HP-IB Command Library for MS™-DOS

HEWLETT PACKARD

DESIGNED FOR HP-IB SYSTEMS

HP Computer Museum
www.hpmuseum.net

For research and education purposes only.

HP-IB Peripheral Driver

Printing History

New editions of this manual will incorporate all material since the previous edition. Update packages, which may be issued between editions, contain replacement and additional pages to be merged into the manual by the user.

The manual printing date and part number indicate its current edition. The printing date changes when a new edition is printed. (Minor corrections and updates which are incorporated at reprint do not cause the date to change.)

The software revision letters adjacent to the date indicate the versions of each disc that were available at the time that this manual was issued. However, some changes to the software product do not require manual changes and, conversely, manual corrections may be done without accompanying product changes. Therefore, do not expect a one-to-one correspondence between software changes and manual updates.

Manual No. 5957-6339
Microfiche No. 5957-6340
Edition 1 ... Sept. 1985 ... 61062AA HP Vectra Disc Rev. B

Notice

HP-IB Peripheral Driver

Introduction

The HP-IB peripheral driver lets you use HP-IB printers and plotters with your HP Vectra or IBM personal computer.

The peripheral driver is part of the 61062AA HP-IB Command Library. To use the driver, you need the HP-IB card, the software disc and this manual.

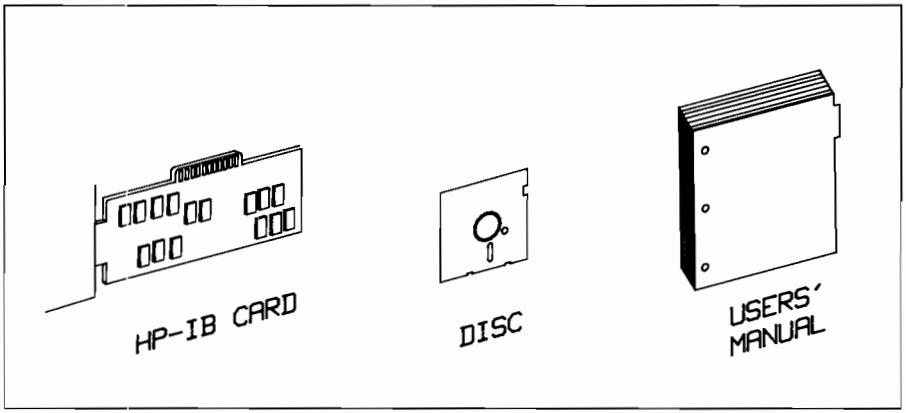


Figure 1. HP-IB Peripheral Driver Package

The card plugs into your computer to provide an HP-IB connector and the necessary hardware for HP-IB communication. The disc contains software to communicate with HP-IB printers and plotters.

This manual explains how to use the software and install the card.

How to Use This Manual

Because the peripheral driver software is so versatile, everyone from the novice to the expert can use it. Accordingly, this manual is organized in sections so you can find the information that *you* need.

Section one, "Getting Started", presents the driver in its simplest form—how to load the files and configure an HP-IB printer as the default system output device.

Section two, "Using the Driver", gives you a more technical description of the driver, and tells you how to change your peripheral configuration.

Section three, "Installing the Card", explains how to configure the HP-IB card and install it in your computer.

Getting Started

The following steps show you how to install the peripheral driver files in your system, and configure an HP-IB printer as your default printer.

1. Check to see if the HP-IB card is installed in your computer. If it's installed, you will see an HP-IB connector in the back of your computer. Figure 2 illustrates an HP-IB connector.

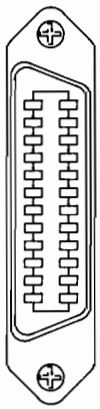


Figure 2. HP-IB Connector

   Go to the "Installing the HP-IB Card" section of this manual for complete instructions on configuring and installing the card.

2. Insert the software disc in drive A. (These steps assume that A is your flexible disc drive, and C is your fixed, or system disc drive.) This disc contains a batch file called PCONFIG.BAT that installs the driver files on your system disc for you.

3. You must run PCONFIG from MS-DOS. If you're using PAM, select the MS-DOS COMMANDS application from the PAM menu.

To copy the files from drive A to drive C, type this line at the MS-DOS prompt:

A:PCONFIG A: C:

Press RETURN. This installs the files from the disc in drive A to the root directory of drive C. If you are using different disc drives, specify them when you run PCONFIG.

When you run PCONFIG as shown above, the following information is displayed on your screen:

```
*****************************************************
*          You specified:                           *
*            - the "copy from" drive as "A:"        *
*            - the "copy to" drive as "C:"          *
*                                                   *
*          Note: "C:" must not be write protected.  *
*                                                   *
*          If this is correct, press any key.       *
*                                                   *
*          If this is incorrect:                    *
*          - press [Ctrl-C]                         *
*          - answer Y to the abort prompt           *
*          - type PCONFIG to start over             *
*****************************************************
```

This verifies your entry and gives you a chance to change it. If the information is correct, press any key. PCONFIG begins copying the files HPIBMODE.COM and HPIB.SYS from the software disc to your system disc.

If you want to change the drive designators, press <Ctrl-C>. The prompt "Terminate batch job (Y/N)?" will appear. Answer Y(es). Then, run PCONFIG again with the correct drive designators.

If you run PCONFIG.BAT without specifying any drives, or if you specify invalid drives, you'll see this message on your screen:

```
*****************************************************
*    Specify the "copy from" drive (the drive containing the *
*    driver software disc), and the "copy to" drive (the     *
*    system disc drive).                                     *
*                                                            *
*    Example: A:PCONFIG A: C:                                *
*                                                            *
*    This installs the driver files from the disc in drive A *
*    to the disc in drive C.                                 *
*                                                            *
*    Remember, the system disc must not be                   *
*    write protected.                                        *
*****************************************************
```

This screen reminds you to run PCONFIG again, specifying drive designators.

In addition to copying the files, PCONFIG also edits your system's CONFIG.SYS file. It adds a command that configures a default HP-IB printer for you. This is the printer for all printed output if you don't select any other device. If you don't have a CONFIG.SYS file, PCONFIG creates one.

Note that you don't have to be concerned about the CONFIG.SYS file since PCONFIG performs this operation automatically. However, if you want to learn more about the CONFIG.SYS file, see your operating system manual.

PCONFIG selects the printer at HP-IB address 701 as the default printer, where 7 is the select code, and 01 is the device address.

The *select code* is the HP-IB designator for the interface. HP computers use 7 as the default select code. Select code 7 is set when the card is installed as described in "Installing the HP-IB Card".

Note *If your HP-IB card was already installed in your computer, check its select code setting. If it is not 7, follow the instructions in "Installing the HP-IB Card" to change it to 7.*

The device address is a unique "address" that is set by switches on the device. Each device on the interface must be set to a different address.

4. Since PCONFIG assumes 701 as the default address, make sure your printer address switches are set to 01. These may be located on the back of your printer, or inside. Address 1 has these switch settings:

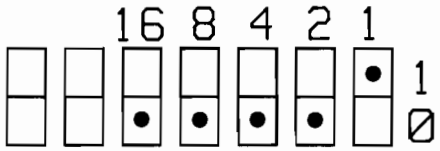


Figure 3. HP-IB Switches

5. Remove the software disc from drive A and restart your system by pressing:

   <CTRL> <Alt> <Delete>

   This re-boots your system and configures the default printer. Because your CONFIG.SYS file was edited, the default printer is now selected each time you turn on your system.

   If you get an "Invalid parameters" error, check your HP-IB card select code setting to make sure it is 7.

6. With the HP-IB card configured and installed, connect the printer to the HP-IB connector on the card with an HP-IB cable:

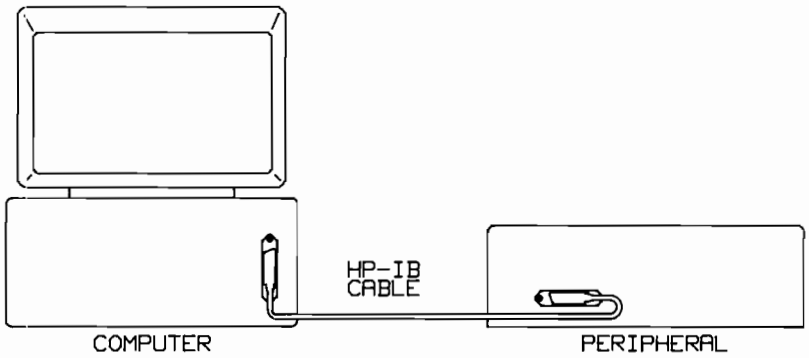


Figure 4. Connecting the Printer

You are now ready to output to the default printer. To test it, press the PRINT SCREEN key. The contents of your screen will be output to the printer.

The HP-IB Peripheral Driver

If you are using only one HP-IB printer with device address 701 as the system output device, all the information you need has been presented in "Getting Started". You can now use your HP-IB printer.

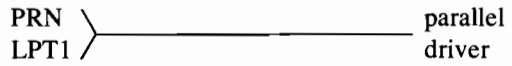
If you have multiple HP-IB printers or an HP-IB printer with a different device address, this section tells you how to use the peripheral driver for any application.

How the Peripheral Driver Works

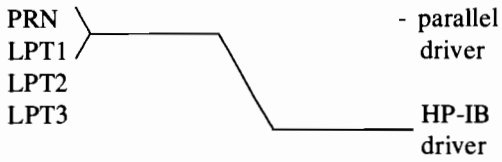
The following system predefined device names are used to identify peripherals in the system.

PRN
LPT1
LPT2
LPT3

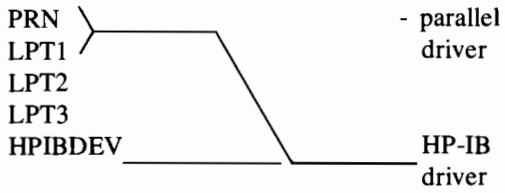
Before the HP-IB peripheral driver is added to your system, an RS-232 driver or a parallel driver or some other device driver should already exist. Without this driver, there would be no way for the operating system to determine where and how to print data. In fact, there may be several types of device drivers already in your system. For example, LPT1 may be assigned to a parallel driver while LPT3 is assigned to an RS-232 driver. The following diagram shows PRN and LPT1 assigned to the parallel driver.



When the HP-IB peripheral driver is added, it redirects output from the parallel driver to the HP-IB driver. The parallel driver remains available in case output later needs to be directed to it again. The diagram below shows PRN and LPT1 redirected to the HP-IB driver.



When the HP-IB peripheral driver is installed, a new device called HPIBDEV is defined as a system device. It is always assigned to the HP-IB driver:



You will learn how to assign and unassign the system devices to the HP-IB driver in this section.

The Peripheral Driver Files

The software disc includes some files that make it easy to configure a single HP-IB printer as the primary system output printer. Other files provide maximum flexibility for assignment of HP-IB devices. The disc contains these peripheral driver files:

PCONFIG.BAT
CONLINE.SYS
HPIB.SYS
HPIBMODE.COM

The PCONFIG.BAT program copies the driver code to your system disc. It is fully explained in "Getting Started". This program uses the CONLINE.SYS and HPIB.SYS files.

The HPIB.SYS file contains the code that controls the HP-IB interface lines when data is sent to the printer. Under default conditions, it also configures the predefined system devices PRN and LPT1 as an HP-IB printer at address 701.

The CONLINE.SYS file is used by PCONFIG.BAT to create the CONFIG.SYS file. The CONFIG.SYS file must reference the HPIB.SYS file with a "DEVICE=" command in order to install and use the peripheral driver.

The HPIBMODE.COM file contains the MS-DOS HPIBMODE command. Once the HPIB.SYS driver is installed, this command lets you assign or reassign a peripheral device from MS-DOS without having to reboot the system.

The DEVICE command for HP-IB peripherals in the CONFIG.SYS file is of the following form.

DEVICE=HPIB.SYS <control string>

The syntax for the MS-DOS HPIBMODE command is:

HPIBMODE <control string>

The <control string> for both these commands is in the form:

[[-] <system device>][= <device selector>][<timeout>]

Parameters:

[-] - the optional minus sign unassigns the current driver. It sets LPT1 and PRN back to their original predefined driver.

<system device> - a predefined system device keyword. It can be LPT1, LPT2, LPT3 or HPIB (For HPIBDEV). Default is LPT1.

Note: MS-DOS usually assigns LPT1 and PRN to the same driver. Changing LPT1 changes PRN accordingly.

<device selector> - a numeric expression specifying the source or destination of an I/O operation. It is the combination of an interface select code and a device primary address. The <device selector> is calculated as:

(100 * interface select code) + device address

Default is 701.

<timeout> - The length of the delay before peripheral-not-ready is assumed. The timeout period is specified in seconds. Default is 30 seconds. The allowable range is 0-999 seconds. Specifying zero seconds disables the timeout.

Note that an application may retry an operation after a timeout. Thus, the effective timeout can be twice the value specified.

Loading the Peripheral Driver

PCONFIG uses the CONLINE.SYS file to create a CONFIG.SYS file. To load the peripheral driver, the CONFIG.SYS file must reference the HPIB.SYS file with a "DEVICE=" command. The PCONFIG program does this for you, but assumes the HP-IB peripheral is the PRN or LPT1 printer at address 701. To use LPT2, LPT3 or another system device, or an HP-IB peripheral with a different address, you create your own CONFIG.SYS file.

Use any editor familiar to you to create a CONFIG.SYS file. It must reference the HPIB.SYS file in a "DEVICE= " command and it must include the appropriate parameters. The optional parameters are described above under "Syntax."

For example, the line shown next would configure the printer at 702 to the predefined device LPT2 with the default 30-second timeout.

```
DEVICE=HPIB.SYS LPT2=702
```

Each time the system is booted, such as at power on, device assignments are made according to the commands in the current CONFIG.SYS file. In the above example, the printer at address 702 would become the system printer LPT2.

Configuring a Peripheral from MS-DOS

Once you have defined the peripheral driver in your system (by referencing it in your CONFIG.SYS file), you can assign or reassign a peripheral device from MS-DOS. To do this, type the HPIBMODE command with the desired parameters at the MS-DOS prompt. Here are some examples:

```
> HPIBMODE
```

- Assign the device at default address 701 to the default driver LPT1/PRN. Use the default 30-second timeout.

```
> HPIBMODE = 802
```

- Assign the device at address 2 on select code 8 to the default driver LPT1/PRN. Use the default 30-second timeout.

```
> HPIBMODE LPT3
```

- Assign the device at default address 701 to the LPT3 driver. Use the default 30-second timeout.

```
> HPIBMODE HPIB=722
```

- This example uses the new driver — HPIBDEV — supplied by the HP-IB peripheral driver. It assigns the device at 722 to the HPIBDEV driver using the default timeout.

Using HPIBDEV

For most applications, you can assign a peripheral to any of the predefined devices. For example, to assign an HP-IB plotter at address 705 to predefined device LPT2 from MS-DOS, you would type the following line at the MS-DOS prompt.

```
> HPIBMODE LPT2=705
```

Then, in your plotter application, you would specify LPT2 as your output device.

However, if you want to read back from the plotter (e.g., return its current pen location), you must use the predefined device HPIBDEV. It is the only predefined device that can read from a device as well as write to it. In this case, you type the line shown next at the MS-DOS prompt.

```
> HPIBMODE HPIB=705
```

In your application, you now specify "HPIBDEV" as the I/O device. This lets you receive information from the plotter at 705, as well as output to it.

You can also use the MS-DOS IOCTL function of HPIBDEV to reconfigure the driver. For example, the following BASIC lines configure a device for program output.

```
10 OPEN HPIBDEV FOR OUTPUT AS #1
20 IOCTL #1, "HPIB=703 4;"
```

Comments

Only one device is configured by the DEVICE=HPIB.SYS <control string> line in your CONFIG.SYS file. To change a device, or to select additional devices, you must use the HPIB-MODE command. Each device must have a separate HP-IB device address.

The name of the HP-IB MS-DOS driver "HPIBDEV" cannot be changed.

You can reconfigure a device as often as you want. It is not necessary to unassign it (with the "-" parameter) before reconfiguring.

You can configure multiple peripherals at startup by using an AUTOEXEC.BAT file. For example, you could configure both a printer and plotter at power-up. You would assign the printer to LPT1 in the CONFIG.SYS file, and assign the plotter to LPT2 with an HPIBMODE command in an AUTOEXEC.BAT file. Then, they would both be configured when you turned on your system.

PRINT SCREEN does not interrupt a print or plot that is taking place. In this case, the peripheral driver ignores the PRINT SCREEN key.

If you specify an invalid select code in the HPIBMODE command, you get an "invalid parameters" error. For example, if there is no HP-IB card at select code 7 when you specify <device selector> = 708, this error is generated.

The following HP-IB activity occurs whenever you reconfigure a peripheral with an HPIBMODE command or an IOCTL call.

- IFC (Interface Clear) is set, then cleared.
- REN (Remote Enable) is cleared, then set. This places all devices on the bus into Local mode.
- ATN (Attention) is cleared.

Installing the HP-IB Card

To use the HP-IB peripheral driver, you must install the HP-IB card into your computer. Begin by observing proper handling procedures:

- HANDLE GENTLY Do not drop the card or handle it roughly. Be careful when unpacking the card and during installation.

- HOLD ONLY BY THE EDGES Never touch any part of the card except the edges. Do not put fingerprints on the connector.

- PROTECT FROM STATIC ELECTRICITY The interface card can be damaged by static electricity. For protecton, the card is packed in an anti-static bag.

  Leave the card in its anti-static bag until you are ready to install it.

  Save the anti-static bag so you can protect the card if you have to remove it from the computer.

Configuring the Card

The HP-IB card configuration switches set the card's operating parameters. They should be set as shown in Figure 5 to use the peripheral driver:

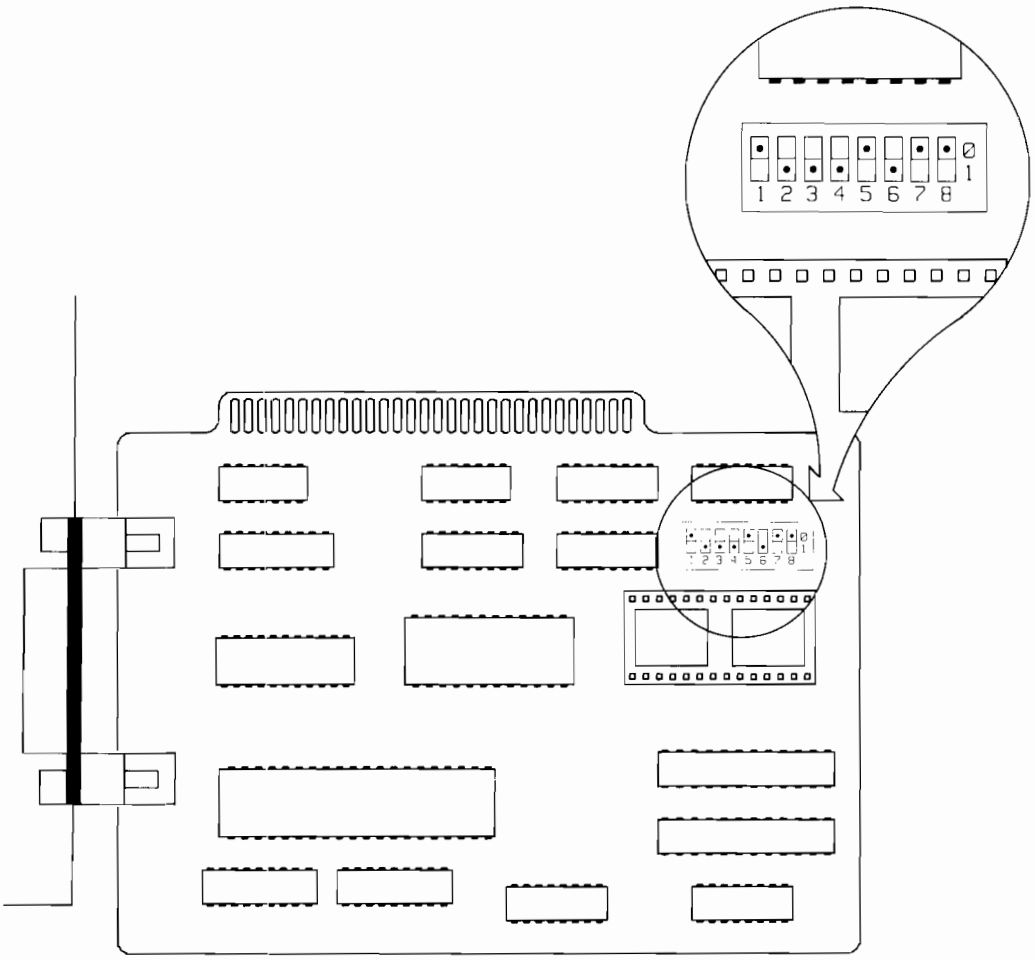
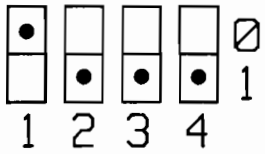


Figure 5. HP-IB Card Configuration Switches

To check the configuration, hold the card with the edge connector up, then locate the switch block from Figure 5. Set the switches as described in the following sections.

