods.

When the server receives an I/O request from the client application, HP VEE, the server uses the timeout value that you enter in the `Device Configuration` dialog box. This is called the SICL timeout. If the server's operation is not completed in the specified time, then the server will send a reply to the client indicating that a timeout occurred, and the normal HP VEE timeout error will occur.

When the client sends an I/O request to the server, the client starts a timer and waits for the reply from the server. If the server does not reply in time, a timeout occurs and an HP VEE timeout error is produced. This is called the LAN timeout. The client timeout differs from the server timeout because the I/O transaction time for the server is usually different than the transmission time over the LAN. Specifically, the server may complete an I/O transaction within five seconds (the HP VEE default timeout period), but the actual transmission over the LAN back to the client may take longer than five seconds due to LAN operating characteristics.

The two timeouts are separate values that are adjusted using two entries in the SICL configuration file:

`sicl_infinity`	Used by the server if the user-defined timeout (the SICL timeout), entered in the `Advanced Device Configuration` dialog box, is infinity (0). The server does not allow an infinite timeout period. The value specifies the number of seconds to wait for a transaction to complete within the server.
`lan_timeout_delta`	Value added to the server's timeout value to determine the client's timeout period (LAN timeout). The calculated LAN timeout only increases as necessary to meet the needs of the I/O devices, and never decreases. This avoids the overhead of readjusting the LAN timeout every time the SICL timeout changes.

Protecting Critical Sections

In a multi-process test system, sharing a resource, such as an instrument, among the processes requires that a locking mechanism be available to protect critical sections. A critical section is needed when one of the processes needs exclusive access to a shared instrument resource. To prevent another process from accessing the instrument during the critical section, the first process locks the instrument. The lock remains in effect for the time necessary to complete its task. During this time, the second process is unable to execute any interaction with the instrument including any attempt to lock the instrument for its own use.

The following EXECUTE transactions let you protect critical sections and can be used in the `Direct I/O`, `MultiDevice Direct I/O`, and `Interface Operations` transaction objects. Notice that the transaction syntax varies depending on the interface and transaction object being used. For HP-IB, Serial, and GPIO, the entire interface is locked. For VXI, individual devices are locked.

To lock VXI devices via direct backplane access in the `Direct I/O` object, use the transactions

```
EXECUTE LOCK DEVICE
EXECUTE UNLOCK DEVICE
```

In the `MultiDevice Direct I/O` object, use the transactions

```
EXECUTE vxiScope LOCK DEVICE
EXECUTE vxiScope UNLOCK DEVICE
```

where `vxiScope` is the configured name of a VXI oscilloscope such as the HPE 1428B.

To lock HP-IB, Serial, and GPIO Interfaces in the `Interface Operations` object, use the transactions

```
EXECUTE LOCK INTERFACE
EXECUTE UNLOCK INTERFACE
```

Supported Platforms

Table 7-2. EXECUTE LOCK/UNLOCK Support

Platform	Supported I/O Interfaces
Windows 95 (PC, HP 6232, HP 6233, or EPC7/8)	• HP-IB[1] • Serial • VXI (PC with VXLink, or embedded)[2]
Windows NT (PC, HP 6232, HP 6233, or EPC7/8)	• HP-IB[1] • Serial • VXI (PC with VXLink, or embedded)[2]
HP-UX (HP 9000 Series 700 or V/743)	• HP-IB • Serial • GPIO • VXI (S700 with MXI, VXLink, or embedded)[2]

1 The National Instruments GPIB interface does not support LOCK.

2 Register and memory access of VXI devices (READ/WRITE REGISTER/MEMORY transactions) are not lockable. Only the very first execution of a transaction that attempts a direct memory access could be locked out if the memory is mapped into the HP VEE process space) by a prior lock in another process. After that there is no way to prevent multiple processes from simultaneously accessing a memory location since this is shared memory.

Execution Behavior

When a version of the EXECUTE LOCK transaction executes, an attempt is made to acquire a lock on the device or interface. If there is no pre-existing lock owned by another process then the transaction executes completely and the lock acquisition succeeds. If, however, a prior lock exists, the transaction will block for the current timeout configured for that device or interface. If the other process gives up the lock within the timeout period the transaction completes and acquires the lock. If the timeout period lapses, an error occurs and an error message box appears. This error can be captured by an error pin on the transaction object.

After the lock has been acquired, all subsequent I/O from `Direct I/O`, `MultiDevice Direct I/O`, `Panel Driver`, `Component Driver`, and `Interface Operations` objects will be protected from any other process attempting to communicate to that device or interface. After the critical section has passed, the corresponding version of the EXECUTE UNLOCK transaction can be executed.

Locks only protect critical sections across process boundaries. A single process can create nested locks by performing two EXECUTE LOCK transactions in sequence. Both transactions will succeed as long as there

are no prior locks by another process. The process must then perform two EXECUTE UNLOCK transactions. If only one EXECUTE UNLOCK transaction is executed the device or interface remains locked. If a transaction attempts an unlock without a prior lock, a run-time error occurs.

Locks only exist while the HP VEE program is executing. When an HP VEE program finishes executing, all locks are removed from devices and interfaces. This protects the user from leaving devices or interfaces locked if the program stops executing due to normal completion, run-time errors, or a pressed `Stop` button, and no EXECUTE UNLOCK transaction has executed.

Example

The following program example shows the EXECUTE LOCK/UNLOCK INTERFACE transactions in an `Interface Operations` object configured for HP-IB. This example would be identical for a serial interface, too. The lock and unlock transactions frame the UserObjects performing I/O to the devices on the HP-IB interface at select code 7. This program will attempt to acquire the lock three times. If the lock cannot be acquired after three attempts, a user-defined error occurs.

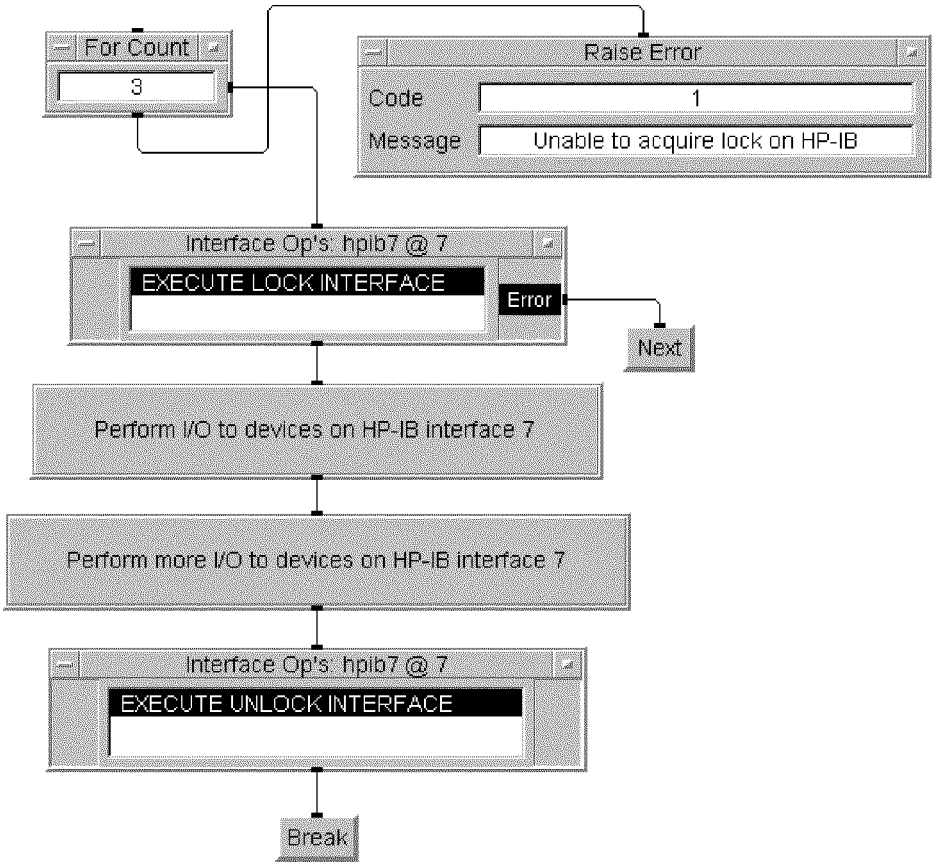

Figure 7-4. EXECUTE LOCK/UNLOCK Transactions—HP-IB

For each attempt, the EXECUTE LOCK INTERFACE transaction tries to acquire the lock in the time allowed by the configured timeout period. You can set the timeout period in the `Properties` dialog box of the `Interface Operation` object. The error pin attached to the `Next` object in the first transaction object will cause the thread to be re-executed in another attempt. The break object after the last transaction object ensures that the thread does not get executed, unnecessarily, a second time.

The following example shows the EXECUTE LOCK/UNLOCK DEVICE transactions in a `MultiDevice Direct I/O` object. You could use the `Direct I/O` object, instead of the `MultiDevice Direct I/O`, but that would

mean using an object for each device instead of one object for the group of devices. This is very similar to the program in the previous figure. A `For Count` object drives a thread which tries to acquire locks on three different devices. After the I/O activity is done in the user objects, a series of unlocks are executed.