Card Address

This card address switches (1 - 4) determine the interface select code:



(The dot indicates the depressed position.)

	Switch				
Address	*1*	*2*	*3*	*4*	*Select Code*
C0000	0	0	0	0	16
C4000	0	0	0	1	1
C8000	0	0	1	0	2 - reserved for Fixed Disc Drive Controller
CC000	0	0	1	1	3
D0000	0	1	0	0	4 - reserved for PC Cluster Adapter (LAN)
D4000	0	1	0	1	5
D8000	0	1	1	0	6
DC000	0	1	1	1	7 <————
E0000	1	0	0	0	8
E4000	1	0	0	1	9
E8000	1	0	1	0	10
EC000	1	0	1	1	11
F0000	1	1	0	0	12 - reserved for system ROM
F4000	1	1	0	1	13 - reserved for system ROM
F8000	1	1	1	0	14 - reserved for system ROM
FC000	1	1	1	1	15 - reserved for system ROM

Table 1. Card Addressing

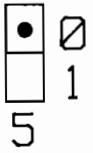
Select a value that does not conflict with any other installed interface card — select code 7 is convenient for HP computers.

If you have multiple HP-IB cards installed, each must be at a separate address. And if one of them is a mass storage interface, that card must be set to a greater select code value than the others. For example, you would set a peripheral driver interface to 7, and a mass storage interface to 8 if they are both plugged into your computer.

Use Table 1 along with your computer documentation to help you set the card address.

Boot Device

Set switch 5 to zero:



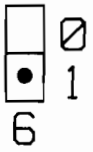
To boot from:	Set Switch 5 to:
- the internal disc drive	0 <———
- an external disc drive	1

Table 2. Boot Device Switch

This switch selects the boot device which can be an internal or an external disc drive. Select the internal disc drive as the boot source to use the peripheral driver.

System Controller

Set switch 6 to position 1:



Set switch 6 to position 1:

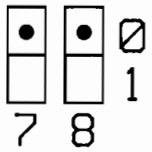
System Controller:	Switch 6:
- not system controller	0
- system controller	1 <———

Table 3. System Controller

This switch determines whether or not the HP-IB card is the system controller. The card must be system controller to work with the peripheral driver.

Interrupt Level

Set switches 7 and 8 to zero:



Interrupt Level	Switch 7	Switch 8	
3	0	0	<———
4	0	1	
5	1	0	
6	1	1	

Table 4. Interrupt Level

These switches are not used by the peripheral driver. Set them both to zero.

Installing the Card

1. Turn off and unplug your computer.
2. In the case of an HP Vectra (or IBM PC/AT), unlock the front cover.
3. Remove the rear panel periphery screws, then remove the cover.

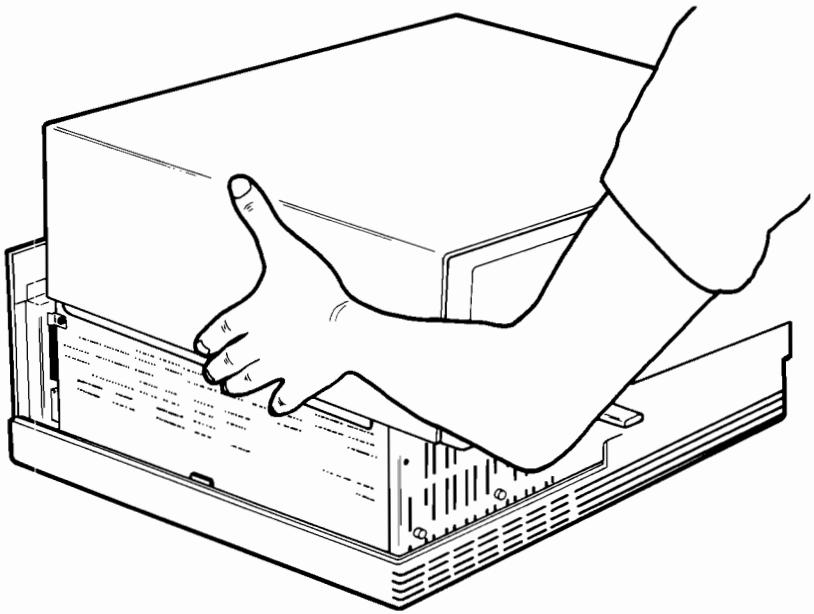


Figure 6. Removing the Cover

4. Select a suitable empty slot. You can install the card in any slot of an HP Vectra or an IBM PC or PC/AT. You can install it in any slot of an IBM PC/XT *except* slot 8 (the slots are labeled).
5. Remove the cover plate of the slot by removing its mounting screw and lifting it from its location (see Figure 7).
   7).

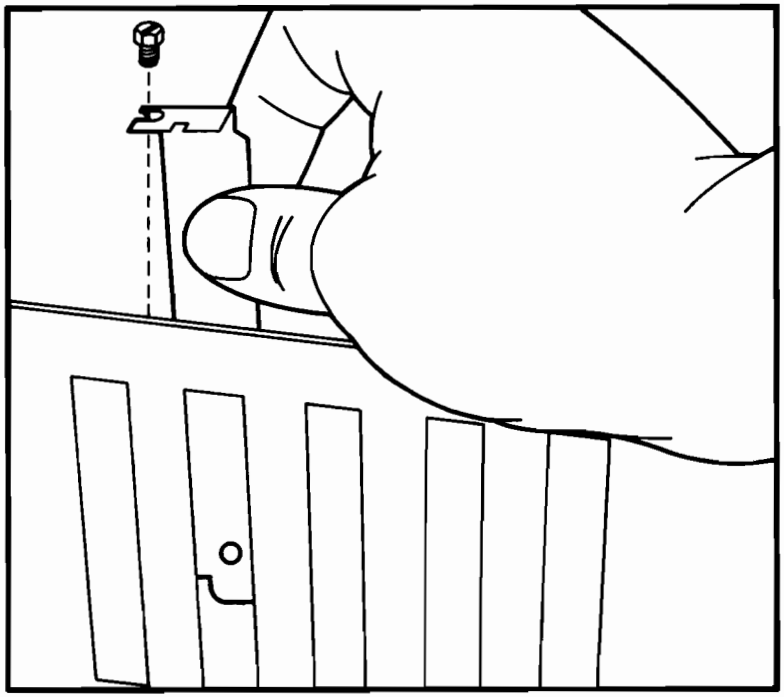


Figure 7. Removing the Slot Cover Plate

6. Remove the HP-IB card from the anti-static bag, holding it by its edges. Check the settings of the switches on the card (see previous sections).

7. Hold the card with the edge connector down and the cable connector toward the rear of the computer. Insert the plug in connector into the slot and press the card down firmly to make sure the connector is fully seated. (Figure 8.)

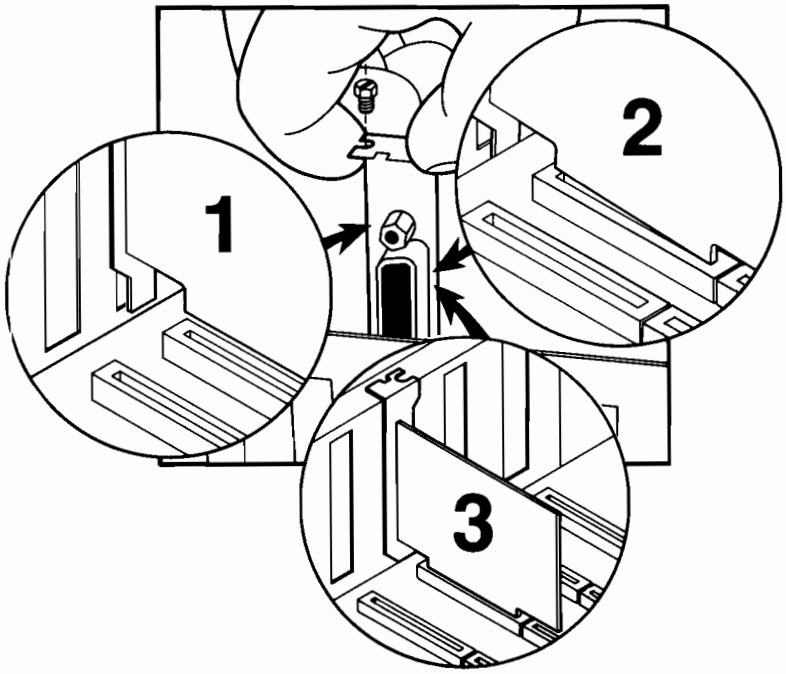


Figure 8. Installing the Card

8. Replace the slot cover screw to hold the HP-IB card in place. (Store the slot cover for later use.)

9. Replace the computer cover and install the cover screws.

Connecting the Cable

An HP-IB system can accommodate up to 14 devices in addition to the computer. To set up a peripheral and connect it to the interface, follow these instructions:

1. Refer to your particular paripheral manual to set up the peripheral, connect its power cables, and run its self-test before connecting it to the computer.

2. Turn off the peripheral.

3. Determine a bus address for the peripheral that does not conflict with the addresses of other devices. Set its HP-IB switches accordingly. (The HP-IB Command Library uses address 30 for the controller address.)

4. Press the HP-IB cable plug into the HP-IB connector on the interface card and tighten the connector screws. The plug and connector fit only one way; if you're having trouble connecting them, rotate the plug 180 degrees and try again.

5. Connect the other end of the HP-IB cable to the HP-IB connector of the peripheral. Tighten the connector screws.

Here are some points to remember when connecting devices to the interface:

- You can connect up to 14 devices to a single interface.

- You can interconnect devices in any scheme as long as there is an unbroken path between each device and the computer:

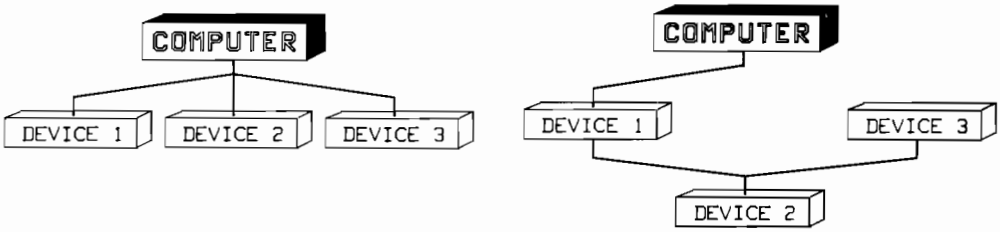


Figure 9. Connecting Devices

- You can connect HP-IB Cables in piggyback fashion. However, do not stack more than three on a device because weight of the connectors and cables could damage the socket.

- The total cable length on one interface should not exceed the lesser of 20 meters, or two meters times the number of connected devices (the interface is considered as one device).

Printing History

New editions of this manual will incorporate all material since the previous edition. Update packages, which may be issued between editions, contain replacement and additional pages to be merged into the manual by the user.

The manual printing date and part number indicate its current edition. The printing date changes when a new edition is printed. (Minor corrections and updates which are incorporated at reprint do not cause the date to change.)

The software revision letters adjacent to the date indicate the versions of each disc that were available at the time that this manual was issued. However, some changes to the software product do not require manual changes and, conversely, manual corrections may be done without accompanying product changes. Therefore, do not expect a one-to-one correspondence between software changes and manual updates.

Edition 1..April 1985..14857A HP Touchscreen Disc Rev. A
...61062A IBM Disc Rev. A

Edition 2..Sept 1985..61062AA HP Vectra Disc Rev. B
..14857AA HP Series I50 Disc Rev. A

Notice

Support for your HP-IB Command Library

The support you receive will depend upon which one of the two HP-IB Command Libraries you are using. The HP 14857A is the HP-IB Command Library for the HP Touchscreen PC; and the HP 61062A contains the HP-IB Interface Card and Command Library for the IBM PC family.

Software Support

Software support is the same for both the HP14857A and the HP 61062A. If you are unable to execute your program and suspect a software bug, please contact your local HP sales office for assistance. All reported problems and solutions are maintained in a central file for easy reference.

Software Updates

Notification of software updates will be made through the Hewlett Packard/ONLINE section of CompuServe and new product announcements from HP. For more details on CompuServe, contact your local HP Dealer or sales office.

Software License

Software products from Hewlett-Packard are copyrighted and licensed by HP. The right to use the software is sold only on the condition that you agree to the License Agreement shown later in this front matter. If you do not agree to the terms of the license, you may return the unopened package for a full refund. ***However, opening the media envelope indicates your acceptance of these terms and conditions.***

Software Warranty

A complete software warranty statement is given on succeeding pages.

Hardware Support

Hardware support differs slightly for the two products. If you experience a hardware interface problem when using the HP 14857A, please consult the Series 100 Support Guide for the HP Touchscreen PC for assistance.

To verify an HP 61062A interface failure, substitute a known good HP-IB device for the instrument(s) you are using in your system. If you still have a problem with the interface, fill out the HP-IB Card Exchange form for the

HP 61062A found in the the back of this guide and return with the defective card to one of the HP Worldwide Field Repair Centers also listed at the rear.

Plug-in interface cards and other low cost items, due to high repair labor costs, are considered a throw-away assembly when they fail. It is important that the *exchange form* accompanies the defective card for both in-warranty and out-of-warranty replacements.

Hardware Warranty

The HP-IB interface card shipped with the HP 61062A has a standard one year return-to-HP warranty. A complete hardware warranty statement is given on succceeding pages. A hardware statement for the HP 14857A product is given in the Series 100 Support Guide.

Software License Agreement

In return for the payment of the one time fee for this software product, Customer receives from Hewlett-Packard (HP) a license to use the product subject to the following terms and conditions.

1. The product may be used without time limit on one personal computer or workstation.
2. A separate license agreement and fee is required for each personal computer or workstation on which the product is used.
3. The software product may not be duplicated or copied except for archive purposes, program error verification, or to replace defective media, and all copies made must bear the copyright notices contained in the original.
4. This license and the software product may be transferred to a third party, with prior written consent from HP, provided the third party agrees to all the terms of this License Agreement and Customer does not retain any copies of the software product.

5. Purchase of this license does not transfer any right, title, or interest in the software product to Customer except as specifically set forth in this License Agreement. Customer is on notice that the software product is protected under the copyright laws. This software product may have been developed by an independent third party software supplier named in this package, which holds copyright or other proprietary rights to the software product. Customer may be held responsible by this supplier for any infringement of such rights by Customer.

6. HP reserves the right to terminate this license upon breach. In the event of termination, Customer will either return all copies of the product to HP or, with HP's prior consent, provide HP with a certificate of destruction of all copies.

7. In the event Customer modifies the software product or includes it in any other software program, upon termination of this license Customer agrees either to remove the software product or any portion thereof from the modified program and return it to HP or to provide HP with a certificate of destruction thereof.

Software Warranty

HP warrants for a period of NINETY (90) DAYS from the date of purchase that the software product will execute its programming instructions when properly installed on the personal computer or workstation indicated on this package. HP does not warrant that the operation of the software will be uninterrupted or error free. In the event that this software product fails to execute its programming instructions during the warranty period, Customer's remedy shall be to return the diskette(s) or tape cartridges(s) (''media'') to HP for replacement. Should HP be unable to replace the media within a reasonable amount of time, Customer's alternate remedy shall be a refund of the purchase price upon return of the product and all copies.

Media. HP warrants the media upon which this product is recorded to be free from defects in materials and workmanship under normal use for a period of NINETY (90) DAYS from the date of purchase. In the event any media prove to be defective during the warranty period, Customer's remedy shall be to return the media to HP for replacement. Should HP be unable to replace the media within a reasonable amount of time, Customer's alternate remedy shall be a refund of the purchase price upon return of the product and all copies.

Notice of Warranty Claims. Customer must notify HP in writing of any warranty claim no later than thirty (30) days after the expiration of the warranty period.

Hardware Warranty Statement (HP 61062A)

Hewlett Packard warrants its instrument products against defects in materials and workmanship for a period of one year from receipt by the end user. During the warranty period, HP will either, at its option, repair or replace products which prove to be defective.

Should HP be unable to repair or replace the product within a reasonable amount of time, the customer's alternate remedy shall be a refund of the purchase price upon return of the product.

Limitation of HP Warranties

HP makes no other express warranty, whether written or oral with respect to these products. Any implied warranty of merchantability or fitness is limited to the duration of these written warranties. Some states or provinces do not allow limitations on how long an implied warranty lasts, so the above limitation or exclusion may not apply to you*.

These warranties give specific legal rights, and you may also have other rights which vary from state-to-state, or province-to-province.

Exclusive Remedies

The remedies provided above are Customer's sole and exclusive remedies. In no event shall HP be liable for any direct, indirect, special, incidental, or consequential damages (including lost profit) whether based on warranty, contract, tort, or any other legal theory. Some states or provinces do not allow the exclusion or limitation of incidental or consequential damages, so the above limitation or exclusion may not apply to you.

Warranty Service

Warranty service can be obtained from any of the Worldwide Field Repair Centers listed at the rear of this manual.

*UK only. If you are ''consumer'' as defined by statutes, parts of this statement may not apply to you.

Table of Contents

Preface

The HP-IB Command Library for MS-DOS lets you control HP-IB instruments with a personal computer. This manual explains how to use the Library for BASIC and Pascal applications.

To use the manual effectively, you should be familiar with MS-DOS computers, as well as BASIC and/or Pascal. You should have a working knowledge of HP-IB programming and instrumentation. (For your convenience, appendix A of this manual reviews some fundamentals of HP-IB communication.)

The chapter for each programming language contains an example program. These serve to illustrate the commands and demonstrate their usage in a typical measurement task. You can step through the examples on your equipment to learn how to program with the HP-IB Command Library.

1 Introduction

What Is The HP-IB Command Library?

The HP-IB Command Library for MS-DOS is a series of commands that lets you control HP-IB instruments with these personal computers:

- HP Vectra Personal Computer
- HP Touchscreen family (150 Series) Personal Computers
- IBM PC/XT/AT (and compatible) Personal Computers

The commands are available on the Library disc in both BASIC and Pascal. For convenience, example programs in both languages are also included. In addition, a READ.ME file gives you current notes about the Library, as well as pointers on using the Library effectively.

The Library for an HP Vectra (or IBM) PC includes a card to provide the necessary electrical/mechanical interface for HP-IB communication. Figure 1-1. illustrates the Library contents. You can use it to make sure you have the correct Library for your system.

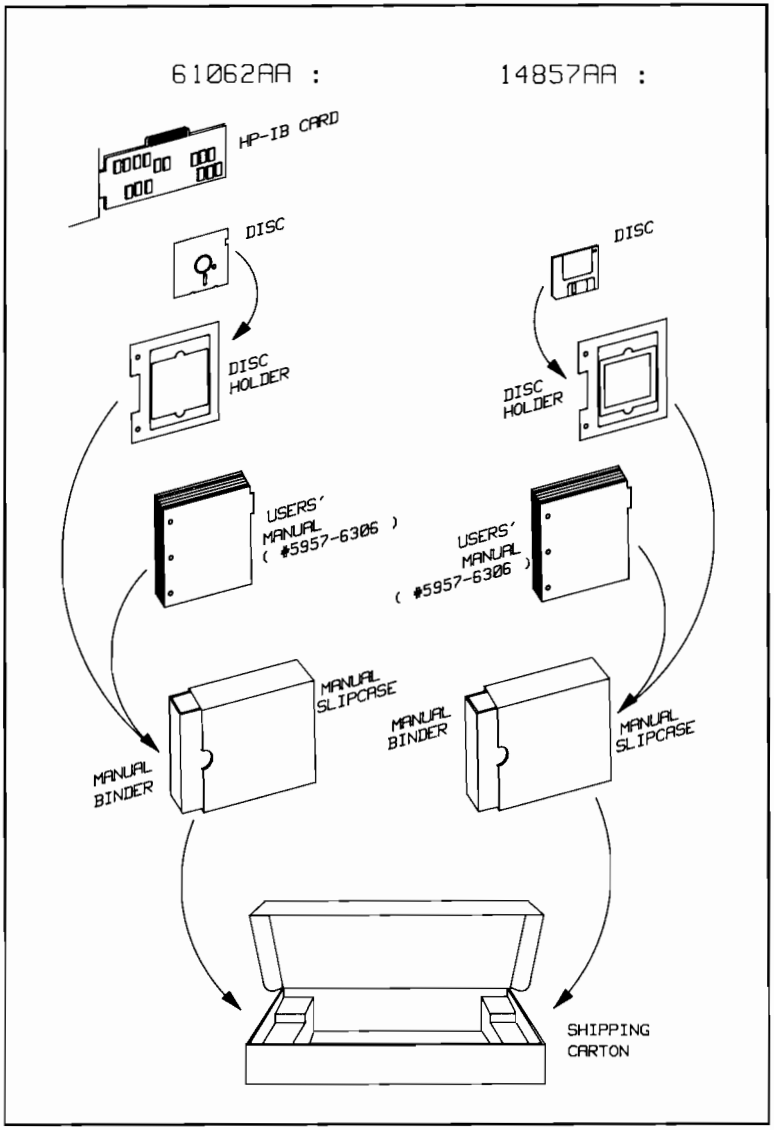


Figure 1-1. HP-IB Library Contents

The Library commands give you access to these IEEE 488 standard control lines and bus commands:

CONTROL LINES:	IFC	- Interface Clear
	ATN	- Attention
	SRQ	- Service Request
	REN	- Remote Enable
	EOI	- End Or Identify
UNIVERSAL BUS COMMANDS:	LLO	- Local Lockout
	DCL	- Device Clear
	SPE	- Serial Poll Enable
	SPD	- Serial Poll Disable
	PPU	- Parallel Poll Unconfigure
ADDRESSED BUS COMMANDS:	GTL	- Go to Local
	SDC	- Selected Device Clear
	PPC	- Parallel Poll Configure
	GET	- Group Execute Trigger
OTHER BUS COMMANDS:	UNL	- Unlisten
	UNT	- Untalk

The HP-IB Command Library is completely compatible with the Hewlett-Packard PC Instruments System. This is a low cost instrument system that uses a personal computer to control a series of quality measurement devices. Thus, you can control both HP-IB instruments and PC instruments from the same BASIC program. See appendix D for an example program using PC Instruments and the HP-IB Command Library.

What Are The Library Files?

The Library disc contains these command setup and execution files:

BASIC files:	PCIBILC.EXE	- Interface module.
	PCIBILC.BLD	- Interface module loader.
	HPIB.SYN	- Syntax for call parameters.
	HPIB.PLD	- BASIC Library commands.
	SETUP.BAS	- Subroutine initialization file.
	PCIBILC.HPE	- Error message file.
	PCIBAS.BAT	- Batch file to run BASIC.
	EXAMPLE.BAS	- BASIC example program.
	NODOC.BAS	- Uncommented SETUP.BAS
Pascal files:	HPIB.LIB	- Pascal Library commands.
	IODECL.EX	- Type and constant declarations.
	IOPROC.EX	- Procedure declarations.
	EXAMPLE.PAS	- Pascal example program.
Information files:	CHECKSUM.EXE	- Data validation program.
	CHECKSUM.DOC	- Data validation comments.
	READ.ME	- Changes, corrections, additions.

The following chapters show you how to copy the files to a work disc for programming convenience.

What Are The Commands?

The Library commands give you a great deal of programming flexibility. For example, you can enter or output data directly as strings, real numbers or arrays:

IOENTER - Enter a single real number from a device.
IOENTERA - Enter an array of real numbers from a device.
IOENTERS - Enter an ASCII string from a device.
IOOUTPUT - Output a single real number to a device.
IOOUTPUTA - Output an array of real numbers to a device.
IOOUTPUTS - Output an ASCII string to a device.

These six commands perform over 80% of the I/O tasks in most applications. Since the Library has a built-in number builder, there's no need to convert data between string and numeric format. This reduces the number of programming statements and the associated overhead.

The Library also provides for serial and parallel device polling and status checking:

IOPPOLL - Perform a parallel poll.
IOPPOLLC - Perform a parallel poll configure.
IOPPOLLU - Perform a parallel poll unconfigure.
IOSPOLL - Perform a serial poll.
IOSTATUS - Determine the status of the HP-IB.

These commands let you check instrument and bus status whenever your program requires it.

The remaining commands give you access to various HP-IB control lines and bus commands:

IOABORT - Abort all interface activity.
IOCLEAR - Return a device to a known state.
IOCONTROL - Write information directly to the interface.
IOEOI - Control the interface EOI mode.
IOEOL - Define an end-of-line string for output.
IOGETTERM - Determine the reason for a read termination.
IOLLOCKOUT - Disable device front panel operation.
IOLOCAL - Enable device front panel operation.
IOMATCH - Define a read termination character.
IOREMOTE - Place a device in REMOTE mode.
IORESET - Set the interface to its start-up configuration.
IOSEND - Send user-specified HP-IB commands.
IOTIMEOUT - Set a timeout value.
IOTRIGGER - Trigger a device.

What Are The System Requirements?

To program with the commands in BASIC, use Vectra BASIC on an HP Vectra PC and GW BASIC on an HP Series 150 computer. For an IBM or compatible, use BASICA (version 2.0 or later).

For Pascal programming, use MS-Pascal, minimum version 3.0, on all systems. (Some earlier versions of MS-Pascal require an 80287 co-processor to properly access the real math routines on an HP Vectra PC or IBM PC/AT.)

Note

Pascal by IBM is not compatible with the HP-IB Command Library.

The Library requires a minimum of 256K bytes of memory. An additional 60K bytes is required to install a BASIC program with Library commands as an application under PAM. Refer to the manual shipped with your Series 150 computer, "Using Your HP Touchscreen Personal Computer", Appendix C, for details on running applications without PAM in memory.

System requirements are summarized in Table 1-1.

	HP Vectra PC	Series 150 PC	IBM PCs (and compatible)
Library	61062AA	14857AA	61062AA
Operating System	MS-DOS (minimum 3.1)	MS-DOS (minimum 2.11)	PC-DOS (minimum 2.1)
BASIC Language	Vectra BASIC	GW BASIC	BASICA (minimum 2.0)
Pascal Language	MS-Pascal	MS-Pascal (minimum version 3.0 for all)	MS-Pascal
Required Memory	256K	256K (512K recommended)	256K
Interface Card	supplied with 61062AA	built-in (select code fixed at 7)	supplied with 61062AA

Table 1-1. System Requirements

What Are The Hardware Requirements?

The Library for an HP Vectra (and IBM) personal computer includes an HP-IB card. It plugs into the computer to provide the electrical/ mechanical hardware for HP-IB instrument control.

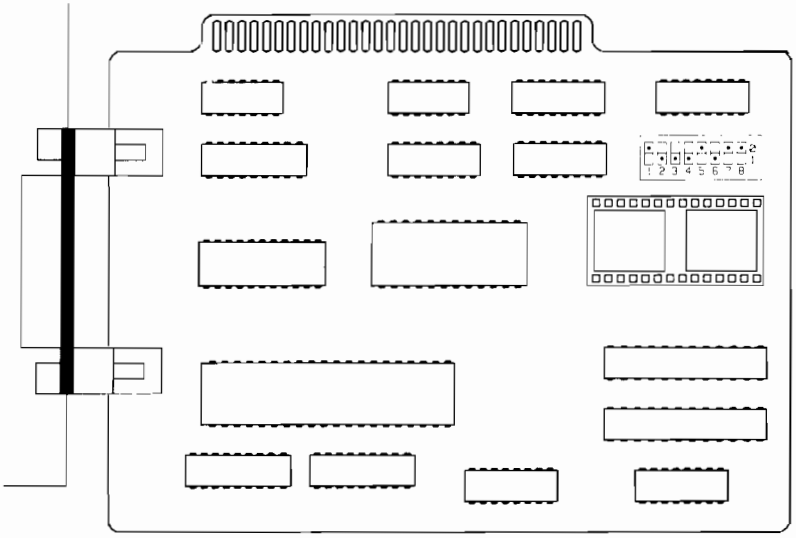


Figure 1-2. HP-IB Card

The card contains a switch block for setting the select code, boot device, system controller and interrupt level parameters.

With the card installed, you use HP-IB cables to connect instruments to the HP-IB connector on the card:

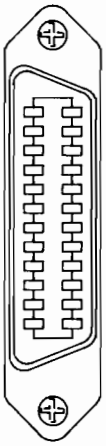


Figure 1-3. HP-IB Connector

You can then use the Library software to write and execute application programs to control those instruments.

To configure and install the card, follow the instructions in Appendix B.

No additional card is necessary for an HP Series 150 computer. It already contains the necessary hardware.

2 Interpretive BASIC Programming

Introduction

This chapter explains how to use the HP-IB Command Library for Interpretive BASIC programming.

The first section tells you how to copy the necessary Library files to a work disc.

The next section shows you how to write a program using several Library commands. Your HP-IB Command Library disc contains a similar version of this program. You can run it on your system to observe the results.

The last section is a syntax reference for the commands as they are used with Interpretive BASIC.

Getting Started

To begin programming in BASIC, copy the BASIC files to a work disc and define some necessary system parameters. The following instructions explain how to do this with both PAM and MS-DOS.

Copying with PAM

These steps assume that B is the source disc and A is the work disc.

1. Enter the File Manager by pressing its softkey or selecting its application from the PAM menu.

2. Create a Library directory on the work disc:

   - Press the Make Dir softkey.
   - Enter A:\LIB in response to the directory name prompt.
   - Press Start Make Dir.
   - Press Exit Make Dir when the directory is created and the prompt appears.

3. Insert the Library disc in drive B and begin copying the BASIC files to the Library directory:

   - Press the Copy File softkey.
   - Enter B:PCIBILC.EXE in response to the "from file" prompt.
   - Enter A:\LIB\PCIBILC.EXE in response to the "to file" prompt.
   - Press the Start Copy softkey.

   Repeat this step, with the appropriate filename, for all BASIC files:

   PCIBILC.EXE
   PCIBILC.BLD
   HPIB.SYN
   HPIB.PLD
   SETUP.BAS
   PCIBILC.HPE
   PCIBAS.BAT
   NODOC.BAS

   - Press Exit Copy when you're finished.

4. Press the Exit FILE MGR softkey.

Copying with MS-DOS

These steps assume that B is the source disc and A is the work disc.

1. Select the MS-DOS COMMANDS application from PAM.

2. From MS-DOS, select the A disc and create a Library directory on it:

```
> A:
> MKDIR \LIB
```

3. Select the Library directory as the current directory:

```
> CD \LIB
```

4. Insert the Library disc in drive B and begin copying the BASIC files to the Library directory:

```
>COPY B:PCIBILC.EXE
>COPY B:PCIBILC.BLD
>COPY B:HPIB.SYN
>COPY B:HPIB.PLD
>COPY B:SETUP.BAS
>COPY B:PCIBILC.HPE
>COPY B:PCIBAS.BAT
>COPY B:NODOC.BAS
```

5. Make sure there is a copy of the appropriate version of BASIC in the root directory of your work disc.

Setting the Environment

In preparation for BASIC programming, you must set some environment characteristics. Follow these steps to assign an environment variable and a search path, and to select BASIC:

1. Select the MS-DOS COMMANDS application from PAM.

2. From MS-DOS, assign the environment variable "PCIB":

```
> SET PCIB=A:\LIB
```

   This variable identifies the location of the Library. If this variable is not set correctly, you get a "file not found" error when you run a BASIC program.

   If you get an "out of environment space" message, type SET to see your current environment variables. To create more environment space, reduce the number of variables and re-boot.

3. In preparation for loading BASIC, set PATH to reflect the location of BASIC. This step assumes BASIC is in the root directory on the work disc.

```
> PATH=A:\
```

4. Finally, load BASIC into memory. Type:

```
> GWBASIC     (for an HP Vectra or series 150 PC)
    or
> BASICA      (for an IBM or compatible PC)
```

The language you choose depends on your computer. Refer to your computer manual for additional language installation instructions.

Note

If you plan to use the BASICA "SHELL" command, you must use the Library file PCIBAS.BAT to load BASICA. Also, you must load BASIC with PCIBAS.BAT on some IBM-compatibles to properly access all required memory.

From MS-DOS, type:

```
>PCIBAS
```

This batch file loads BASICA and reserves sufficient memory to execute the SHELL command from within BASICA.

Programming in BASIC

For BASIC programming, the Library is implemented as a series of assembly language subroutine calls. To access the subroutines, your application program must include the information from the SETUP.BAS Library file. This file acts as a header for your application program to provide entry points into the subroutine calls. (Appendix C has a complete listing of SETUP.BAS.)

To save memory, you can use the uncommented version of SETUP.BAS called NODOC.BAS supplied on the Library disc.

Figure 2-1 illustrates your application program with the SETUP.BAS information:

```
1
2 These lines contain
3 the SETUP.BAS file
. information that
. calls the necessary
. command subroutines.

999

1000
.
. These lines contain
. your BASIC program.

9999
```

Figure 2-1. SETUP.BAS And Your Program

There are several ways to combine your application program with the SETUP.BAS information. One way is to write your program in a separate file, then merge SETUP.BAS into it.

With this method, your program should begin at line 1000. When you are ready to merge, load your program and type:

```
MERGE "SETUP"
```

Since SETUP.BAS starts at line 5 and your program at line 1000, this merges SETUP.BAS into the beginning of your application program. You can save the result under your application program name:

```
SAVE "PROGRAM"
```

Another way to use the SETUP.BAS file is to load it and write your application program within it. Again, start line numbering at 1000, after the SETUP.BAS program lines.

In this case, you do not have to merge anything, but you will want to save the result under a new filename so you don't overwrite SETUP.BAS. Example:

```
LOAD "SETUP"
```

Start your application at line 1000. When you're finished, save the result:

```
SAVE "PROGRAM"
```

In the following example, the BASIC program is written within the SETUP.BAS file.

Writing a BASIC Program

In an application program, you typically use the Library commands in the following manner to execute an operation:

- Set up the required variables.
- Perform the operation.
- Test to see if the operation completed successfully.

In this example, you follow these steps to program two instruments - an HP3325A Synthesizer/Function Generator and an HP3456A Digital Voltmeter. You program the source to output a 2 Vrms signal, swept from 1 KHz to 10 KHz. You program the DVM to take 20 readings from the signal and output them to an array. Finally, you display the readings on the screen.

From BASIC, begin by loading SETUP.BAS:

```
LOAD "SETUP"
```

Generally, you start line numbering at 1000, after the subroutine calling information, with "AUTO 1000". Here, the line numbers have been set to coincide with those of the example file EXAMPLE.BAS on the Library disc.

First, define some working variables:

```
1070  OPTION BASE 1
1080  MAX.ELEMENTS = 20
1090  DIM READINGS (MAX.ELEMENTS)
1100  ACT.ELEMENTS = 0
1110  CODES$ = SPACE$(50)
```

1070 Set the array base to 1 so array element numbering begins with 1 instead of 0.

1080 Define a maximum-readings variable (MAX.ELEMENTS) so you can easily change this parameter when desired.

1090 Dimension an array (READINGS) to hold the readings taken by the voltmeter.

1100 Set the variable for the actual number of elements read by IOENTERA - ACT.ELEMENTS - to 0. ACT.ELEMENTS is an array parameter that must be dimensioned or initialized prior to its use in IOENTERA.

1110 Initialize a string (CODES$) to hold a sufficient number of instrument programming codes.

Note *If your program chains to other programs, you will need COMMON declarations to pass parameters to those programs. Also, you must call DEF.ERR upon entering a chained program to set up pointers to the Library error variables. The SETUP.BAS file in Appendix C contains information on these topics. For more information on chaining, see your BASIC manual.*

Define address variables and initialize the bus:

```
1150  ISC = 7
1170  DVM =722
1200  SOURCE =717
1230  CALL IORESET (ISC)
1240  IF PCIB.ERR <> THEN ERROR PCIB.BASERR
1250  TIMEOUT = 5
1260  CALL IOTIMEOUT (ISC, TIMEOUT)
1270  IF PCIB.ERR <> NOERR THEN ERROR PCIB.BASERR
1280  CALL IOCLEAR (ISC)
1290  IF PCIB.ERR <> NOERR THEN ERROR PCIB.BASERR
```

1150	Set the interface select code variable ISC to 7.
1170	Set the voltmeter address variable DVM to 722.
1200	Set the signal source address variable SOURCE to 717. This example assumes select code 7, voltmeter address 22, and source address 17.
1230	Set the interface to its default configuration.
1250/1260	Define a system timeout of 5 seconds.
1280	Perform IOCLEAR to put all instruments into a known, device-dependent state.

Note *This program includes an error checking line after each Library command:*

IF PCIB.ERR < > NOERR THEN ERROR PCIB.BASERR

If there is an error (i.e., the error variable PCIB.ERR does not equal NOERR), this line calls the SETUP.BAS error handling code. This code prints error information and stops the program. The BASIC Error Handling section later in this chapter discusses error detecting and handling in more detail.