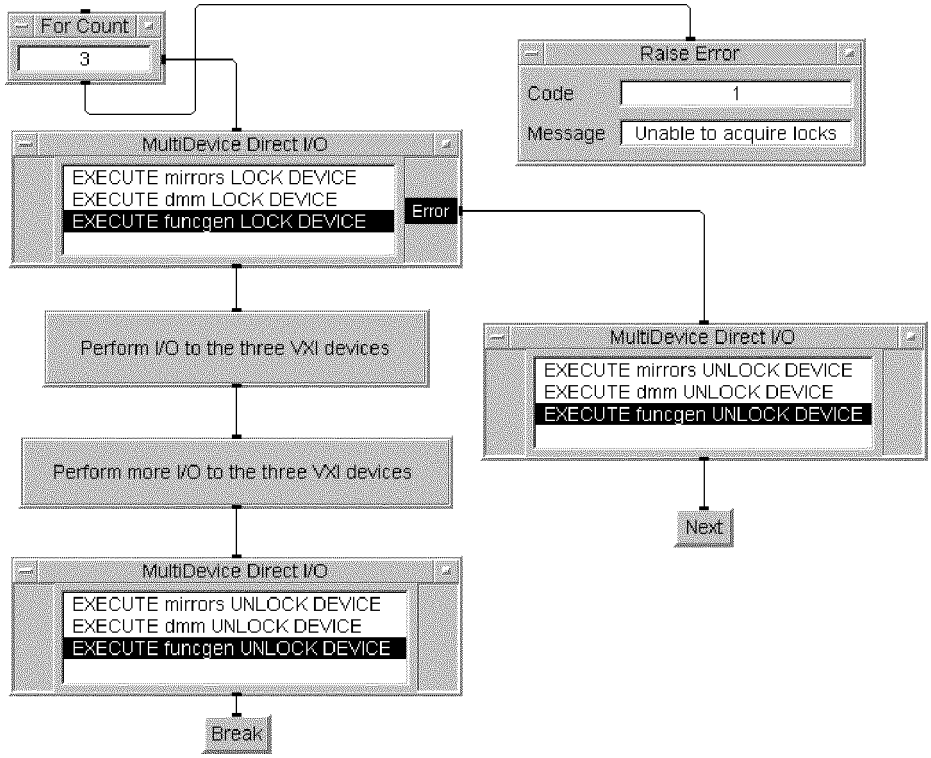

Figure 7-5. EXECUTE LOCK/UNLOCK Transactions—VXI

Each transaction tries to acquire its respective lock for the timeout period configured for each device. If any of the three transactions timeout an error occurs which is trapped by the error pin. If a successful lock is followed by an attempt resulting in a timeout error, the error pin traps the error. However, before the program can re-execute the lock transactions, all acquired locks must be unlocked. That is the reason for the `MultiDevice Direct I/O` object attached to the error pin. It is very important that this object try to unlock each device *in the same order* as the first object acquired the locks. Since an error occurs if an unlock transaction is executed before the lock transaction, an error pin is also added to the object with the unlock transactions. If a transaction fails to acquire the lock in the first object then the same unlock transaction fails in the following object.

I/O Control Techniques

This section describes some additional techniques for instrument I/O control.

Polling

HP VEE supports all the serial poll operations defined by IEEE 488.1. All HP-IB instruments, and all VXI message-based instruments, support serial poll operations. VXI message-based devices are, by definition, IEEE 488.2 compliant. VXI register-based devices are IEEE 488.2 compliant if an I-SCPI driver is available. HP VEE does not support parallel poll operations.

You can obtain an instrument's serial poll response in two ways:

Object	Serial Poll Behavior
`Device Event`	The `Device Event` object can poll the specified instrument once and output a scalar integer, which is the serial poll response using the `NO WAIT` option. The `Device Event` object can also wait for a specific bit pattern within the serial poll response byte by using a user supplied bit mask and the `ALL CLEAR` and `ANY SET` options.
`Direct I/O`	`Direct I/O` objects for HP-IB instruments support a `WAIT SPOLL` transaction. This transaction repeatedly polls an instrument until the serial poll response byte matches a specific bit pattern, using a user-supplied bit mask and the `ALL CLEAR` or `ANY SET` options. Refer to Chapter 4 for additional information about `Direct I/O`.

The `Device Event` object has special execution properties when configured for `Spoll` that are discussed in the next section, "Service Requests." This behavior allows for other concurrent threads to continue execution while waiting for a specific bit pattern using the mask value and the `ALL CLEAR` or `ANY SET` options. `NO WAIT` will simply execute immediately and return the status byte of the HP-IB or message-based VXI instrument. Both objects have a `Timeout` control input available from their object menus (`Add Terminal`) so you can programmatically set a timeout period.

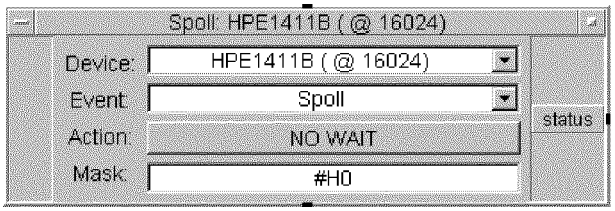

Figure 7-6. Device Event Configured for Serial Polling

Service Requests

To detect a service request (SRQ message) for a VXI instrument, use the `Device Event` object (`I/O` ⟹ `Advanced I/O` ⟹ `Device Event`). To detect a service request for an HP-IB instrument or RS-232, use the `Interface Event` object (`I/O` ⟹ `Advanced I/O` ⟹ `Interface Event`).

The `Device Event` and `Interface Event` objects provide special behavior for interrupt-like execution. To view this behavior, you may wish to run your program with `Debug` ⟹ `Show Execution Flow` enabled.

For example, `Interface Event` behaves in a program as follows:

1. Before an `Interface Event` object (configured for HP-IB and with the `WAIT` option specified) operates, execution proceeds normally with each thread sharing execution with equal priority.
2. When a `Interface Event` object operates, execution of the thread attached to the `Interface Event` data output pauses at the `Interface Event` object. Other threads not attached to `Interface Event` *will continue to execute*.
3. When an SRQ is detected on the specified interface, the data output of `Interface Event` is activated.

   At this point, *execution o fall other threads is blocked* until the thread attached to the data output of `Interface Event` completes execution.

The program shown in Figure 7-7 shows how to handle service requests. In the case shown, it is possible that either `dvm` or `scope` is responsible for a service request. The program determines the originator of the service request by using `Device Event` to obtain the status byte of each instrument. Each status byte is tested using `If/Then/Else` and the `bit(x,n)` function to determine if bit 6 is true. If bit 6 is set, then the corresponding instrument is responsible for the service request. The `Until Break` object automatically re-enables the entire thread to handle any subsequent service requests. The `Device Event` object is configured for `NO WAIT`, meaning the status byte is returned without using the mask value. If a mask value of `64` is used and the `Device Event` object is configured for `ANY SET`, the `If/Then/Else` and `bit(x,n)` function need not be used.

Note that different instruments have different requirements for clearing and re-enabling service requests. In Figure 7-7, `dvm` requires only a serial poll to clear and re-enable its SRQ capability. However, `scope` requires the additional step of clearing the originating event register.

The `Device Event` object can be used to detect a service request from a message-based VXI instrument. The instrument that writes a request true event (RT), which is evaluated as a request for service, into the VXI controller's signal register will receive a *Read STB* word serial protocol command. The message-based instrument will send its status byte back to the controller, and will write a request false event (RF) into the VXI controller's signal register. The status byte will be used with the supplied mask value and the `ANY SET` or `ALL CLEAR` options to determine which bit (besides bit 6) is set. Thus one object, the `Device Event` can be used to detect a service request from a message-based VXI device and determine why the request occurred.

Both objects have a `Timeout` control input available from their object menus (`Add Terminal`) so you can programmatically set a timeout period. For further information see the `Device Event` and `Interface Event` reference sections in the HP VEE on-line help.

NOTE

The program shown in Figure 7-7 will run only if the specified instruments are connected, configured, and powered up. However, you can use this program as an example of programming techniques to use in your own programs, or you can modify the program to communicate with your own instruments.

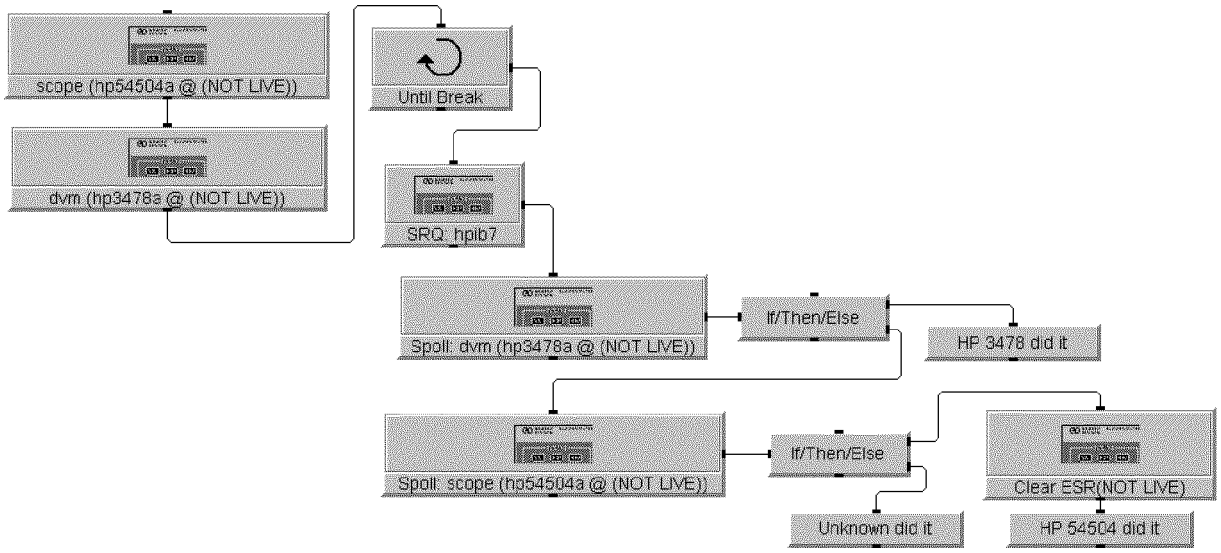

Figure 7-7. Handling Service Requests

This program is saved in the file `manual16.vee` in your examples directory.

Monitoring Bus Activity

You can use the `Bus I/O Monitor` object (`I/O` ⟹ `Bus I/O Monitor`) to record all bus messages transmitted between HP VEE and any talkers and listeners. Note that `Bus I/O Monitor` records *only* those bus messages inbound or outbound from HP VEE.

You can monitor any supported interface (HP-IB, VXI, serial, or GPIO) using a `Bus I/O Monitor`. Each instance of a `Bus I/O Monitor` object monitors just one hardware interface.

Figure 7-8 shows the bus messages sent to write `*RST` to an instrument at HP-IB address 717.

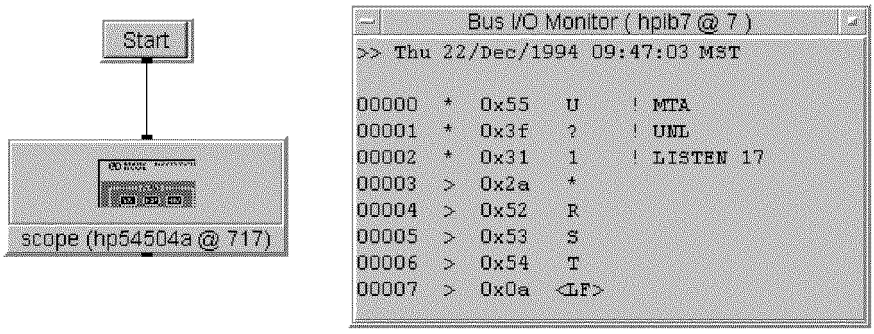

Figure 7-8. The Bus I/O Monitor

The display area of `Bus I/O Monitor` contains five columns:

- Column 1 - Line number
- Column 2 - Bus command (`*`), or outbound data (`>`), or inbound data (`<`)
- Column 3 - Hexadecimal value of the byte transmitted
- Column 4 - 7-bit ASCII character corresponding to the byte transmitted
- Column 5 - Bus command mnemonic (bus commands only, blank for data)

Note that the `Bus I/O Monitor` executes much faster as an icon than as an open view object.

Low-Level Bus Control

You can send low-level bus messages in two ways:

Object	Bus Message Capability
`Interface Operations`	This object allows you to send arbitrary bus messages to any HP-IB device, or reset the VXI interface and fire various VXI backplane trigger lines.
`Direct I/O`	`Direct I/O` objects for HP-IB, message-based VXI instruments, and I-SCPI supported register-based VXI instruments lets you send `CLEAR`, `LOCAL`, `REMOTE`, and `TRIGGER` commands using `EXECUTE` transactions.

For further information regarding `Interface Operations` and `Direct I/O`, please refer to Chapter 4.