Program the instruments:

```
1350 CODES$ = "RF2 FU1 ST1KH SP10KH MF1KH AM2VR TI5SE"
1360 LENGTH = LEN (CODES$)
1370 CALL IOOUTPUTS (SOURCE,CODES$,LENGTH)
1380 IF PCIB.ERR < > NOERR THEN ERROR PCIB. BASERR
1410 '
1420 CODES$ = "H SM004 F2 R4 FL0 Z0 4STG 20STN RS1 T4"
1430 LENGTH = LEN(CODES$)
1440 CALL IOOUTPUTS (DVM,CODES$,LENGTH)
1450 IF PCIB.ERR < > NOERR THEN ERROR PCIB.BASERR
```

1350 Define the programming codes string CODES$ to hold:

RF2	- select the rear panel signal output.
FU1	- select the sine wave function.
ST1KH	- select a starting frequency of 1 kHz.
SP10KH	- select a stopping frequency of 10 kHz.
MF1KH	- select a marker frequency of 1 kHz.
AM2VR	- select an amplitude of 2 Vrms.
TI5SE	- select a sweep time of 5 seconds.

1360/1370 Use the IOOUTPUTS command to send the programming codes to the source with the proper length parameter.

1420 Define the programming codes string CODES$ to hold:

H	- software-reset the voltmeter.
SM004	- set the service request mask to enable the voltmeter to set the interface SRQ line when it finishes taking readings (when the Data Ready bit of the serial poll response byte is set).
F2	- select the AC volts function.
R4	- select the 10 volt range.
FL0	- turn off filtering.
Z0	- turn off auto zero.
4STG	- select the 4-digit display.
20STN	- take 20 readings.
RS1	- turn on reading storage.
T4	- select trigger hold.

1430/1440 Send the programming codes to the voltmeter with the proper length parameter.

Trigger the instruments:

```
1490 CALL IOTRIGGER(DVM)
1500 IF PCIB.ERR <> NOERR THEN ERROR PCIB.BASERR
1510 CODES$ = "SS"
1520 LENGTH = LEN(CODES$)
1530 CALL IOOUTPUTS (SOURCE,CODES$,LENGTH)
1540 IF PCIB.ERR <> NOERR THEN ERROR PCIB.BASERR
```

1490 Execute the IOTRIGGER command for the voltmeter.

1510 Define the programming codes string to hold the source's trigger code.

1520/1530 Send the programming codes to the source with the proper length parameter.

These lines demonstrate that some instruments respond to an HP-IB trigger command, while others must be triggered with instrument-specific programming codes.

Wait for the voltmeter to finish reading:

```
1580 SRQ = 1
1590 CALL IOSTATUS (ISC, SRQ, STATUS)
1595 IF PCIB.ERR <> NOERR THEN ERROR PCIB.BASERR
1600 IF STATUS = 0 THEN GOTO 1590
1610 CALL IOSPOLL (DVM, RESPONSE)
1615 IF PCIB.ERR <> NOERR THEN ERROR PCIB.BASERR
1620 IF (RESPONSE AND 68) <> 68 THEN GOTO 1590
```

1580 Define the interface condition whose status is being checked. In this case, check for condition 1 - is the SRQ line set?

1590 Execute the status command and return the result in STATUS.

1600/1610 As long as STATUS is 0, the SRQ line is not set, indicating that the voltmeter is not finished taking readings. As soon as it changes to 1, perform a serial poll on the voltmeter to learn which of its conditions, if any, set the SRQ line. The serial poll also clears the SRQ.

1620 The result of the serial poll is the status byte of the voltmeter, returned in RESPONSE. Compare RESPONSE and the value 68 - the sum of the Request Service bit (64) and the Data Ready bit (4). If these bits are set, continue because the voltmeter is finished. If they are not set, perform the status check again.

Enter the readings into an array and print them:

```
1710  CODES$ = "SO1 -20STR RER"
1720  LENGTH = LEN(CODES$)
1730  CALL IOOUTPUTS (DVM,CODES$,LENGTH)
1740  IF PCIB.ERR <> NOERR THEN ERROR PCIB.BASERR
1741  '
1745  STATE = 0
1750  CALL IOEOI (ISC,STATE)
1760  IF PCIB.ERR <> NOERR THEN ERROR PCIB.BASERR
1770  CALL IOENTERA (DVM, READINGS(1),MAX.ELEMENTS,ACT.ELEMENTS)
1780  IF PCIB.ERR <> NOERR THEN ERROR PCIB.BASERR
1810  '
1820  PRINT "THE READINGS ARE: "
1830  FOR I = 1 TO ACT.ELEMENTS
1840  PRINT I, READINGS(I)
1850  NEXT I
1860  '
1870  END
```

1710 Define the programming codes string to direct the voltmeter to output its stored readings:

SO1	- turn on system output mode.
-20STR	- un-store the 20 readings from register R.
RER	- recall (readout) the 20 readings.

1720/1730 Output the programming codes to the voltmeter with the proper length parameter.

1745/1750 Disable the EOI mode so reading won't terminate after entering only one value.

1770 Enter the voltmeter readings into the READINGS array starting at the first element. Include the maximum and actual length parameters.

1820/1870 Print the readings and end the program.

Saving the BASIC Program

1. When you have finished writing the program, press <CTRL-C> to end auto line numbering.

2. Then, save the program as an executable BASIC file in ASCII:

```
SAVE "PROGRAM",A
```

Your program, with the SETUP.BAS material, now resides as an ASCII file in the Library directory as "PROGRAM.BAS". Saving the file in ASCII format (the "A" parameter) allows you to perform a subsequent MERGE.

Running the BASIC Program

When you are ready to run the program, connect the SIGNAL output of the source to the VOLTS input of the DVM. (Include a 50 ohm load in this line to insure proper readings.) Use HP-IB cables to connect the instruments to your computer.

To execute your program from BASIC, load it and type RUN.

Watch the display on the function generator. You see the various functions (sine wave, AC volts, sweep time) displayed as they are programmed.

The voltmeter displays its operation as well - you can watch it take readings, store them and output them to the READINGS array.

As the program ends, it displays the readings on your screen.

BASIC Error Handling

During BASIC program execution, errors may occur as a result of I/O operations. For example, a device may not respond, a timeout may occur, or you might specify an incorrect select code or some invalid parameter.

Since there is no built-in error reporting procedure for Library commands, you should include error handling code in your application program. Otherwise, errors go untreated and may result in unpredictable results. To help you provide the necessary code, this section explains error handling in general for BASIC programming.

Error Message Mnemonics

As an aid to error testing, each error has an associated mnemonic variable. The mnemonics are declared in the BASIC "SETUP.BAS" file.

You can use the mnemonic variables to test for the occurrence of errors. For example, error 0 - no error occurred - has the mnemonic NOERR. In BASIC, "IF PCIB.ERR = NOERR THEN GOTO 1420" would cause program execution to continue at line 1420 if no error occurred.

For a complete list of the BASIC Library errors and their mnemonics, see Table 2-2 at the end of this chapter.

Error Variables

To assist you in detecting and handling errors, the Library provides several error status variables:

PCIB.ERR — This is a return status variable indicating whether an error was detected in the last Library CALL statement. It is good practice to check the value of PCIB.ERR after every CALL statement. If PCIB.ERR = 0, the command terminated without error.

PCIB.ERR$ — When PCIB.ERR is nonzero, this string variable contains an error message corresponding to the value of PCIB.ERR.

PCIB.GLBERR — This variable indicates if an error occurred *anywhere* in the program. If PCIB.GLBERR = 0, all preceding Library calls completed without error. Otherwise, PCIB.GLBERR contains the most recent error value. To locate the exact statement that caused the error, you must perform more detailed error checking.

These three status variables reflect some PC Instruments errors (numbered 1 to 29) and some HP-IB-specific errors (100000 to 100008). They do not, however, reflect spelling errors or parameter passing errors such as passing the wrong number or type of parameters in the CALL statement.

Caution

Spelling errors and parameter passing errors are not reflected by the error status variables. If the wrong number or type of parameters are passed in a CALL statement, the resulting error may require the system to be restarted. Since your program may contain such errors, it is a good idea to always save your program before you run it.

Error Reporting

You should check for errors after each Library command. Since a program normally has more than one CALL statement, you can write a single error handling routine to use after all CALLs. SETUP.BAS provides such a routine in lines 400-445:

```
400 'Error handling routine
405 '
410 IF ERR=PCIB.BASERR THEN GOTO 425
415 PRINT "BASIC error #";ERR;" occurred in line ";ERL
420 STOP
425 TMPERR = PCIB.ERR
430 IF TMPERR = 0 THEN TMPERR = PCIB.GLBERR
435 PRINT "PC Instrument error #";TMPERR;" detected at line ";ERL
440 PRINT "Error: ";PCIB.ERR$
445 STOP
```

This error handling routine is based on the BASIC function ERROR and its associated variables ERR and ERL. (If you need more details about ERROR, ERR and ERL, refer to your BASIC manual.) The variable PCIB.BASERR provides a mechanism for differentiating between a BASIC error and an HP-IB Command Library error. If ERR = 255, the error is generated by the I/O library; otherwise it is a BASIC system error.

The above routine prints the type of error (BASIC or Instrument), the error number, the error message; and the line on which the error was detected. Then, it stops the program. An ON ERROR statement — line 250 of SETUP.BAS — defines a branch to the error handling routine for any BASIC or program-initiated call to the ERROR function.

To invoke the ON ERROR branching if an error is detected, follow each Library command with:

```
IF PCIB.ERR <> NOERR THEN ERROR PCIB.BASERR
```

NOERR is the mnemonic variable that corresponds to the HP-IB error status value indicating that no error occurred. (Other error message mnemonics are listed later in this chapter.)

For example, you may want to create an error handling routine that treats various errors (e.g. timeout, invalid select code) differently, or that resumes program execution after an error is logged or acknowledged. Whether you use the error handling routine provided or write your own, you will save time during program testing if you check for errors after each Command Library CALL.

Table 2-2 at the end of this chapter lists the BASIC Library errors.

Interpretive BASIC Syntax Reference

This section presents syntax descriptions of the Library commands as they are used with Interpretive BASIC. Several preliminary definitions are given, followed by a syntax description of each command.

Parameter Passing

In BASIC, all parameters used in CALL statements must be passed by reference. That is, variable names must appear as parameters; literals or expressions are not permitted. For example, statements such as:

```
1050  ISC = 7
1060  CALL IOCLEAR (ISC)
```

are valid, but:

```
1050  CALL IOCLEAR (7)
```

is not.

Three types of variables are used to describe parameters to Library command calls in the syntax reference.

Numeric Variable

This is a single precision real number variable. It is distinguished from other identifiers either by "!" appended to the variable name, or by no suffix appended at all. The valid range for numeric variables is approximately 2.9E-39 to 1.7E+38 (negative or positive). Single precision real numbers have approximately seven digits of accuracy. Note that integers and double precision real numbers may *not* be used as parameters. Examples:

Valid Numeric Variables:	ISC DEVICE.ADDRESS READING!
Invalid Numeric Variables:	REASON% A$ AREA#

String Variable

This is a variable that can contain any valid sequence of 0 or more ASCII characters. It is identified by a "$" appended to its identifier. The length of a string variable is not fixed, but may be anywhere from 0 (the null string) to 255. String variables should be initialized with SPACE$ before data is entered with the IOENTERS command. Otherwise, data or program code may be corrupted.

Numeric Array

This is an array of single-precision real numbers. Arrays are declared using the DIM statement. Although the theoretical maximum size for a BASIC array is 32,767, the actual limit is approximately 14,000 elements depending on the length of your program.

Interpretive BASIC does not permit the use of array names as parameters to CALL statements. Therefore, when you use arrays in Library function calls, use the first element of the array to be accessed as the parameter. Example:

```
1050 CALL IOOUTPUTA (DEVICE, VALUES(0), ELEMENTS)
```

This indicates that ELEMENTS values are to be output starting at element 0 of the VALUES array. This line is also valid:

```
1050 CALL IOOUTPUTA (DEVICE, VALUES(5), ELEMENTS)
```

It indicates that values are to be output beginning with the value in array element 5 and proceeding from there.

Be sure the number of elements parameter value does not exceed the number of elements available in the array. In the case of data output, the Library continues through memory sending the contents of the individual cells until the specified count is satisfied. For data input, the results are similar, with the potential hazard of overwriting existing data or code.

Command Addressing

The first parameter of each Library command specifies an interface select code or a device bus address. Some commands allow only a select code (e.g., IOABORT); some permit only a device address (e.g., IOSPOLL); and some permit either (e.g., IOCLEAR).

If you specify a select code, the command is directed to the interface, and no bus addressing is done prior to the transfer of data or commands. If you specify a device address, devices on the bus are first addressed using the sequence:

- UNLISTEN
- TALK or LISTEN ADDRESS of the System Controller
- LISTEN or TALK ADDRESS of the Target Device

The addressing is followed by data or bus command transfer, depending on the command.

A select code is differentiated from a device address by the magnitude of the value specified. The interface treats a value in the range 0 to 99 as a select code. Only 1 through 16 are valid select codes, however, and there is only one valid select code for each HP-IB in your computer. If you specify a non-existent select code, an error results.

A device address is composed of the select code and the primary bus address of a device. It is calculated as:

(select code * 100) + primary bus address

A valid primary bus address is in the range 0 to 30. Address 30 is reserved as the address of the controller.

Secondary Addressing

You can use extended talker and listener functions (secondary addressing) with the Library. To specify an extended address, use the formula:

(select code * 10000) + (primary bus address * 100) + secondary address

A secondary address may be in the range 0 to 31. You can use extended addressing with any command in which the first parameter may be a device address.

As an example, consider a system with select code 7 that has a device with primary address 9 and secondary address 15. The first parameter of a Library call could be:

```
1000  ISC = 7
1010  CALL IOCLEAR (ISC)          ' Select Code Only
1020  DEVICE = 709
1030  CALL IOCLEAR (DEVICE)       ' Primary Address
1040  DEVICE = 70915
1050  CALL IOCLEAR (DEVICE)       ' Secondary Address
```

HP-IB Mnemonics

Throughout the syntax description, HP-IB terms are listed by abbreviation rather than by name. For example, Go To Local is listed as GTL. For your convenience, the mnemonics are summarized in Table 2-1.

MNEMONIC	DEFINITION
ATN	Attention
DCL	Device Clear
EOI	End Or Identify
EOL	End Of Line
GET	Group Execute Trigger
GTL	Go To Local
IFC	Interface Clear
LAD	Listen Address
LLO	Local Lockout
MLA	My Listen Address
MTA	My Talk Address
OSA	Other Secondary Address
PPC	Parallel Poll Configure
PPD	Parallel Poll Disable
PPU	Parallel Poll Unconfigure
REN	Remote Enable
SDC	Selected Device Clear
SPD	Serial Poll Disable
SPE	Serial Poll Enable
SRQ	Service Request
TAD	Talk Address
UNL	Unlisten
UNT	Untalk

Table 2-1. HP-IB Mnemonics

Command Syntax

The following pages present a detailed syntax description of the Library commands as they are used in Interpretive BASIC.

IOABORT

This command aborts all activity on the interface.

Syntax

IOABORT*(select code)*

select code - a numeric variable specifying the interface select code.

Examples

```
1100 ISC = 7
1110 CALL IOABORT (ISC)
```

Bus Activity

- IFC is pulsed (> 100 uS).
- REN is set.
- ATN is cleared

Comments

Devices in Local Lockout will remain locked out.

Possible errors are NOERR, ETIME and ESEL.

IOCLEAR

This command returns a device to a known, device-dependent state. It can be addressed to the interface or to a specific device.

Syntax

IOCLEAR *(device address)*
IOCLEAR *(select code)*

device address - a numeric variable specifying the address of a device to be cleared.

select code - a numeric variable specifying the select code of the interface on which all devices are to be cleared.

Examples

```
1100 ISC = 7
1110 DVM = 723
1120 CALL IOCLEAR (DVM) 'Clear the device at address 23.
   .
   .
1150 CALL IOCLEAR (ISC) 'Clear all devices on the interface.
```

Bus Activity

If a device address is specified:

- ATN is set.
- UNL is sent.
- MTA is sent.
- LAD is sent.
- OSA is sent if specified.
- SDC is sent.

If a select code is specified:

- ATN is set.
- DCL is sent.

Comments

Possible errors are NOERR, ETIME and ESEL.

IOCONTROL

This command directly sets status conditions in the interface. It can be used to address/unaddress the interface as a talker or listener, or set the interface's bus address.

Note

IOCONTROL should be used with caution since it operates directly on the interface. For example, if the interface's device address is changed, some operating system code - such as printer or disc drivers - may not function properly.

Syntax

IOCONTROL *(select code,condition,status)*

select code - a numeric variable specifying the interface select code.

condition - a numeric variable specifying the status condition which is to be set. Conditions which can be set are:

Value	Description
5	- address or unaddress the interface as talker.
6	- address or unaddress the interface as listener.
7	- set the interface's bus address.

status - a numeric variable specifying zero (clear) or a non-zero value (set) for conditions 5 and 6; or a bus address (0-30) for condition 7.

Examples

```
1100 ISC = 7
1110 COND = 5
1120 STATUS = 1
1130 CALL IOCONTROL (ISC,COND,STATUS) 'Address the interface as the
                                       talker.
```

Bus Activity

None.

Comments

Possible errors are NOERR, ESEL and ERANGE.

IOENTER

This command reads a single real number.

Reading continues until:

- the EOI line is sensed true (if enabled).
- a linefeed is encountered after the number starts.

Numeric characters include digit "E", "+", "-" and "." in the proper sequence for representing a number. ""(space) is not a numeric character.

Syntax

IOENTER *(device address,data)*
IOENTER *(select code,data)*

device address	- a numeric variable specifying a device address.
select code	- a numeric variable specifying the interface select code.
data	- a numeric variable into which the reading is placed.

Examples

```
1100 DEVICE = 722
1110 CALL IOENTER (DEVICE,READING) 'Input a number from device 722;
                                    place it in READING.
```

Bus Activity

If a device address is specified:

- ATN is set.
- UNL is sent.
- MLA is sent.
- TAD is sent.
- OSA is sent if specified.
- ATN is cleared.
- Data is entered.

If a select code is specified:

- If the interface is not addressed to listen, an error results. Otherwise, ATN is cleared and the data is read from the interface.

Comments

If a select code is to be specified in the command, the interface must first be addressed to listen (with IOSEND or a previous IOENTER, for example) or an error occurs.

Possible errors are NOERR, ETIME, ESEL, EADDR and ENUM.

IOENTERA

This command enters numbers from a device or from the interface and places them in a numeric array.

Reading continues until:

- the EOI line is sensed true (if enabled).
- a linefeed is encountered after the specified number of elements is received.

Numeric characters include digit "E", "+", "-" and "." in the proper sequence for representing a number. "'(space) is not a numeric character.

Syntax

IOENTERA *(device address,readings,max.elements, actual.elements)*
IOENTERA *(select code,readings,max.elements,actual.elements)*

device address	- a numeric variable specifying a device address.
select code	- a numeric variable specifying the interface select code.
readings	- a numeric array into which the readings are placed.
max.elements	- an numeric variable specifying the maximum number of elements to be read.
actual.elements	- an numeric variable specifying the number of elements actually read.

Examples

```
1100 DIM READINGS(49)
1110 DEVICE = 723
1120 MAX.ELEMENTS = 50:ACTUAL.ELEMENTS=0
1130 CALL IOENTERA (DEVICE,READINGS(0),MAX.ELEMENTS,
     ACTUAL.ELEMENTS) 'Read a maximum of 50 values from device 723; put
                      them in READINGS.
```

Bus Activity

If a device address is specified:

- ATN is set.
- UNL is sent.
- MLA is sent.
- TAD is sent.
- OSA is sent if specified.
- ATN is cleared.
- Data is entered.