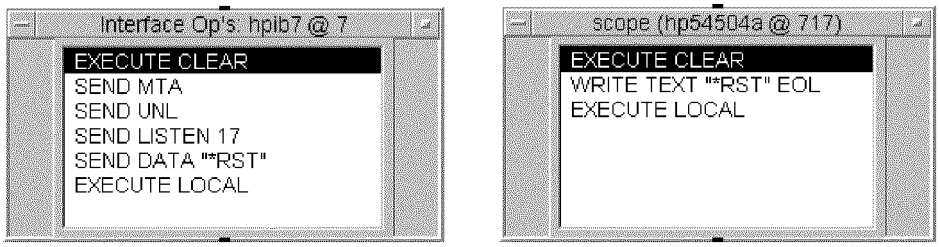

Figure 7-9. Two Methods of Low-Level HP-IB Control

Instrument Downloading

Some instruments allow you to download macros, measurement routines, or complete measurement programs. For example, some HP instruments support HP Instrument BASIC—you can write complete HP Instrument BASIC programs that execute inside the instrument. Here is one approach for using HP VEE to download a measurement routine to an instrument:

1. Create and maintain your measurement routine using a text editor, such as `vi`. Save the measurement routine in an ordinary text file.
2. Use `From File` to read the file.
3. Use `Direct I/O` to write the contents of the file to the instrument.

This section presents a complete example of downloading using this approach. Please refer to Chapter 4 for further information regarding `Direct I/O`.

Figure 7-10 shows a program that downloads a measurement subprogram to the HP 3852A. This example downloads a simple subprogram, `BEEP2`, that beeps twice and displays a message.

NOTE

The program shown in Figure 7-10 will run only if the specified instruments are connected, configured, and powered up. However, you can use this program as an example of programming techniques to use in your own programs, or you can modify the program to communicate with your own instruments.

Since the HP 3852A is not included in the default I/O configuration, you must follow these steps to open the example program:

1. Use `I/O` ⟹ `Instrument Manager ...` to add a device with the settings listed here. Enter these settings in the `Device Configuration` and `Advanced Device Configuration` dialog boxes *exactly* as shown, including spaces:

   `Name: HP 3852A`

   `Interface: HP-IB`

   `Address:` Enter `0` if you do not have an HP 3852A connected to your computer. If you do have an HP 3852A, enter its address instead; the factory default is `709`.

   `Timeout: 5`

   `Live Mode:` Enter `OFF` if an HP 3852A is *not* connected to your computer or `ON` if an HP 3852A is connected.

   `Byte Ordering: MSB`

2. Click on `OK`, and then click on the `Save Config` button.

Here are the contents of the downloaded file `manual17.dat`:

```
DISP MSG "LOADING BEEP2"
WAIT 1

SUB BEEP2
DISP "BEEP2 CALLED"
BEEP
WAIT .5
BEEP
SUBEND

DISP MSG "BEEP2 LOADED"
```

The `manual17.dat` file is provided in your examples directory.

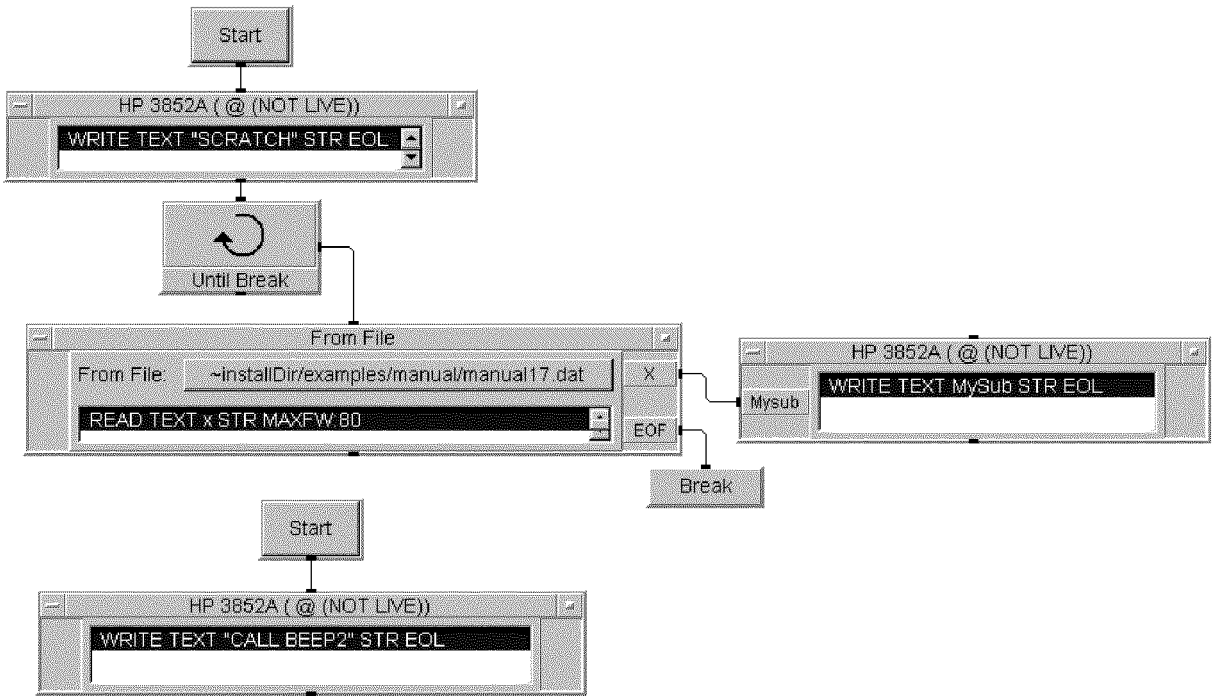

Figure 7-10. Downloading to an Instrument

This program is saved in the file `manual17.vee` in your examples directory.

A

Select Codes and I/O Addressing

Select Codes and I/O Addressing

To access an I/O device, you'll need to determine the correct address and enter it in the `Address` field in the `Device Configuration` dialog box, using the `Instrument Manager` as described in Chapter 3 of this manual. This appendix covers the HP VEE I/O addressing scheme, including interface select codes and instrument addresses, that supports `Direct I/O`, `Panel Driver`, and `Component Driver` I/O operations. Note that this addressing scheme is *not* used for VXI*plug&play* I/O operations. Refer to "Advanced Device Configuration: Plug&play Driver" in Chapter 3 for information about VXI*plug&play* addressing in HP VEE.

NOTE

HP VEE supports the HP-IB, GPIB, RS-232 serial, and GPIO interfaces. (HP-IB is Hewlett-Packard's implementation of the IEEE-488 interface bus standard. Other implementations are commonly called GPIB.) Also, you can access VXI devices by using an HP E1406 Command Module connected to one of the supported HP-IB or GPIB interfaces.

HP VEE supports direct VXI backplane access for embedded VXI controllers including the HP V743 VXI Embedded Controller, the HP 6232 and HP 6233 VXI Pentium Controllers, and the EPC-7 and EPC-8 VXI Controllers. HP VEE also supports direct VXI backplane access for the E1383A and E1483A VXLink interfaces for PCs, and for the HP E1489C EISA/ISA-to-MXIbus interface for HP 9000 Series 700 computers.

The HP VEE addressing scheme uses **select codes**, which you can set up using the `I/O Config` utility program as part of installing and configuring the HP I/O libraries included with HP VEE. Refer to *Installing the HP I/O Libraries - HP VEE* for information about installing and configuring the HP I/O libraries, and setting up select codes using `I/O Config`. It is recommended that you set up the select codes for your interfaces according to the list in Table A-1.

Recommended I/O Select Codes for HP VEE

The following interface select codes are recommended for use with HP VEE. Refer to *Installing the HP I/O Libraries - HP VEE* for information about installing and configuring the HP I/O libraries, and setting up select codes for your interfaces using the `I/O Config` utility program.

Table A-1. Recommended I/O Select Codes

Select Code	PC (Windows 95, NT)	Series 700 (HP-UX)
1	HP-IB (HP 82340 or HP 82341)	HP-IB (HP E2070 or HP E2071)
2	HP-IB (HP 82340 or HP 82341)	HP-IB (HP E2070 or HP E2071)
3	HP-IB (HP 82340 or HP 82341)	HP-IB (HP E2070 or HP E2071)
4	HP-IB (HP 82340 or HP 82341)	HP-IB (HP E2070 or HP E2071)
5	HP-IB (HP 82340 or HP 82341)	HP-IB (HP E2070 or HP E2071)
6	HP-IB (HP 82340 or HP 82341)	HP-IB (HP E2070 or HP E2071)
7	HP-IB (HP 82340 or HP 82341)	HP-IB (HP E2070 or HP E2071)
8	HP-IB (HP 82340 or HP 82341)	HP-IB (HP E2070 or HP E2071)
9	COM1 serial port	COM1 serial port
10	COM2 serial port	COM2 serial port
11	COM3 serial port	COM3 serial port
12	COM4 serial port	COM4 serial port
13	GPIO (HP E2075)	GPIO (HP E2075)
14	GPIB0 (National GPIB card)	Unused
15	GPIB1 (National GPIB card)	Unused
16	VXI (Embedded, or PC using VXLink)	VXI (Embedded, or S700 using EISA/ISA-to-MXIbus)
17	GPIB2 (National GPIB card)	Unused
18	GPIB3 (National GPIB card0	Unused

NOTE

Select code 7 is the recommended default for the *first* HP-IB card. Each card must have a unique select code.

The HP 82335 HP-IB Card is also supported for Windows 95 on the PC (*not* for Windows NT). However, only select codes 3 through 7 are recommended for the HP 82335 HP-IB Card, and the select code is set by the on-card switch settings (the default setting is 7). In addition, you must exclude address space for the HP 82335 as described in "Excluding Address Space for the HP 82335 HP-IB Card (Windows 95 Only)" later in this appendix.

Only select codes 14, 15, 17, and 18 are supported for National GPIB cards on the PC. These GPIB cards are not supported for HP 9000 Series 700 computers.

I/O Addressing

The addressing schemes for various types of devices are described in the following sections.

To Address Serial Ports

Serial ports are supported by using the select codes that you assigned to them using `I/O Config`. Normally, the COM1 serial port is assigned select code 9 (refer to Table A-1). In this case, use `9` as the address of the device connected to COM1.

To Address GPIO Devices

GPIO devices are supported by using the select code that you assigned to the GPIO interface using `I/O Config`. Normally, the select code 13 is used for GPIO. In this case, use `13` as the address for the GPIO device.

To Address HP-IB and GPIB Interfaces and Devices

HP-IB and GPIB devices are addressed using the following scheme:

SPA[*SA*]

Where:

S	is the select code of the HP-IB or GPIB interface.
PA	is the primary address of an HP-IB or GPIB device (the valid range is 00 through 30).
SA	is the optional secondary address (the valid range is 00 through 31).

Let's look at a couple of examples to see how this works:

- For an HP-IB device at select code 7, primary address 01, enter `701` in the `Address` field of the `Device Configuration` dialog box.
- For a GPIB device at select code 14, primary address 09, secondary address 02, enter `140902` in the `Address` field of the `Device Configuration` dialog box.