If a select code is specified:

- If the interface is not addressed to listen, an error results. Otherwise, ATN is cleared and the data is read.

Comments

You must initialize the *act.elements* parameter before you use it in an IOENTERA command. Failure to do so may result in a BASIC syntax error.

Non-numeric characters which do not properly belong in a real number are considered value separators. Thus, the sequence "1,234,567" will be entered as three numbers, not as "1234567".

The number of readings available is dependent upon the device.

If a select code is to be used in the command, the interface must first be addressed to listen (with IOSEND, for example) or an error occurs.

Possible errors are NOERR, ETIME, ESEL, EADDR and ENUM.

IOENTERS

This command enters a string from a device or from the interface.

Reading continues until:

- the EOI line is sensed true (if enabled).
- the termination character set by IOMATCH is received (linefeed is the default).
- the number of characters specified by the *max.length* parameter is received.

Syntax

IOENTERS *(device address,data,max.length,actual.length)*
IOENTERS *(select code,data,max.length,actual.length)*

device address	- a numeric variable specifying a device address.
select code	- a numeric variable specifying the interface select code.
data	- a string variable into which the read string is placed.
max.length	- a numeric variable specifying the maximum number of bytes to be read.
actual.length	- a numeric variable specifying the number of bytes actually read.

Note: If the dimensioned length of the *data* string is less than the *max.length* parameter value, then the *data* string will remain unchanged by the command, but input bytes will be read.

Examples

```
1100 DEV = 723
1110 MAX.LENGTH = 10:ACTUAL.LENGTH = 0
1120 INFO$ = SPACE$(MAX.LENGTH)
1130 CALL IOENTERS (DEV,INFO$,MAX.LENGTH,ACTUAL.LENGTH)
     'Read a string of 10 characters maximum from device 723; place it in INFO$.
1140 INFO$ = LEFT$(INFO$,ACTUAL.LENGTH)
```

Bus Activity

If a device address is specified:

- ATN is set.
- UNL is sent.
- MLA is sent.
- TAD is sent.
- OSA is sent if specified.
- ATN is cleared.
- Data is entered.

If a select code is specified:

- If the interface is not addressed to listen, an error results. Otherwise, ATN is cleared and the data is read.

Comments

If a select code is to be specified in the command, the interface must first be addressed to listen (with IOSEND or a previous IOENTER, for example) or an error occurs.

You should initialize the string into which data is read with the SPACE$ function before you call IOENTERS:

```
1150 INFO$ = SPACE$(10)
```

This prevents corruption of data if the string was previously uninitialized, or was set to some literal string.

The termination character is entered as part of the string. To remove it, use the BASIC LEFT$ function. For example, to remove CR/LF from the end of the string, you could enter:

```
1260 INFO$ = LEFT$(INFO$,ACT.LENGTH-2)
```

Possible errors are NOERR, ETIME, ESEL, EADDR and ENUM.

IOEOI

This command enables or disables the End Or Identify (EOI) mode of the interface. It is used to:

- enable or disable a write operation to set the EOI line on the last byte of the write.
- enable or disable a read operation to terminate upon sensing the EOI line true.

The default is EOI enabled.

Syntax

IOEOI *(select code,state)*

select code	- a numeric variable specifying the interface select code.
state	- a numeric variable; a non-zero value enables EOI and zero disables it.

Examples

```
1100 ISC = 7
1110 STATE = 0
1120 CALL IOEOI (ISC,STATE) 'Disable EOI.
```

Bus Activity

None.

Comments

When reading with EOI enabled, receipt of a byte with EOI set causes the I/O operation to terminate, regardless of whether you are reading a string, a real number, or an array of real numbers.

When writing, EOI is set on the last byte of the End Of Line sequence if EOI is enabled. Note that if the EOL sequence is zero length, EOI is set on the last data byte sent.

When sending real number arrays, the EOL sequence (and subsequent EOI) is appended after the last element in the array, not after each element.

Note that IOSEND does not set EOI since this line has a different meaning in Command mode.

Possible errors are NOERR and ESEL.

IOEOL

This command defines the End Of Line (EOL) string that is to be sent following every IOOUTPUT, IOOUTPUTA, or IOOUTPUTS command. Default is carriage return/linefeed.

Syntax

IOEOL *(select code,endline,length)*

select code	- a numeric variable specifying the interface select code.
endline	- a string variable specifying the EOL string that is to be sent following a data transmission. A maximum of 8 characters may be specified.
length	- a numeric variable specifying the length of the EOL string. If zero is specified, no characters are appended to a data transmission. If *length* is less than zero, an error occurs.

Examples

```
1100 ISC = 7
1110 ENDLINE$ = CHR$(13)+CHR$(10)
1120 LENGTH = LEN(ENDLINE$)
1130 CALL IOEOL (ISC,ENDLINE$,LENGTH) 'EOL =CR/LF.
```

Bus Activity

None.

Comments

With IOOUTPUTA, the EOL sequence is appended after *all* the array elements have been sent, not following each element.

Possible errors are NOERR, ESEL and ERANGE.

IOGETTERM

This command determines the reason for which the last read terminated.

Syntax

IOGETTERM *(select code,reason)*

select code - a numeric variable specifying the interface select code.

reason - a numeric variable to receive the logical OR-ing of the reasons for which the last read operation terminated. The possible reasons for termination are:

Value	Description
0	- the read was terminated for some reason not covered by any of the other reasons (e.g., error or timeout).
1	- the expected number of elements was received.
2	- the termination character set by IOMATCH was encountered.
4	- the EOI line was sensed true.

Examples

```
1100 ISC = 7
1110 CALL IOGETTERM (ISC,TERM.REASON)
1120 IF ((TERM.REASON AND 4) = 4) THEN PRINT "EOI ENCOUNTERED"
```

Bus Activity

None.

Comments

Upon return, the *reason* integer contains the logical OR-ing of the reasons for termination. For example, if the last read terminated when the termination character was encountered, and EOI was set, the value of *reason* would be 2 + 4 = 6.

Possible errors are NOERR, and ESEL.

IOLLOCKOUT

This command executes a Local Lockout (LLO) to disable a device front panel. It is received by all devices on the bus, whether or not they are addressed to listen.

Syntax

IOLLOCKOUT *(select code)*

select code - a numeric variable specifying the interface select code.

Examples

```
1100 ISC = 7
1110 CALL IOLLOCKOUT (ISC)
```

Bus Activity

- ATN is set.
- LLO is sent.

Comments

If a device is in Local mode when LLO is received, LLO does not take effect until the device becomes addressed to listen.

If the REN line is not set, this command has no effect.

Possible errors are NOERR, ETIME and ESEL.

IOLOCAL

This command executes a Go To Local (GTL) or clears the REN line to enable a device front panel.

Syntax

IOLOCAL *(device address)*
IOLOCAL *(select code)*

device address	- a numeric variable specifying a device address.
select code	- a numeric variable specifying the interface select code.

Examples

```
1100 DEVICE = 722
1110 CALL IOLOCAL (DEVICE) 'Place device 722 in Local mode.
```

Bus Activity

If a device address is specified:

- ATN is set.
- UNL is sent.
- MTA is sent.
- LAD is sent.
- OSA is sent if specified.
- GTL is sent.

If a select code is specified:

- REN is cleared.
- ATN is cleared.

Comments

If a device address is specified, the device is temporarily placed in Local mode - it will return to Remote mode if it is later addressed to listen. If Local Lockout is in effect, the device will return to the lockout state if it is later addressed to listen.

If an interface select code is specified, all instruments on the bus are placed in Local mode, and any Local Lockout is cancelled.

Possible errors are NOERR, ETIME and ESEL.

IOMATCH

This command establishes the character upon which an IOENTERS operation terminates. The character becomes part of the entered string when received and is initially enabled and set to linefeed.

Syntax

IOMATCH *(select code,character,flag)*

select code	- a numeric variable specifying the interface select code.
character	- a string variable specifying the character upon which an IOENTERS terminates.
flag	- a numeric variable indicating whether character matching should be enabled or disabled. Zero disables it and any non-zero value enables it.

Examples

```
1100 ISC = 7
1110 MATCHS = CHR$(10) 'Terminate IOENTERS on linefeed.
1120 FLAG = 1
1130 CALL IOMATCH (ISC,MATCH$,FLAG)
```

Bus Activity

None.

Comments

Only a single match character may be specified in this command.

IOMATCH does not apply to IOENTER or IOENTERA.

Possible errors are NOERR and ESEL.

IOOUTPUT

This command outputs a real number to a device or to the interface. After the number is sent, the EOL string is sent and the EOI line is set (if enabled).

Syntax

IOOUTPUT *(device address,data)*
IOOUTPUT *(select code,data)*

device address	- a numeric variable specifying a device address.
select code	- a numeric variable specifying the interface select code.
data	- a numeric variable specifying the real number to be output.

Examples

```
1100 INFO = 12.3
1110 DEV = 722
1120 CALL IOOUTPUT (DEV,INFO) 'Output "12.3" to device 722.
```

Bus Activity

If a device address is specified:

- ATN is set.
- UNL is sent.
- MTA is sent.
- LAD is sent.
- OSA is sent if specified.
- ATN is cleared
- Data is output.
- EOL is output.

If a select code is specified:

- If the interface is not addressed to talk, an error results. Otherwise, ATN is cleared and the data is sent followed by EOL.

Comments

Numbers with absolute values between 1E-5 and 1E+6 are rounded to seven significant digits and output in floating point notation. If the number rounds to an integer value, the decimal point is not sent. Numbers outside this range are rounded to seven significant digits and output in scientific notation.

If the number is positive, leading space is output for the sign; if negative, a leading "-" is output.

If a select code is to be specified in the command, the interface must first be addressed to talk (using IOSEND, for example), or an error occurs.

Possible errors are NOERR, ETIME, ESEL and EADDR.

IOOUTPUTA

This command outputs an array of real numbers to a specified device or to the interface. Values output are separated by commas. After the last byte of data is sent, the EOL string is sent and the EOI line is set (if enabled).

Syntax

IOOUTPUTA *(device address,data,elements)*
IOOUTPUTA *(select code,data,elements)*

device address	- a numeric variable specifying a device address.
select code	- a numeric variable specifying the interface select code.
data	- a numeric array containing the array of real numbers to be transmitted.
elements	- a numeric variable specifying the number of elements in the array to be transmitted.

Examples

```
1100 DIM INFO(9)
1110 ELEMENTS = 10
1120 DEV = 722
1130 CALL IOOUTPUTA (DEV,INFO(0),ELEMENTS) 'Output array INFO to device
                                            722; begin with element 0.
```

Bus Activity

If a device address is specified:

- ATN is set.
- UNL is sent.
- MTA is sent.
- LAD is sent.
- OSA is sent if specified.
- ATN is cleared.
- Data is output.
- EOL is output.

If a select code is specified:

- If the interface is not addressed to talk, an error results. Otherwise, ATN is cleared and the data is sent followed by EOL.

Comments

Numbers between 1E-5 and 1E+6 are rounded to seven significant digits and output in floating point notation. If the number rounds to an integer value, the decimal point is not sent. Numbers outside of this range are rounded to seven significant digits and output in scientific notation.

If the number is positive, a leading space is output for the sign; if negative, a leading "-" is output.

If a select code is to be used as a parameter, the interface must first be addressed to talk (with IOSEND, for example), or an error occurs.

Possible errors are NOERR, ETIME, ESEL and EADDR.

IOOUTPUTS

This command outputs a string to a specified device or to the bus. After the string is sent, the EOL string is sent and the EOI line is set (if enabled).

Syntax

IOOUTPUTS *(device address,data,length)*
IOOUTPUTS *(select code,data,length)*

device address	- a numeric variable specifying a device address.
select code	- a numeric variable specifying the interface select code.
data	- a string variable specifying the string to be sent.
length	- a numeric variable specifying the length of the data string.

Examples

```
1100 DEV = 723
1110 INFO$ = "1ST1"
1120 LENGTH = LEN(INFO$)
1130 CALL IOOUTPUTS (DEV,INFO$,LENGTH) 'Send "1ST1" to device 723.
```

Bus Activity

If a device address is specified:

- ATN is set.
- UNL is sent.
- MTA is sent.
- LAD is sent.
- OSA is sent if specified.
- ATN is cleared
- Data is output.
- EOL is output.

If a select code is specified:

- If the interface is not addressed to talk, an error results. Otherwise, ATN is cleared and the data is sent followed by EOL.

Comments

If a select code is to be used in the command, the interface must first be addressed to talk (with IOSEND, for example), or an error occurs.

Possible errors are NOERR, ETIME, ESEL and EADDR.

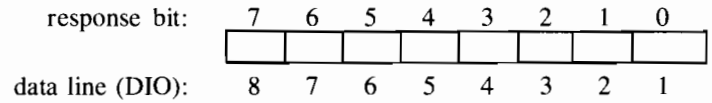
IOPPOLL

This command performs a parallel poll of the interface. It sets a variable to a value (0 - 225) representing the response of those instruments on the interface that respond to a parallel poll.

Syntax

IOPPOLL *(select code,response)*

response - a numeric variable into which the parallel poll response byte is placed. The allowable range is 0 to 255. The eight bits of the response byte correspond to the eight data lines:



Hence, a value of 32 would indicate that a device(s) has responded to the parallel poll with a '1' on DIO 6.

Examples

```
1100 ISC = 7
1110 CALL IOPPOLL (ISC,RESPONSE)
```

Bus Activity

- ATN and EOI are asserted for 25 microseconds.
- The poll byte is read.
- EOI is cleared.
- ATN is restored to its previous state.

Comments

Not all devices are capable of responding to a parallel poll. Consult your particular device manuals for specifics.

Possible errors are NOERR and ESEL.

IOPPOLLC

This command performs a Parallel Poll Configure. In preparation for a parallel poll command, it tells an instrument how to respond affirmatively to the parallel poll, and on which data line to respond.

In general, it sets a parallel poll response byte to reflect the response of a desired arrangement of instruments. Typically, you could define the bits to reflect the responses of particular instruments, or the logical-ORing of several instrument responses.

(Refer to IOPPOLL for more information.)

Syntax

IOPPOLLC *(device address,configuration)*
IOPPOLLC *(select code,configuration)*

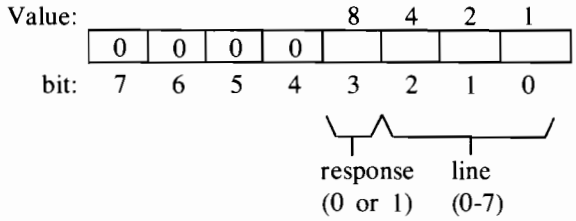
device address - a numeric variable specifying the address of the device to be configured.

select code - a numeric variable specifying the interface select code.

configuration - a numeric variable sent to the specified device indicating:

- how to respond: 1 or 0.
- on which data line to respond: 0 to 7 (corresponding to DIO 1-8).

The configuration byte represented by the value is:



Bits 0-2 specify the data line (0 - 7). Bit 3 specifies the affirmative response (1 or 0). The allowable *configuration* range is 0 - 15.

Hence, to indicate an affirmative response of 1 on line 4 (DIO 5), the configuration value would be 8 + 4 = 12.

Examples

```
1100 DEVICE = 723
1110 CONF = 10 'Respond with a "1" on line 2.
1120 CALL IOPPOLLC (DEVICE,CONF)
```

Bus Activity

If a device address is specified:

- ATN is set.
- UNL is sent.
- MTA is sent.
- LAD is sent.
- OSA is sent if specified.
- PPC is sent.
- PPE is sent.

If a select code is specified:

- PPC is sent.
- PPE is sent.

Comments

Not all devices can be configured to respond to a parallel poll. Consult your particular device manuals for specifics.

Possible errors are NOERR, ETIME, ESEL and ERANGE.

IOPPOLLU

This command performs a Parallel Poll Unconfigure - it directs an instrument to *not* respond to a parallel poll. It can be addressed to the interface or to a specific device. (Refer to IOPPOLLC for more information.)

Syntax

IOPPOLLU *(device address)*
IOPPOLLU *(select code)*

device address	- a numeric variable specifying a device address.
select code	- a numeric variable specifying the interface select code.

Examples

```
1100 DEV = 722
1110 CALL IOPPOLLU (DEV)
```

Bus Activity

If a device address is specified:

- ATN is set.
- UNL is sent.
- MTA is sent.
- LAD is sent.
- OSA is sent if specified
- PPC is sent.
- PPD is sent.

If a select code is specified:

- ATN is set.
- PPU is sent.

Comments

Some devices cannot be unconfigured from the bus. Consult your particular device manuals for specifics.

Possible errors are NOERR, ETIME and ESEL.

IOREMOTE

This command places a device in Remote mode. It can be addressed to a specific device, or to the interface.

Syntax

IOREMOTE *(device address)*
IOREMOTE *(select code)*

device address	- a numeric variable specifying a device address.
select code	- a numeric variable specifying the interface select code.

Examples

```
1100 ISC = 7
1110 DVM = 723
1110 CALL IOREMOTE (DVM) 'Place the DVM in Remote mode.
   :
1140 CALL IOREMOTE (ISC) 'Set the interface REN line.
```

Bus Activity

If a device address is specified:

- REN is set.
- ATN is set.
- UNL is sent.
- MTA is sent.
- LAD is sent.
- OSA is sent if specified.

If a select code is specified:

- REN is set.

Comments

If a select code is specified, a device does not switch into Remote until it is addressed to listen.

Possible errors are NOERR, ETIME, and ESEL.

IORESET

This command resets the interface to its start-up state in which it is system controller, active controller, not listening, and not talking.

In addition, it sets the following HP-IB parameters as indicated:

- The interface timeout is set to 0 seconds (i.e., timeout is disabled).
- The interface EOI mode is enabled.
- CR/LF is set as the EOL default.
- LF is set as the IOMATCH default.

Syntax

IORESET *(select code)*

select code - a numeric variable specifying the interface select code.

Examples

```
1100 ISC = 7
1110 CALL IORESET (ISC)
```

Bus Activity

- IFC is pulsed (>100 uS).
- REN is cleared (>10 uS), then set.
- ATN is cleared.

Comments

IORESET returns all devices on the interface to local (front panel) control.

Possible errors are NOERR, ETIME and ESEL.

IOSEND

This command sends a string of user-specified HP-IB commands to the interface. For example, to send an output command to several instruments simultaneously, you can establish their talk/listen status with the IOSEND command, then issue the command specifying a select code rather than a device address.

Syntax

IOSEND *(select code,commands,length)*

select code	- a numeric variable specifying the interface select code.
commands	- a string variable specifying a string of characters, each of which is to be treated as a bus command.
length	- a numeric variable specifying the number of characters in the command string.

Examples

```
1100 ISC = 7
1110 COMMANDS$ ="?)\4" 'Specifies listen addresses 9, 15 and 20.
1120 LENGTH = 4
1130 CALL IOSEND (ISC,COMMANDS$,LENGTH)
1140 CALL IOTRIGGER (ISC) 'Triggers devices at addresses 9, 15 and 20.
```

Bus Activity

- ATN is set.
- Command bytes are sent.

Comments

All bytes are sent with ATN set. The EOL sequence is not appended, nor is EOI set.

Possible errors are NOERR, ETIME, ESEL and EPASS.

IOSPOLL

This command performs a serial poll of a specified device. It sets a value representing the device's response byte.

Syntax

IOSPOLL *(device address,response)*

device address - a numeric variable specifying the bus address of the device to be polled.

response - a numeric variable into which the response byte is placed.