HP-IB Select Codes

The HP-IB interfaces are supported by using the select codes that you assigned to them using `I/O Config`. The recommended select codes for HP-IB interfaces are as listed in Table A-1. If the recommended select codes (1 through 8) are configured by the I/O libraries for HP-IB interfaces, HP VEE can theoretically access up to eight HP-IB cards, which can be a mix of the supported cards:

- For an HP E2070 or HP E2071 HP-IB Card (for Series 700 computers), the select code is assigned by the software. The select codes are assigned in the order: 7, 8, 1, 2, 3, 4, 5, and 6. However, each card must be set to a unique base address. (Refer to the owner's manual for information on setting the base address.)
- For an HP 82340 or HP 82341 HP-IB Card (for PCs), the select code is assigned by the software. The select codes are assigned in the order: 7, 8, 1, 2, 3, 4, 5, and 6. However, each card must be set to a unique base address. (Refer to the owner's manual for information on setting the base address.)

- For an HP 82335 HP-IB Card (for PCs, Windows 95 only), the select code is determined by switch settings on the card (the default is 7). If you install more than one HP 82335 card, each card must be set for a unique select code in the range 3 through 7. (Refer to the owner's manual for instructions.) Also, you must exclude address space for each card. Refer to "Excluding Address Space for the HP 82335 HP-IB Card (Windows 95 Only)" later in this chapter.

GPIB Select Codes (PCs Only)

The National Instruments GPIB driver configures up to four GPIB cards with the designations GPIB0, GPIB1, GPIB2, and GPIB3. In order to access these GPIB cards, you *must* assign the select codes 14, 15, 17, and 18 to the GPIB cards (as listed in Table A-1) using `I/O Config`. HP VEE does not support any other select codes for GPIB cards. Otherwise, the addressing is the same as for an HP-IB card.

To Address VXI Devices on the HP-IB or GPIB

To access VXI devices through the HP-IB (or GPIB) with an HP-IB command module, you can use secondary addresses. If you are using an HP E1406 Command Module in a VXI card cage, the primary address is set by a switch on the command module (default=09) and the secondary address is the individual VXI device's logical address divided by eight.

For example, suppose you have an HP E1406A Command Module (address=09) in an HP E1401A C-Size High-Power Mainframe connected to the HP-IB at select code 7. If you have an HP E1326B Multimeter in a VXI slot, with its logical address set to 24, you would enter the value `70903` for the address.

Two instrument drivers are provided to help you find the correct addresses for VXI devices connected by means of an HP-IB command module:

- Use the `hpe140x.cid` driver to locate VXI devices connected by means of an HP E1405 or HP E1406 HP-IB Command Module in a C-size VXI card cage.
- Use the `hpe1300a.cid` driver to locate VXI devices connected by means of an HP E1306 HP-IB Command Module in a B-size VXI card cage. (This driver also supports the HP E1300 and HP E1301 B-Size VXI Mainframes, which include built-in command modules.)

To use either of these drivers, add an instrument panel for the driver using the `Instrument Manager` as described in Chapter 3 of this manual.

NOTE

Do not enter a sub address value for VXI devices, except for modules in a VXI switch box. Refer to the next section for details.

To Set Address/Sub Address Values

Most HP-IB, GPIB, and VXI devices do not use sub addresses. Do not enter a sub address value unless you are accessing a VXI switch box, or one of the card cage devices that uses sub addresses (for example, the HP 3235A Switch/Test Unit or the HP 3488A Switch/Control Unit).

NOTE

Sub address values are used only if you are using an HP Instrument Driver for a device that supports sub addresses. Do not use sub address values if you are using Direct I/O.

Let's look at a couple of examples:

- If you are accessing a module in an HP 3235A Switch/Test Unit, enter the HP-IB or GPIB address (for example, `701`) of the HP 3235A itself in the `Address` field of the `Device Configuration` dialog box, using the `Instrument Manager` as described in Chapter 3 of this manual. Enter the sub address of the individual module in the `Sub Address` field of the `Advanced Device Configuration` dialog box (on the `Panel Driver` tab). For information on what to put in the `Sub Address` field, refer to the online help for the HP 3235A instrument driver (`Help` ⟹ `Instruments`).
- If you are accessing a module in a VXI switch box, enter the HP-IB or GPIB address of the switch box (for example, `70902`) in the `Address` field, and the sub address of the individual module in the `Sub Address` field. For information on what to put in the `Sub Address` field, refer to the online help for the VXI switch box instrument driver.

To Address the VXI Backplane Directly

HP VEE can address the VXI backplane directly for the following systems:

- An HP 6232 or HP 6233 VXI Pentium Controller.
- An EPC-7 or EPC-8 VXI Controller, provided the EPConnect software is installed.
- A PC connected to a VXI card cage using an E1383A or E1483A VXLink (ISA-to-VXI) interface, provided the EPConnect software is installed.
- An HP V743 VXI Embedded Controller.
- An HP 9000 Series 700 computer connected to a VXI card cage using an HP E1489C EISA/ISA-to-MXIbus interface.

Assuming that the recommended select codes have been set up using `I/O Config` (refer to Table A-1), HP VEE accesses the VXI backplane via select code 16. The address for a VXI device is simply the select code (16) with the logical address of the VXI device appended. Let's look at an example.

Suppose you have installed an EPC-7 VXI Controller and an HP 1411B Digital Multimeter in your VXI card cage. If the logical address of the HP 1411B is set to 24 (as described in the HP 1411B manual), the VXI address is `16024`. Note that you do not divide the logical address by 8 as you would if you were accessing the VXI device through the HP-IB, as described earlier.