Examples

```
1100 DEVICE = 723
1110 CALL IOSPOLL (DEVICE,RESPONSE) 'Perform a serial poll on device 723;
                                    put the response byte in RESPONSE.
```

Bus Activity

- ATN is set.
- UNL is sent
- MLA is sent.
- TAD is sent.
- OSA is sent if specified.
- SPE is sent.
- ATN is cleared.
- Poll byte is read.
- ATN is set.
- SPD is sent.
- UNT is sent.

Comments

Some devices are not capable of responding to a serial poll in which case polling may result in an error. Consult the appropriate manual to determine if an instrument can respond, and how its response byte is interpreted.

Possible errors are NOERR, ETIME and ESEL.

IOSTATUS

This command determines the interface status. It sets a variable representing a particular interface status condition.

Syntax

IOSTATUS *(select code,condition,status)*

select code - a numeric variable specifying the interface select code.

condition - a numeric variable from 0 to 7 specifying the condition whose status is being checked. Possible conditions are:

Value	Description
0	Is the interface currently in the Remote state? (always false)
1	What is the current state of the SRQ line?
2	What is the current state of the NDAC line?
3	Is the interface currently system controller? (always true)
4	Is the interface currently active controller? (always true)
5	Is the interface currently addressed as a talker?
6	Is the interface currently addressed as a listener?
7	What is the interface's bus address?

status - a numeric variable into which the condition's status is placed. *status* will be 1 for set or zero for clear; or an address (0 - 30) for condition 7.

Examples

```
1100 ISC = 7
1110 SELECT = 1
1120 CALL IOSTATUS (ISC,SELECT,STATUS) 'Determine if SRQ is set.
```

Bus Activity

None.

Comments

Possible errors are NOERR, ESEL and ERANGE.

IOTIMEOUT

This command sets an interface timeout value in seconds for those cases where an I/O operation does not complete (e.g., the printer runs out of paper). Default is 0 seconds, indicating that timeout is disabled.

Syntax

IOTIMEOUT *(select code,timeout)*

select code	- a numeric variable specifying the interface select code.
timeout	- a numeric variable specifying the length of the timeout period. 0.0 disables the timeout. A negative value is an error.

Examples

```
1100 ISC = 7
1110 TIMEOUT.VAL = 2 'Timeout = 2 seconds.
1120 CALL IOTIMEOUT (ISC,TIMEOUT.VAL)
```

Bus Activity

None.

Comments

Timeout is effective for any interface operation that transfers data or commands.

A timeout error occurs only if timeout is enabled (i.e., set to some value other than zero).

A call to IOTIMEOUT should be one of the first statements in your BASIC program. It provides a way to recover from I/O operations that are otherwise incomplete.

The timeout value is a real number specified in seconds. To timeout after five seconds, use *timeout*=5.0; to timeout after one-half second, use *timeout*=0.5. Note that a timeout of 0.0 effectively disables any timeouts. The maximum timeout allowable is 4,096 seconds.

The resolution of a timeout may vary for different computers. For example, some computers have a timeout resolution of one second. Thus, a timeout resolution cannot be set more accurately than plus or minus 1 second on these computers.

Upon timeout, the error variable PCIB.ERR returns a value of 100004, and the error string PCIB.ERR$ returns the message "HPIB: timeout".

Possible errors are NOERR, ESEL and ERANGE.

IOTRIGGER

This command triggers a device or several devices.

Syntax

IOTRIGGER *(device address)*
IOTRIGGER *(select code)*

device address	- a numeric variable specifying a device address.
select code	- a numeric variable specifying the interface select code.

Examples

```
1100 DEV = 723
1110 CALL ICTRIGGER (DEV)
```

Bus Activity

If a device address is specified:

- ATN is set.
- UNL is sent.
- MTA is sent.
- LAD is sent.
- OSA is sent if specified.
- GET is sent.

If a select code is specified:

- ATN is set.
- GET is sent.

Comments

Only one device can be triggered at a time if a device address is specified.

If a select code is specified, all devices on the bus that are addressed to listen (with IOSEND, for example) are triggered.

Possible errors are NOERR, ETIME, and ESEL.

Table 2-2. lists the BASIC Library errors.

BASIC ERROR	MNEMONIC	DEFINITION
0	NOERR	No error occurred.
100001	EUNKNOWN	HP-IB: Unknown error. An error not covered in this list has occurred. Check for malfunctioning equipment or incorrect hardware configuration. Also, check for invalid lines in your application program.
100002	ESEL	HP-IB: Invalid select code or device address. This error would most likely occur under these conditions: - The first parameter of a call should be a valid select code, but a device address or an invalid select code was specified instead. - A device address was expected, but a select code was given; or a primary address outside the range 00 - 30, or a secondary address outside the range 00 - 31 was specified. - The device address of the interface was specified as a parameter in commands such as IOSPOLL, IOREMOTE and IOCLEAR.
100003	ERANGE	HP-IB: Value out of range. A command parameter was specified outside its allowable range. This error would most likely occur in these commands: - IOSTATUS - The status specified was outside the range 0 - 7. - IOCONTROL - The specified condition must be from 5 to 7. If 7 is selected, the valid address values are 0 - 30. - IOTIMEOUT - The specified timeout value must be greater than or equal to 0; a negative timeout value may not be entered. - IOPPOLLC - The configuration value must be between 0 and 15.

		- IOEOL - The specified length must be between 0 and 8.
100004	ETIME	HP-IB: Timeout. The time specified by IOTIMEOUT has elapsed since the last byte was transferred.
100005	ECTRL	HP-IB: must be controller. The interface must be active controller and/or system controller.
100006	EPASS	HP-IB: Pass control not permitted. Pass control is not supported by the Library.
100007	ENUM	HP-IB: Invalid number. No digit or an improperly formatted number was received during a real number read (IOENTER, IOENTERA). In this case, a value of zero is returned as the reading value.
100008	EADDR	HP-IB: Improper addressing. An attempt was made to input/output data when the interface was not addressed to listen/talk. This error occurs if a select code is specified for a read or write, but the interface is not properly addressed to talk or listen.

Table 2-2. BASIC Library Errors

Table 2-3 lists PC Instruments errors that you may encounter in BASIC programming. If any of these errors occur in your program, consult a PC Instruments manual for more information.

ERROR	DEFINITION
01	Break detected
02	Undefined function — System must be reset
03	PCIBILC.BLD not at paragraph boundary
04	Out of memory
05	File not found
06	Incorrect file format
07	Unable to open the file
08	Unable to close the file
09	End of file while reading
10	Error while reading
11	Disk not ready
12	Primary initialization error
13	"Panels" not loaded
14	Input string too big
15	Invalid value
29	Unknown error

Table 2-3. PC Instruments Errors

3 Pascal Programming

Introduction

This chapter explains how to use the HP-IB Command Library for Pascal programming.

The first section tells you how to copy the necessary Library files to a work disc.

The next section shows you how to write a program using several Library commands. Your HP-IB Command Library disc contains a similar version of this program. You can run it on your system to observe the results.

The last section is a syntax reference for the commands as they are used with Pascal.

Getting Started

To begin programming in Pascal, you must copy the Pascal files from the Library disc to a work disc. This section explains how to do this with both PAM and MS-DOS.

Copying with PAM

These steps assume that B is the source disc and A is the work disc.

1. Enter the File Manager by pressing its softkey or selecting its application from the PAM menu.

2. Create a Library directory (if one does not already exist) on the work disc:

   - Press the Make Dir softkey.
   - Enter A:\LIB in response to the directory name prompt.
   - Press Start Make Dir.
   - Press Exit Make Dir when you're finished.

3. Insert the Library disc in drive B and begin copying the Pascal files to the Library directory:
   - Press the Copy File softkey.
   - Enter B:HPIB.LIB in response to the "from file" prompt.
   - Enter A:\LIB\HPIB.LIB in response to the "to file" prompt.
   - Press the Start Copy softkey.

   Repeat this step, with the appropriate filename, for all Pascal files:

   HPIB.LIB
   IODECL.EX
   IOPROC.EX

   - Press Exit Copy when you're finished.

4. Copy all appropriate MS-Pascal files to your computer. Place them in the root directory. Refer to the MS-Pascal manual for instructions.

5. The Library must later be linked with your Pascal program. Make sure a copy of the Pascal system library is named PASCAL.LIB and is placed in the Library directory.

6. Press the Exit FILE MGR softkey.

Copying with MS-DOS

These steps assume that B is the source disc and A is the work disc.

1. Select the MS-DOS COMMANDS application from PAM.

2. From MS-DOS, select the A disc and create a Library directory on it:

```
> A:
> MKDIR \LIB
```

3. Select the Library directory as the current directory:

```
> CD \LIB
```

4. Insert the Library disc in drive B and begin copying the Pascal Library files to the Library directory:

```
> COPY B:HPIB.LIB
> COPY B:IODECL.EX
> COPY B:IOPROC.EX
```

5. Copy all appropriate MS-Pascal files to your computer. Place them in the root directory. Refer to the MS-Pascal manual for instructions.

6. The Library must later be linked with your Pascal program. Make sure a copy of the Pascal system library is named PASCAL.LIB and is placed in the Library directory.

With these preliminary tasks accomplished, you're ready to begin writing a Pascal program.

Programming in Pascal

For Pascal programming, the Library is implemented as a series of functions that execute the commands. The functions always return a value indicating the error status of the command. In an application program, you would typically use the commands in the following manner to execute an operation:

- Set up the required variables.
- Perform the operation.
- Test to see if the operation completed successfully.

In this example, you follow these steps to program two instruments - an HP3325A Synthesizer/Function Generator and an HP3456A Digital Voltmeter. You program the source to output a 2 Vrms signal, swept from 1 KHz to 10 KHz. You program the DVM to take 20 readings from the signal and output them to an array. Finally, you display the readings on the screen.

Writing a Pascal Program

Use a convenient text editor to write your program.

First, define some preliminary information:

```
PROGRAM example (input, output);

{$INCLUDE: 'IODECL.EX'}

CONST
  isc    = 7;
  source = 717;
  dvm    = 722;

TYPE
  str10 = STRING(10);

VAR
  cmd : INTEGER;

{$INCLUDE: 'IOPROC.EX'}
```

- Include a compiler directive to access IODECL.EX which declares constants and types.
- For error handling, declare a 10-character string type to hold the name of the command in which an error may occur.
- Declare a variable *cmd* to represent the return status of subsequent function calls.
- Define an interface select code constant *isc* as 7.
- Define a source address constant *source* as 717.
- Define a voltmeter address constant *dvm* as 722.

This example assumes select code 7, voltmeter address 22, and source address 17.

- Include a compiler directive to access IOPROC.EX which declares procedures and functions.

Note *Be sure to include the compiler directives in their proper places: IODECL.EX after the program name; and IOPROC.EX at the end of the declaration section, before any user-defined procedures or functions that use Library calls. For a complete listing of IODECL.EX and IOPROC.EX, see Appendix C.*

Write an error handling procedure:

```
PROCEDURE Error_handler (error:integer; routine;str10);

VAR
  estring : STRING(40);

BEGIN
  IF error <> noerr THEN BEGIN
    Errstr (cmd,estring);
    WRITELN ('Error in call to ',ROUTINE);
    WRITELN (ERROR:6, ESTRING);
    WRITE ('Press <RETURN> to continue: ');
    READLN;
  END;
END;
```

- In the procedure header, declare an *error* variable to hold the error number, and a *routine* string to hold the name of the command in which the error occurred.

- Declare *estring* to hold the error message string.

- If there is an error (error <> noerr):
  - call the Pascal Library error handling procedure Errstr. Pass it the command return variable *cmd*, and *estring* to hold the error message. (The Errstr routine returns Pascal error message strings. It is explained in the Error Handling section of this chapter.)
  - print a message telling where the error occurred.
  - print the error number and the error message string.
  - Display a prompt to continue when <RETURN> is pressed.

- Include a READLN statement to accept the <RETURN>. (<CTRL-C> terminates the program.)

This procedure displays the error and waits for you to type RETURN or <CTRL-C>. RETURN continues the program from where the error occurred, and <CTRL-C> aborts the program and returns control to the MS-DOS level.

Initialize the bus and the instruments:

```
PROCEDURE Initialize;

BEGIN
  cmd := IORESET (isc);
  Error_handler (cmd, 'IORESET     ');
  cmd := IOTIMEOUT (isc, 5.0);
  Error_handler (cmd, 'IOTIMEOUT ');
  cmd := IOCLEAR (isc);
  Error_handler (cmd, 'IOCLEAR     ');
END;
```

- Set the interface to its default configuration.
- Define a timeout of 5 seconds. Note that the timeout parameter, passed by value, can be expressed as a literal (5.0) in Pascal.
- Return all devices to a known state with IOCLEAR.
- This program calls the Error_handler procedure after each command. Pass it the command's return variable (which contains the error number) and the name of the command. Include enough spaces to set the command name field to 10 spaces.

Program the source:

```
PROCEDURE Source_setup;

VAR
  codes : STRING(38);
BEGIN
  codes := 'RF2 FU1 ST1KH SP10KH MF1KH AM2VR TI5SE';
  cmd := IOOUTPUTS (source, codes, 38);
  Error_handler (cmd, 'IOOUTPUTS ');
END;
```

- Declare the *codes* string to hold instrument programming codes.

- Assign the necessary source programming codes to the *codes* string:

  RF2 - select the rear panel signal output.
  FU1 - select the sine wave function.
  ST1KH - select a starting frequency of 1 kHz.
  SP10KH - select a stopping frequency of 10 kHz.
  MF1KH - select a marker frequency of 1 kHz.
  AM2VR - select an amplitude of 2 Vrms.
  TI5SE - select a sweep time of 5 seconds.

- Send the programming codes to the source with IOOUTPUTS.

Program the voltmeter:

```
PROCEDURE Dvm_setup;

VAR
  codes: STRING (38);
BEGIN
  codes := 'H SM004 F2 R4 FL0 Z0 4STG 20STN RS1 T4';
  cmd := IOOUTPUTS (dvm, codes, 38);
  Error_handler (cmd, 'IOOUTPUTS ');
END;
```

- Declare the *codes* string to hold instrument programming codes.

- Assign the necessary programming codes to *codes*:

H	- software-reset the voltmeter.
SM004	- set the service request mask to enable the voltmeter to set the interface SRQ line when it finishes taking readings (when the Data Ready bit of the response byte is set).
F2	- select the AC volts function.
R4	- select the 10 volt range.
FL0	- turn off filtering.
Z0	- turn off auto zero.
4STG	- select the 4-digit display.
20STN	- take 20 readings.
RS1	- turn on reading storage.
T4	- select hold trigger.

- Send the codes to the voltmeter with IOOUTPUTS.

Trigger the instruments:

```
PROCEDURE Trigger;

VAR
  codes : STRING(2);
BEGIN
  cmd := IOTRIGGER (dvm);
  Error_handler (cmd, 'IOTRIGGER ');
  codes := 'SS';
  cmd : =IOOUTPUTS (source, codes,2);
  Error_handler (cmd, 'IOOUTPUTS ');
END;
```

- Declare the *codes* string to hold instrument programming codes.

- Use IOTRIGGER to trigger the voltmeter.

- Enter the programming code necessary to trigger the source into the *codes* string. Send it, with the proper length parameter to the source with IOOUTPUTS.

These lines demonstrate that some instruments respond to an HP-IB trigger command, while others must be triggered with instrument-specific programming codes.

Wait for the voltmeter to finish reading:

```
PROCEDURE Wait_for_SRQ;

CONST
  srqline = 1;
VAR
  response : INTEGER;
  done     : BOOLEAN;
BEGIN
  done := FALSE;
  REPEAT
    REPEAT
      cmd := IOSTATUS (isc, srqline, response);
      Error_handler (cmd, 'IOSTATUS   ');
    UNTIL response <> 0;
    cmd:= IOSPOLL (dvm, response);
    Error_handler (cmd, 'IOSPOLL    ');
    done := (response AND 68) = 68;
  UNTIL done;
END;
```

- Assign the interface condition whose status is being checked to *srqline*. In this case, check for condition 1 - the SRQ line.

- Declare *response* and *done* variables for use in status checking.

- Set *done* to false before the status check.

- Use IOSTATUS with *srqline* set to 1 to see if the interface SRQ line has been set. Put the result in *response*.

- As long as *response* is zero, repeat the status check because the voltmeter has not yet set the SRQ line (indicating it is finished). As soon as *response* changes to some non-zero value, perform a serial poll of the voltmeter to see which of its conditions, if any, set the SRQ line. Also, the serial poll clears the SRQ.

- The result of the serial poll is the status byte of the voltmeter returned in *response*. Compare *response* and the value 68 - the sum of the SRQ bit (64) and the Data Ready bit (4). If these bits are set, *done* is true and the program continues. If they are not set, *done* is false and the status check is performed again.

Enter the readings into an array and print them:

```
PROCEDURE Readout;

VAR
  codes      : STRING (14);
  length     : INTEGER;
  readings   : REALARRAY(20);
  i          : INTEGER;
  numvalues : INTEGER;
BEGIN
  numvalues :=20;
  length := 14;
  codes := 'S01 -20STR RER';
  cmd := IOOUTPUTS (dvm,codes,length);
  Error_handler (cmd, 'IOOUTPUTS ');
  cmd := IOEOI (isc,0);
  Error_handler (cmd, 'IOEOI      ');
  cmd := IOENTERA (dvm, readings, numvalues);
  Error_handler (cmd, 'IOENTERA   ');
  WRITELN ('The readings are: ');
  FOR i := 1 TO numvalues DO WRITELN (readings [i]);
END;
```

- Declare all necessary variables: *codes* to hold instrument programming codes; *length* to define the length of the codes;

readings to hold the voltmeter readings; *i* for a for/next loop; and *numvalues* for the number of values to enter.

- Program the voltmeter to output its stored readings:

  S01 - turn on system output mode.
  -20STR - un-store 20 readings.
  RER - recall (output) the 20 readings.

- Disable the EOI mode so reading doesn't terminate after entering only one value.

- Set the number of readings to 20, and use IOENTERA to enter the readings from the voltmeter. Put them in the *readings* array.

- Use a loop to print the readings from the *readings* array.

Assemble the procedures for execution:

```
BEGIN {main}
  Initialize;
  Source_setup;
  Dvm_setup;
  Trigger;
  Wait_for_SRQ;
  Readout;
END.
```

Saving the Pascal Program

When you have finished writing the program, save it as "PROGRAM.PAS" in the Library directory LIB. (The method by which you save it depends on your text editor.)

Compiling the Pascal Program

Once the Pascal program is finished, you must compile it. This example assumes that the necessary compiling and linking programs are in the root directory of disc A, as is the Library directory.

Note

These instructions are for MS-Pascal version 3.13. If you have version 3.30 (or later), you may have to set up some environment variables. See your Pascal manual for instructions.

From MS-DOS, begin by setting the path:

```
> PATH=A:\
```

Select the Library directory as the current directory:

```
> A:
> CD \LIB
```

Pass One

From MS-DOS, type:

```
> PAS1
```

As pass 1 is executing, you are prompted for several filenames.

- When you are prompted for a source filename, type:

  PROGRAM

  The compiler will append the extension .PAS to the source filename resulting in PROGRAM.PAS.

- When you are prompted for an object filename, press RETURN. The compiler will use the source filename and append .OBJ to it: PROGRAM.OBJ.

- When you are prompted for a source listing filename, press RETURN; a source listing is not required for this example.

- When you are prompted for an object listing, press RETURN; an object listing is not required for this example.

If there are no errors, the following message is displayed at the end of pass 1:

```
Pass One    No errors detected.
```

Pass Two

To run pass 2, type:

```
> PAS2
```

Pass 2 displays no intermediate prompts on the screen.

Assuming there are no errors, pass 2 displays the following when it is completed:

```
Code Area Size =
Cons Area Size =
Data Area Size =

Pass Two    No Error Detected
```

Pass three is not required for this example.

Linking the Pascal Files

When compilation is successfully completed, you are ready to link the Pascal program.

From MS-DOS, begin by typing:

```
> LINK
```

As linking executes, several prompts appear on the screen:

- When you are prompted for object modules, type:

  PROGRAM

  The linker will append the extension .OBJ to the file name resulting in PROGRAM.OBJ.

- When you are prompted for a run file, press RETURN. The linker will use the source filename and attach .EXE to it: PROGRAM.EXE.

- When you are prompted for a listing file, press RETURN; a listing file is not required for this example.

- When you are prompted for runtime libraries, type:

HPIB

This directs the linker to link the Pascal command library file HPIB.LIB to your program. It also links the Pascal system library file which should be named PASCAL.LIB and placed in the Library directory. See Getting Started at the beginning of this chapter for more information on copying the necessary MS-Pascal files to your system.

With this information, the system begins linking the example program with all necessary modules in the runtime library. The MS-DOS prompt is displayed when linking is complete.

Running the Pascal Program

When you are ready to run the program, connect the SIGNAL output of the source to the VOLTS input of the DVM. (Include a 50 ohm load in this line to insure proper readings.) Use HP-IB cables to connect the instruments to your computer.

The result of the compilation and linking procedures is an executable program file called "PROGRAM.EXE" which is saved in the current directory.

When you are ready to run it type:

> PROGRAM

Watch the display on the function generator. You will see the various functions (sine wave, AC volts, sweep time) displayed as they are programmed.

The voltmeter displays its operation as well - you can watch it take readings, store them and output them to the "Readings" array.

As the program ends, it displays the readings on your screen.

Pascal Error Handling

During Pascal program execution, errors may occur as a result of I/O operations. For example, a device may not respond, a timeout may occur, or you might specify an incorrect select code or some invalid parameter.

Since there is no built-in error reporting procedure for Library commands, you should include error handling code in your application program. Otherwise, errors go untreated and may result in unpredictable results. To help you provide the necessary code, this section explains error handling in general for Pascal programming.

Error Message Mnemonics

As an aid to error testing, each error has an associated mnemonic variable. The mnemonics are declared in the Pascal "IODECL.EX" file.

You can use the mnemonic variables to test for the occurrence of errors. For example, error 0 - no error occurred - has the mnemonic NOERR. In Pascal,

```
"IF cmd <> noerr THEN
WRITELN ('Error in setup =',cmd);
```

would print the specified message and the error number (returned in *cmd*) if an error occurred.

Table 3-2 at the end of this chapter lists the Pascal Library errors and their mnemonics.

Error Reporting

In Pascal, each Library call is referenced as a function whose return value represents the error status of the operation. A return value of 0 indicates the HP-IB command completed without error. Values 1 through 8 indicate various other errors as shown in Table 3-2.

It is good practice to check for errors after each HP-IB function call. Since a typical program has more than one function call, you can write a single error handling routine to use with all of them. Here is an example:

```
PROCEDURE Checkerror (error: integer; routine: str12);

VAR
  estr: STRING(40);

BEGIN
  IF error <> noerr THEN BEGIN
  Errstr (error, estr);
  WRITELN ('Error ', error:4, estr, 'was encountered');
  WRITELN (' in call to HP-IB function', routine);
  WRITELN;
  WRITELN ('Press <RETURN> to continue: ');
  READLN;
  END;
END;
```

This example uses "Errstr" — a procedure provided in the Pascal IOPROC.EX file that returns an error messge string corresponding to the error value.

The parameter *routine* in Checkerror gives you an indication of where the error occurred.

noerr is one of the mnemonic constants established in the file IODECL.EX that correspond to the possible HP-IB errors. Other error mnemonics are listed in Table 3-2 at the end of this chapter.

Checkerror might then be used as follows:

```
.
.
.
cmd := IOENTER (709, reading);
Checkerror (cmd, ' IOENTER ');
.
.
.
```

Table 3-2. lists the Pascal Library errors.

Pascal Syntax Reference

This section presents syntax descriptions for the Library commands as they are used with Pascal. Several preliminary definitions are given, followed by a syntax description of each command.

Parameter Passing

In Pascal, you can pass parameters to procedures and functions in two ways: pass by value and pass by reference. Those passed by reference are indicated by "_REF" attached to their designators in the syntax reference.

With pass by value, a copy of the current value of the parameter is made for the called routine, and the original copy of the data is unchanged. In this case, you may specify a variable (e.g., isc), a literal (e.g., 709) or an expression (i.e., isc * 100 + 9) as the parameter in the call statement.

With call by reference, only a single copy of the data exists. Therefore, any changes made to the variable in the subroutine are reflected in both the called routine and the calling procedure. Whenever a reference parameter is indicated, you must use a variable; literals and expressions are not allowed.

A variety of parameter types and modes are used in the descriptions of the Pascal Library commands. The following definitions describe the parameters.

Integer Expression

This is any valid combination of integers, integer variables, integer functions and integer operators that evaluate to a single integer value. The range of integers is limited to whole numbers -32767 to 32767. Examples:

```
701
isc
isc * 100 + 9
700 + ord('$')
```

4-Byte Integer Expression

This is similar to an Integer Expression, except the range of valid values is -2,147,483,647 through 2,147,483,647. Anywhere that a 4-Byte Integer Expression is indicated, you can use an Integer Expression.

The first parameter of each Library command is declared as an INTEGER4 (a built-in MS-Pascal type) and is therefore a 4-Byte Integer Expression. This permits you to use Extended Listener and Talker Addressing as discussed later.

Integer Variable

This is a variable declared to be of type INTEGER that is to be passed by reference. You may not use variables to type INTEGER4 when an Integer Variable is specified, nor may you use a constant, literal or integer expression such as isc * 100 + 9. Integer Variables are indicated when a Library function returns a meaningful value to a parameter, such as with IOGETTERM or IOSPOLL.

Real Expression

This is a valid combination of real numbers, real variables, real functions and real number operators that evaluates to a single real value. Only 4-byte real numbers are allowed, limiting the range of real numbers to approximately 1E-38 to 1E+38 and the precision to 7 digits. Example:

```
1.23
data_value
sqrt(data_value)
2 * pi * r
```

Real Variable

This is a variable declared to be of type REAL. For this variable, you may not use constants, literals or other expressions. Real Variables are required only with the IOENTER statement which returns a value read from an instrument or other device.

Real Array

This is an array of REAL values. When you declare arrays for the IOENTERA and IOOUTPUTA commands, use the super type REALARRAY as defined in the file IODECL.EX. Although the theoretical size limit for arrays is 32,767 elements, the actual limit is smaller. This is due to memory size limitations of MS-Pascal.

Be careful when using very large arrays. Declared globally, they can result in link time errors. Declared within subroutines, they may compile and link without error, but cause system problems later when printed or accessed in computations. Examples:

```
VAR values: REALARRAY(20);
                -or-
TYPE real20 = REALARRAY(20);
VAR values: REAL20;
```

Each of the above examples allocates an array of 20 elements which can be accessed as values[1] ... values[20]. Each element is a real number. If the array is to be used as a parameter to a user-defined subroutine, use the latter form.

String Variable

This is a variable declared to be of type STRING. You must establish the string size before you call a Library function that uses a string variable. Valid string sizes are 1 through 32,767. As with Real Arrays, be careful when you use very large strings. Also, you may not use LSTRINGs or string literals (such as 'My String'). Examples:

```
VAR instring: STRING(10);
                -or-
TYPE str10 = STRING(10);
VAR instring: STR10;
```

If the variable is to be used as a parameter to a user-defined subroutine, use the latter form.

Character Expression

This is any variable of type CHAR or a constant, literal, or character function that represents a single ASCII character. A variable with type STRING(1) is not compatible with a CHAR variable. In the example above, instring[1] is a character variable. Examples:

```
ch
'A'
chr(10)
```

Command Addressing

The first parameter of each Library command specifies an interface select code or a device bus address. Some commands allow only a select code (e.g., IOABORT); some permit only a device address (e.g., IOSPOLL); and some permit either (e.g., IOCLEAR).

If you specify a select code, the command is directed to the interface. No bus addressing is done prior to the transfer of data or commands. If you specify a device address, devices on the bus are first addressed using the sequence:

- UNLISTEN
- TALK or LISTEN ADDRESS of the System Controller
- LISTEN or TALK ADDRESS of the Target Device

The addressing is followed by data or bus command transfer, depending on the function.

A select code is differentiated from a device address by the magnitude of the value specified. The interface treats a value in the range 0 to 99 as a select code. Only 1 through 16 are considered valid select codes, however, and there is only one valid select code for each HP-IB in your computer. If you specify a non-existent select code, an error results.

A device address is composed of the select code and the primary bus address of a device. It is calculated as:

(select code * 100) + primary bus address

A valid primary bus address is in the range 0 to 30. Address 30 is reserved as the address of the controller.

Secondary Addressing

You can use extended talker and listener functions (secondary addressing) with the Library. To specify an extended address, use the formula:

(select code * 10000) + (primary bus address * 100) + secondary address

A secondary address may be in the range 0 to 31. You can use extended addressing with any command in which the first parameter may be a device address.

As an example, consider a system with select code 7 that has a device with primary address 9 and secondary address 15. The first parameter of a Library call could be:

```
cmd := IOCLEAR (7);          { Select code       }
cmd := IOCLEAR (709);        { Primary Address   }
cmd := IOCLEAR (70915);      { Secondary Address}
```

HP-IB Mnemonics

Throughout the syntax description, HP-IB terms are listed by abbreviation rather than by name. For example, Go To Local is listed as GTL. For your convenience, the mnemonics are summarized in Table 3-1.

MNEMONIC	DEFINITION
ATN	Attention
DCL	Device Clear
EOI	End Or Identify
EOL	End Of Line
GET	Group Execute Trigger
GTL	Go To Local
IFC	Interface Clear
LAD	Listen Address
LLO	Local Lockout
MLA	My Listen Address
MTA	My Talk Address
OSA	Other Secondary Address
PPC	Parallel Poll Configure
PPD	Parallel Poll Disable
PPU	Parallel Poll Unconfigure
REN	Remote Enable
SDC	Selected Device Clear
SPD	Serial Poll Disable
SPE	Serial Poll Enable
SRQ	Service Request
TAD	Talk Address
UNL	Unlisten
UNT	Untalk

Table 3-1. HP-IB Mnemonics

Command Syntax

The following pages present a detailed syntax description of the Library commands as they are used in Pascal.

IOABORT

This command aborts all activity on the interface.

Syntax

IOABORT *(select code)*

select code - a 4-byte integer expression specifying the interface select code.

Examples

```
VAR
  cmd:INTEGER;

cmd:=IOABORT (7);
```

Bus Activity

- IFC is pulsed (>100 uS).
- REN is set.
- ATN is cleared.

Comments

Devices in Local Lockout will remain locked out.

Possible errors are NOERR, ETIME and ESEL.

IOCLEAR

This command returns a device to a known, device-dependent state. It can be addressed to the interface, or to a specific device.

Syntax

IOCLEAR *(device address)*
IOCLEAR *(select code)*

device address - a 4-byte integer expression specifying the address of a device to be cleared.

select code - a 4-byte integer expression specifying the select code of the interface on which all devices are to be cleared.

Examples

```
VAR
  cmd:INTEGER;

cmd:=IOCLEAR (723); {Clear the device at address 23.}
    :
cmd:=IOCLEAR (7); {Clear all devices on the interface.}
```

Bus Activity

If a device address is specified:

- ATN is set.
- UNL is sent.
- MTA is sent.
- LAD is sent.
- OSA is sent if specified.
- SDC is sent.

If a select code is specified:

- ATN is set.
- DCL is sent.

Comments

Possible errors are NOERR, ETIME and ESEL.

IOCONTROL

This command directly sets status conditions in the interface. It can be used to address/unaddress the interface as a talker or listener, or set the interface's bus address.

Note

IOCONTROL should be used with caution since it operates directly on the interface. For example, if the interface's device address is changed, some operating system code - such as printer or disc drivers - may not function properly.

Syntax

IOCONTROL *(select code,condition,status)*

select code - a 4-byte integer expression specifying the interface select code.

condition - an integer expression specifying the status condition which is to be set. Conditions which can be set are:

Value	Description
5	- address or unaddress the interface as talker.
6	- address or unaddress the interface as listener.
7	- set the interface's bus address.

status - an integer expression specifying zero (clear) or a non-zero value (set) for conditions 5 and 6; or a bus address (0-30) for condition 7.

Examples

```
VAR
  cmd:INTEGER;

cmd:=IOCONTROL (7, 5, 1); {Address the interface as talker.}
```

Bus Activity

None.

Comments

Possible errors are NOERR, ESEL and ERANGE.

IOENTER

This command reads a single real number. Reading continues until:

- the EOI line is sensed true (if enabled).
- a linefeed is encountered after the number starts.

Numeric characters include digit "E", "+", "-" and "." in the proper sequence for representing a number. " "(space) is not a numeric character.

Syntax

IOENTER *(device address,data_REF)*
IOENTER *(select code,data_REF)*

device address	- a 4-byte integer expression specifying a device address.
select code	- a 4-byte integer expression specifying the interface select code.
data_REF	- a real variable passed by reference into which the reading is placed.

Examples

```
VAR
  reading:REAL;
  cmd:INTEGER;

cmd:=IOENTER (722,READING); {Input a number from device 722; place it in
                             READING.}
```

Bus Activity

If a device address is specified:

- ATN is set.
- UNL is sent.
- MLA is sent.
- TAD is sent.
- OSA is sent if specified.
- ATN is cleared.
- Data is entered.

If a select code is specified:

- If the interface is not addressed to listen, an error results. Otherwise, ATN is cleared and the data is read from the interface.

Comments

If a select code is to be specified in the command, the interface must first be addressed to listen (with IOSEND, for example) or an error occurs.

Possible errors are NOERR, ETIME, ESEL, EADDR and ENUM.

IOENTERA

This command enters numbers from a device or from the interface and places them into a realarray. Reading continues until:

- the EOI line is sensed true (if enabled).
- a linefeed is encountered after the specified number of elements is received.

Numeric characters include digit "E", "+", "-" and "." in the proper sequence for representing a number. ""(space) is not a numeric character.

Syntax

IOENTERA *(device address,readings_REF,elements_REF)*
IOENTERA *(select code,readings_REF,elements_REF)*

device address	- a 4-byte integer expression specifying a device address.
select code	- a 4-byte integer expression specifying the interface select code.
reading_REF	- a real array passed by reference into which the readings are placed.
elements_REF	- an integer variable passed by reference specifying the maximum number of elements to be read. Upon return, it indicates the number of elements actually read.

Examples

```
TYPE
  realarray=SUPER ARRAY [1..*] OF REAL; {This is declared in 'IODECL.EX'}

VAR
  readings:REALARRAY (50);
  elements:INTEGER;
  cmd:INTEGER;

  elements:= 50;
  cmd:= IOENTERA (723,readings,elements); {Read a maximum of 50 values from
                                           device 723; put them in READINGS.}
```

Bus Activity

If a device address is specified:

- ATN is set.
- UNL is sent.
- MLA is sent.
- TAD is sent.
- OSA is sent if specified.
- ATN is cleared.
- Data is entered.

If a select code is specified:

- If the interface is not addressed to listen, an error results. Otherwise, ATN is cleared and the data is read.

Comments

Non-numeric characters that do not properly belong in a real number are considered value separators. Thus, the sequence "1,234,567" is entered as three numbers, not as "1234567".

The number of readings available is dependent upon the device.

If a select code is to be used in the command, the interface must first be addressed to listen (with IOSEND, for example) or an error occurs.

Possible errors are NOERR, ETIME, ESEL, EADDR and ENUM.

IOENTERS

This command enters a character string from a device or from the interface. Reading continues until:

- the EOI line is sensed true (if enabled).
- the termination character set by IOMATCH is received (linefeed is the default).
- the number of characters specified by the *length_REF* parameter is received.

Syntax

IOENTERS *(device address,data_REF,length_REF)*
IOENTERS *(select code,data_REF,length_REF)*

device address	- a 4-byte integer expression specifying a device address.
select code	- a 4-byte integer expression specifying the interface select code.
data_REF	- a string variable passed by reference into which the string read is placed.
length_REF	- an integer variable passed by reference specifying the maximum number of bytes to be read. On return, it indicates the number of bytes actually read.

Examples

```
VAR
  info:STRING(10);
  length:INTEGER;
  cmd:INTEGER;

length:=10;
cmd:=IOENTERS (723,INFO,LENGTH); {Read a string of 10 characters maximum
                                  from device 723.}
```

Bus Activity

If a device address is specified:

- ATN is set.
- UNL is sent.
- MLA is sent.
- TAD is sent.
- OSA is sent if specified.
- ATN is cleared.
- Data is entered.

If a select code is specified:

- If the interface is not addressed to listen, an error results. Otherwise, ATN is cleared and the data is read.

Comments

If a select code is to be specified in the command, the interface must first be addressed to listen (with IOSEND or a previous IOENTER, for example) or an error occurs.

The termination character is entered as part of the string.

Possible errors are NOERR, ETIME, ESEL, EADDR and ENUM.

IOEOI

This command enables or disables the End Or Identify (EOI) mode of the interface. It is used to:

- enable or disable a write operation to set the EOI line on the last byte of the write.
- enable or disable a read operation to terminate upon sensing the EOI line true.

Default is EOI enabled.

Syntax

IOEOI *(select code,state)*

select code	- a 4-byte integer expression specifying the interface select code.
state	- an integer expression; a non-zero value

Examples

```
VAR
  state:INTEGER;
  cmd:INTEGER;

state:= 0;
cmd:=IOEOI (7,state); {Disable EOI.}
```

Bus Activity

None.

Comments

When reading with EOI enabled, receipt of a byte with EOI set causes the I/O operation to terminate, regardless of whether you are reading a string, a real number, or an array of real numbers.

When writing, EOI is set on the last byte of the End Of Line sequence if EOI is enabled. Note that if the EOL sequence is of 0 length, EOI is set on the last data byte sent.

When sending real number arrays, the EOL sequence (and subsequent EOI) is appended after the last element in the array, not after each element.

Note that IOSEND does not set EOI since this line has a different meaning in Command mode.

Possible errors are NOERR and ESEL.

IOEOL

This command defines the End Of Line (EOL) string that is to be sent following every IOOUTPUT, IOOUTPUTA, or IOOUTPUTS command. Default is carriage return/linefeed.

Syntax

IOEOL *(select code,endline_REF,length)*

select code - a 4-byte integer expression specifying the interface select code.

endline_REF - a string expression passed by reference specifying the EOL string which is to be sent following a data transmission. A maximum of 8 characters may be specified.

length - an integer expression specifying the length of the termination string. If zero is specified, no characters are appended to a data transmission. If *length* is less than zero an error occurs.

Examples

```
VAR
  length:INTEGER;
  endline:STRING(2);
  cmd:INTEGER;

length:=2;
endline[1]:=CHR(13);
endline[2]:=CHR(10);
cmd:=IOEOL (7, endline, length); {EOL = CR/LF.}
```

Bus Activity

None.

Comments

With IOOUTPUTA, the EOL sequence is appended after *all* the elements have been sent, not following each element.

Possible errors are NOERR, ESEL and ERANGE.