Excluding Address Space for the HP 82335 HP-IB Card (Windows 95 Only)

If you are using an HP 82335 HP-IB Card, which uses memory-mapped I/O addressing, you must exclude the address space required by the HP-IB so that memory manager programs won't try to use that space.

NOTE

The HP 82340 and HP 82341 HP-IB Cards, and the National Instruments GPIB cards do not use memory-mapped I/O addressing, so this section does not apply to those cards. Also, this section does not apply to the built-in HP-IB of an embedded controller.

The HP 82335 HP-IB Card is supported for Windows 95 only, not for Windows NT.

Install the HP 82335 HP-IB Card, following the instructions that came with it. The HP 82335 is pre-set at the factory for select code 7, but the instructions tell you how to change this setting. Normally you should use select code 7. However, if you are using more than one HP 82335 HP-IB card, each card must be set for a different select code in the range 3 through 7.

Once you have installed the HP 82335 HP-IB Card, do the following:

1. Add the appropriate line for your select code to the `[386Enh]` section of your SYSTEM.INI file (in the C:\WIN95 directory):

For Select Code:	Add to SYSTEM.INI:
3	`EMMEXCLUDE=0CC00-0CFFF`
4	`EMMEXCLUDE=0D000-0D3FF`
5	`EMMEXCLUDE=0D400-0D7FF`
6	`EMMEXCLUDE=0D800-0DBFF`
7 (default)	`EMMEXCLUDE=0DC00-0DFFF`

2. If there is a memory manager DEVICE line (for example, `DEVICE=EMM386.EXE`) in the CONFIG.SYS file (in the root directory), you need to modify it. Add a parameter to exclude the address space (for example, `X=DC00-DFFF` for select code 7), as shown in the following table:

For Select Code:	Modify in CONFIG.SYS:
3	`DEVICE=EMM386.EXE X=CC00-CFFF`
4	`DEVICE=EMM386.EXE X=D000-D3FF`
5	`DEVICE=EMM386.EXE X=D400-D7FF`
6	`DEVICE=EMM386.EXE X=D800-DBFF`
7 (default)	`DEVICE=EMM386.EXE X=DC00-DFFF`

3. Reboot your computer (select `Start` ⟹ `Shut Down`) and restart Windows.

If you have installed multiple HP 82335 HP-IB Cards, you must exclude address space for each of them. For example, if you have installed two cards, set to select codes 3 and 7, you'll need to add both of the following lines to the `[386Enh]` section of SYSTEM.INI:

```
EMMEXCLUDE=0CC00-0CFFF
EMMEXCLUDE=0DC00-0DFFF
```

Also, if your CONFIG.SYS file contains the DEVICE line for EMM386.EXE, you must add parameters to it as shown below:

```
DEVICE=EMM386.EXE X=CC00-CFFF X=DC00-DFFF
```

B

Troubleshooting

Troubleshooting

Table B-1. Instrument Control Troubleshooting

Problem	Remedy/Cause
Instruments do not respond at all.	The following conditions must be met: • Instruments must be powered up and connected to the interface by a functioning cable. The appropriate I/O libraries must be installed. • For `To/From VXIplug&play` objects: You must have installed and configured the appropriate VXI*plug&play* driver files for your instrument. Also, the correct VXI*plug&play* address string must be specified in the `Advanced Device Configuration` dialog box for each instrument. The address for each instrument must be unique. • For `Direct I/O`, `Panel Driver`, and `Component Driver` objects: The interface select code and instrument addresses must match settings in the `Address` field of the `Device Configuration` dialog box. The address for each instrument must be unique. Also, the `Live Mode` field in the `Advanced Device Configuration` dialog box must be set to `ON`. • You or your system administrator must properly configure HP VEE to work with instruments. Normally this is done during HP VEE installation. Refer to the installation guide. • For UNIX systems, the UNIX kernel must be configured with the proper drivers and interface cards.
You cannot determine the instrument address.	For GPIO and serial interfaces, the instrument address is the same as the interface select code. HP-IB instrument addresses are set by hardware switches or front panel commands. Refer to your instrument's programming manual for details. VXI devices have logical addresses set by switches on the outside of the cards (usually the cards must be removed from the card cage to access the switches). Refer to Chapter 3 for further information about configuring addresses.
You cannot determine the interface select code.	The interface select codes must be configured with the `I/O Config` utility supplied with the HP I/O libraries. Refer to *Installing the HP I/O Libraries - HP VEE* for further information. The recommended select codes are listed in Table A-1.

C

Instrument I/O Data Type Conversions

Instrument I/O Data Type Conversions

On instrument I/O transactions involving numeric data, HP VEE performs an automatic data-type conversion according to the rules listed below. (These data-type conversions are completely automatic. Normally, you won't need to be concerned with them.)

- On an input transaction (read), `Int16` or `Byte` values from an instrument are converted to `Int32` values, preserving the sign extension. Also, `Real32` values from an instrument are converted to 64-bit `Real` numbers.
- On an output transaction (write), `Int32` or `Real` values are converted to the appropriate output format for the instrument:
  - If an instrument supports the `Real32` format, HP VEE converts 64-bit `Real` values to `Real32` values, which are output to the instrument. If the `Real` value is outside of the range for `Real32` values, an error will occur.
  - If an instrument supports the `Int16` format, HP VEE truncates `Int32` values to `Int16` values, which are output to the instrument. `Real` values are first converted to `Int32` values, which are then truncated and output. However, if a `Real` value is outside the range for an `Int32`, an error will occur.
  - If an instrument supports the `Byte` format, HP VEE truncates `Int32` values to `Byte` values, which are output to the instrument. `Real` values are first converted to `Int32` values, which are then truncated and output. However, if a `Real` value is outside the range for an `Int32`, an error will occur.