IOGETTERM

This command determines the reason for which the last read terminated.

Syntax

IOGETTERM *(select code,reason_REF)*

select code - a 4-byte integer expression specifying the interface select code.

reason_REF - an integer variable passed by reference to receive the logical OR-ing of the reasons for which the last read terminated. The possible reasons for termination are:

Value	Description
0	- The read was terminated for some reason not covered by any of the other reasons (e.g., error or timeout).
1	- The expected number of elements was received.
2	- The termination character set by IOMATCH was encountered.
4	- The EOI line was sensed true.

Examples

```
VAR
  reason:INTEGER;
  cmd:INTEGER;

cmd:=IOGETTERM (7,reason);
IF ((reason AND 4)=4) THEN
WRITELN ('EOI ENCOUNTERED');
```

Bus Activity

None.

Comments

Upon return, the *reason* integer contains the logical OR-ing of the reasons for termination. For example, if the last read terminated when the termination character was encountered and EOI was set, the value of *reason* would be 2 + 4 = 6.

Possible errors are NOERR, and ESEL.

IOLLOCKOUT

This command sends a Local Lockout (LLO) to disable a device front panel. It is received by all devices on the interface, whether or not they are addressed to listen.

Syntax

IOLLOCKOUT *(select code)*

select code - a 4-byte integer expression specifying the interface select code.

Examples

```
VAR
  cmd:INTEGER;

cmd:=IOLLOCKOUT (7);
```

Bus Activity

- ATN is set.
- LLO is sent.

Comments

If a device is in Local mode when LLO is received, LLO does not take effect until the device is addressed to listen.

If the REN line is not set, this command has no effect.

Possible errors are NOERR, ETIME and ESEL.

IOLOCAL

This command executes a Go To Local (GTL) or clears the REN line to enable a device front panel.

Syntax

IOLOCAL *(device address)*
IOLOCAL *(select code)*

device address - a 4-byte integer expression specifying a device address.

select code - a 4-byte integer expression specifying the interface select code.

Examples

```
VAR
  cmd:INTEGER;

cmd:=IOLOCAL (722); {Place device 722 in Local mode.}
```

Bus Activity

If a device address is specified:

- ATN is set.
- UNL is sent.
- MTA is sent.
- LAD is sent.
- OSA is sent if specified.
- GTL is sent.

If a select code is specified:

- REN is cleared.
- ATN is cleared.

Comments

If a device address is specified, the device is temporarily placed in Local mode - it will return to Remote if it is later addressed to listen. If Local Lockout is in effect, the device will return to the Lockout state if it is later addressed to listen.

If an interface select code is specified, all instruments on the bus are placed in Local mode, and any Local Lockout is cancelled.

Possible errors are NOERR, ETIME and ESEL.

IOMATCH

This command defines the character upon which an IOENTERS operation terminates. The character becomes part of the entered string when received and is initially set to linefeed.

Syntax

IOMATCH *(select code,character,flag)*

select code	- a 4-byte integer expression specifying the interface select code.
character	- a character upon which an IOENTERS terminates.
flag	- an integer expression indicating whether character matching should be enabled or disabled. Zero disables it and any non-zero value enables it.

Examples

```
VAR
  match:CHAR;
  flag:INTEGER;
  cmd:INTEGER;

match:=CHR(10); {Terminate IOENTERS on linefeed.}
flag:=1;
cmd:=IOMATCH (7,match,flag);
```

Bus Activity

None.

Comments

Only a single match character may be specified in this command.

IOMATCH does not apply to IOENTER or IOENTERA.

Possible errors are NOERR and ESEL.

IOOUTPUT

This command outputs a real number to a device or to the interface. After the number is sent, the EOL string is sent and the EOI line is set (if enabled).

Syntax

IOOUTPUT *(device address,data)*
IOOUTPUT *(select code,data)*

device address	- a 4-byte integer expression specifying a device address.
select code	- a 4-byte integer expression specifying the interface select code.
data	- a real expression specifying the number to be output.

Examples

```
VAR
  data:REAL;
  cmd:INTEGER;

data:=12.3;
cmd:=IOOUTPUT (722,data); {Output'12.3' to device 722.}
```

Bus Activity

If a device address is specified:

- ATN is set.
- UNL is sent.
- MTA is sent.
- LAD is sent.
- OSA is sent if specified.
- ATN is cleared
- Data is output.
- EOL is output.

If a select code is specified:

- If the interface is not addressed to talk, an error results. Otherwise, ATN is cleared and the data is sent followed by EOL.

Comments

Numbers between 1E-5 and 1E+6 are rounded to seven significant digits and output in floating point notation. If the number rounds to an integer value, the decimal point is not sent. Numbers outside this range are rounded to seven significant digits and output in scientific notation.

If the number is positive, a leading space is output for the sign; if negative, a leading "-" is output.

If a select code is to be specified, the interface must first be addressed to talk (with IOSEND, for example), or an error occurs.

Possible errors are NOERR, ETIME, ESEL and EADDR.

IOOUTPUTA

This command outputs an array of real numbers to a specified device, or to the bus. Values output are separated by commas. After the last byte of data is sent, the EOL string is sent and the EOI line is set (if enabled).

Syntax

IOOUTPUTA *(device address,data_REF,elements)*
IOOUTPUTA *(select code,data_REF,elements)*

device address	- a 4-byte integer expression specifying a device address.
select code	- a 4-byte integer expression specifying the interface select code.
data_REF	- a real array passed by reference containing the real numbers to be transmitted.
elements	- an integer variable specifying the number of elements in the array to be transmitted.

Examples

```
TYPE
  realarray=SUPER ARRAY [1..*] OF REAL; {-declared in 'IODECL.EX'}
  info:REALARRAY(10);
  num_elements:INTEGER;
  cmd:INTEGER;

num_elements:=10;
cmd:=IOOUTPUTA (722,info,num_elements); {Output the array INFO to device
                                          722.}
```

Bus Activity

If a device address is specified:

- ATN is set.
- UNL is sent.
- MTA is sent.
- LAD is sent.
- OSA is sent if specified.
- ATN is cleared.
- Data is output.
- EOL is output.

If a select code is specified:

- If the interface is not addressed to talk, an error results. Otherwise, ATN is cleared and the data is sent followed by EOL.

Comments

If a select code is to be used as a parameter, the interface must first be addressed to talk (with IOSEND,for example), or an error occurs.

Numbers between 1E-5 and 1E+6 are rounded to seven significant digits and output in floating point notation. If the number rounds to an integer value, the decimal point is not sent. Numbers outside this range are rounded to seven significant digits and output in scientific notation.

If the number is positive, a leading space is output for the sign; if negative, a leading "-" is output.

Possible errors are NOERR, ETIME, ESEL and EADDR.

IOOUTPUTS

This command outputs a string to a specified device, or to the interface. After the string is sent, the EOL string is sent and the EOI line is set true (if enabled).

Syntax

IOOUTPUTS *(device address,data_REF,length)*
IOOUTPUTS *(select code,data_REF,length)*

device address	- a 4-byte integer expression specifying a device address.
select code	- a 4-byte integer expression specifying the interface select code.
data_REF	- a string variable passed by reference specifying the string to be sent.
length	- an integer expression specifying the length of the data string.

Examples

```
VAR
  info:STRING(4);
  length:INTEGER;
  cmd:INTEGER;

info:='1ST1';
length:=4;
cmd:=IOOUTPUTS (723,info,length); {Send the programming code "1ST1" to device
                                   723.}
```

Bus Activity

If a device address is specified:

- ATN is set.
- UNL is sent.
- MTA is sent.
- LAD is sent.
- OSA is sent if specified.
- ATN is cleared
- Data is output.
- EOL is output.

If a select code is specified:

- If the interface is not addressed to talk, an error results. Otherwise, ATN is cleared and the data is sent followed by EOL.

Comments

If a select code is to be used in the command, the interface must first be addressed to talk (with IOSEND, for example), or an error occurs.

Possible errors are NOERR, ETIME, ESEL and EADDR.

IOPPOLL

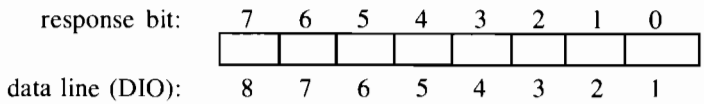
This command performs a parallel poll of the interface. It sets a variable to a value (0 - 255) representing the response of those instruments on the interface that respond to a parallel poll.

Syntax

IOPPOLL *(select code,response_REF)*

select code - a 4-byte integer expression specifying the interface select code.

response_REF - an integer variable passed by reference into which the parallel poll response byte is to be placed. The allowable range is 0 to 255. The eight bits of the response byte correspond to the eight data lines:



Hence, a value of 32 would indicate that a device(s) has responded to the parallel poll with a 'I' on DIO 6.

Examples

```
VAR
  response:INTEGER;
  cmd:INTEGER;
       :
cmd:=IOPPOLL (7,response);
```

Bus Activity

- ATN and EOI are asserted for 25 microseconds.
- The poll byte is read.
- EOI is cleared.
- ATN is restored to its previous state.

Comments

Not all devices are capable of responding to a parallel poll. Consult your particular device manuals for specifics.

Possible errors are NOERR, ETIME and ESEL.

IOPPOLLC

This command performs a Parallel Poll Configure. In preparation for a parallel poll command, it tells an instrument how to respond affirmatively to the parallel poll, and on which data line to respond.

In general, it sets a parallel poll response byte to reflect the response of a desired arrangement of instruments. Typically, you could define the bits to reflect the responses of particular instruments, or the logical-ORing of several instruments.

(Refer to IOPPOLL for more information.)

Syntax

IOPPOLLC *(device address,configuration)*
IOPPOLLC *(select code,configuration)*

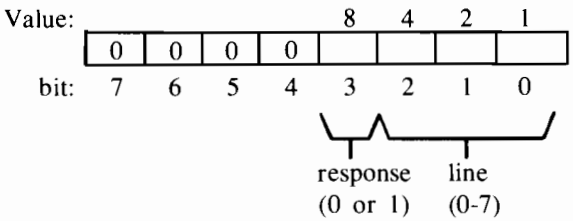
device address - a 4-byte integer expression specifying the bus address of the device to be configured.

select code - a 4-byte integer expression specifying the interface select code.

configuration - an integer expression sent to the specified device indicating:

- how to respond: 1 or 0.
- on which data line to respond: 0 to 7 (corresponding to DIO 1-8).

The configuration byte represented by the value is:



Bits 0 - 2 specify the data line (0 - 7). Bit 3 specifies the affirmative response (1 or 0). The range of *configuration* is 0 - 15.

Thus, to indicate an affirmative response of 1 on data line 4 (DIO 5), the configuration value would be 8 + 4 = 12.

Examples

```
VAR
  configuration:INTEGER;
  cmd:INTEGER;

configuration:=10; {Respond with a '1' on line 2.}
cmd:=IOPPOLLC (723,configuration);
```

Bus Activity

If a device address is specified:

- ATN is set.
- UNL is sent.
- MTA is sent.
- LAD is sent.
- OSA is sent if specified.
- PPC is sent.
- PPE is sent.

If a select code is specified:

- PPC is sent.
- PPE is sent.

Comments

Not all devices can be configured to respond to a parallel poll. Consult your particular device manuals for specifics.

Possible errors are NOERR, ETIME, ESEL and ERANGE.

IOPPOLLU

This command performs a Parallel Poll Unconfigure - it directs an instrument to *not* respond to a parallel poll. It can be addressed to the interface or to a specific device. (Refer to IOPPOLLC for more information.)

Syntax

IOPPOLLU *(device address)*
IOPPOLLU *(select code)*

device address - a 4-byte integer expression specifying a device address.

select code - a 4-byte integer expression specifying the interface select code.

Examples

```
VAR
  cmd:INTEGER;

cmd:=IOPPOLLU (722);
```

Bus Activity

If a device address is specified:

- ATN is set.
- UNL is sent.
- MTA is sent.
- LAD is sent.
- OSA is sent if specified.
- PPC is sent.
- PPD is sent.

If a select code is specified:

- ATN is set.
- PPU is sent.

Comments

Some devices cannot be unconfigured from the bus. Consult your particular device manuals for specifics.

Possible errors are NOERR, ETIME and ESEL.

IOREMOTE

This command places a device in Remote mode. It can be addressed to the interface or to a specific device.

Syntax

IOREMOTE *(device address)*
IOREMOTE *(select code)*

device address - a 4-byte integer expression specifying a device address.
select code - a 4-byte integer expression specifying the interface select code.

Examples

```
VAR
  cmd:INTEGER;

cmd:=IOREMOTE (723); {Place device 723 in Remote.}
      :
cmd:=IOREMOTE (7); {Set the interface REN line.}
```

Bus Activity

If a device address is specified:

- REN is set.
- ATN is set.
- UNL is sent.
- MTA is sent.
- LAD is sent.
- OSA is sent if specified.

If a select code is specified:

- REN is set.

Comments

If a select code is specified, a device will not switch into Remote until it is addressed to listen.

Possible errors are NOERR, ETIME, and ESEL.

IORESET

This command sets the interface to its start-up state in which it is system controller, active controller, not listening, and not talking.

In addition, it sets the following HP-IB parameters as indicated:

- The interface timeout is set to 0 seconds. (i.e., timeout is disabled.)
- The interface EOI mode is enabled.
- CR/LF is set as the EOL default.
- LF is set as the IOMATCH default.

Syntax

IORESET *(select code)*

select code - a 4-byte integer expression specifying the interface select code.

Examples

```
VAR
  cmd:INTEGER;

cmd:=IORESET(7);
```

Bus Activity

- IFC is pulsed (>100 uS).
- REN is cleared (>10 uS), then set.
- ATN is cleared

Comments

This command returns all devices on the interface to local (front panel) control.

Possible errors are NOERR, ETIME and ESEL.

IOSEND

This command sends a string of user-specified HP-IB commands to the interface. For example, to send an output command to several instruments simultaneously, you can establish their talk/listen status with the IOSEND command, then issue the command specifying a select code rather than a device address.

Syntax

IOSEND *(select code,commands_REF,length)*

select code - a 4-byte integer expression specifying the interface select code.

commands_REF - a string variable, passed by reference specifying a string of characters, each of which is treated as an interface command.

length - an integer expression specifying the number of characters in the command string.

Examples

```
VAR
  commands:STRING(4);
  length:INTEGER;
  cmd:INTEGER;

commands:='?)/4'; {Specifies unlisten, listen addresses 9, 15 and 20.}
length:=4;
cmd:=IOSEND (7,commands, length);
cmd:=IOTRIGGER (7); {Triggers devices at addresses 9, 15 and 20.}
```

Bus Activity

- ATN is set.
- Command bytes are sent.

Comments

All bytes are sent with ATN set. The EOL sequence is not appended, nor is EOI set.

Possible errors are NOERR, ETIME, ESEL and EPASS.

IOSPOLL

This command performs a serial poll of a specified device. It sets a variable representing the device's response byte.

Syntax

IOSPOLL *(device address,response_REF)*

device address - a 4-byte integer expression specifying the bus address of the device to be polled.

response_REF - an integer variable passed by reference into which the response byte is placed.

Examples

```
VAR
  response:INTEGER;
  cmd:INTEGER;

cmd:=IOSPOLL (723,response); {Perform a serial poll on device 723; put the response
                              byte in RESPONSE.}
```

Bus Activity

- ATN is set.
- UNL is sent
- MLA is sent.
- TAD is sent.
- OSA is sent if specified.
- SPE is sent.
- ATN is cleared.
- Poll byte is read.
- ATN is set.
- SPD is sent.
- UNT is sent.

Comments

Some devices are not capable of responding to a serial poll in which case polling may result in an error. Consult the appropriate manual to determine if an instrument can respond to a serial poll, and how its response byte is interpreted.

Possible errors are NOERR, ETIME and ESEL.

IOSTATUS

This command determines the interface status. It sets a variable representing a particular interface status condition.

Syntax

IOSTATUS *(select code,condition,status_REF)*

select code - a 4-byte integer expression specifying the interface select code.

condition - an integer expression from 0 - 7 specifying the condition whose status is being checked. The possible conditions are:

Value	Description
0	Is the interface currently in the Remote state? (always false)
1	What is the current state of the SRQ line?
2	What is the current state of the NDAC line?
3	Is the interface currently system controller? (always true)
4	Is the interface currently active controller? (always true)
5	Is the interface currently addressed as talker?
6	Is the interface currently addressed as listener?
7	What is the interface's bus address?

status_REF - an integer variable passed by reference into which the condition's status is placed. *status_REF* can be 1 for set or zero for clear; or an address (0 - 30) for condition 7.

Examples

```
VAR
  select:INTEGER;
  status:INTEGER;
  cmd:INTEGER;

select:=1;
cmd:=IOSTATUS (7,select,status); {Determine if SRQ is set.}
```

Bus Activity

None.

Comments

Possible errors are NOERR, ESEL and ERANGE.

IOTIMEOUT

This command sets an interface timeout value in seconds for those cases where an I/O operation does not complete (e.g., the printer runs out of paper). Default is 0 seconds, indicating that timeout is disabled.

Syntax

IOTIMEOUT *(select code,timeout)*

select code - a 4-byte integer expression specifying the interface select code.

timeout - a real expression specifying the length of the timeout period. 0.0 disables the timeout. A negative value is an error.

Examples

```
VAR
  timeout_val:REAL;
  cmd:INTEGER;

timeout_val:=1.23; {Timeout after 1.23 seconds.}
cmd:=IOTIMEOUT (7,timeout_val);
```

Bus Activity

None.

Comments

Timeout is effective for any interface operation that transfers data or commands.

A timeout error will occur only if timeout is enabled (i.e., set to some positive value).

Timeout should be established early in your program. It provides a way to recover from I/O operations that are otherwise incompleted.

The timeout value is a real number specified in seconds. To timeout after five seconds, use *timeout*=5.0; to timeout after one-half second, use *timeout*=0.5. Note that a timeout of 0.0 effectively disables any timeouts. The maximum allowable timeout is 4,096 seconds.

The resolution of a timeout may vary for different computer systems. For example, some computers have a timeout resolution of one second. Thus, timeout resolutions cannot be set more accurately than plus or minus 1 second on these computers.

Upon timeout, the function return variable (cmd in the examples) is set to 4.

Possible errors are NOERR, ESEL and ERANGE.

IOTRIGGER

This command triggers a device, or several devices.

Syntax

IOTRIGGER *(device address)*
IOTRIGGER *(select code)*

device address - a 4-byte integer expression specifying a device address.
select code - a 4-byte integer expression specifying the interface select code.

Examples

```
VAR
  cmd:INTEGER;

cmd:=IOTRIGGER (723);
```

Bus Activity

If a device address is specified:

- ATN is set.
- UNL is sent.
- MTA is sent.
- LAD is sent.
- OSA is sent if specified.
- GET is sent.

If a select code is specified:

- ATN is set.
- GET is sent.

Comments

Only one device can be triggered at a time if a device address is specified.

If a select code is specified, all devices on the bus that are addressed to listen (with IOSEND, for example) are triggered.

Possible errors are NOERR, ETIME, and ESEL.

Table 3-2 lists the Pascal Library errors.

Table 3-2. Pascal Library Errors

PASCAL ERROR	MNEMONIC	DEFINITION
0	NOERR	No error occurred.
1	EUNKNOWN	Unknown error.
		An error not covered in this list has occurred. Check for malfunctioning equipment or incorrect hardware configuration. Also, check for invalid lines in your application program.
2	ESEL	Invalid select code or device address.
		This error would most likely occur under these conditions:
		- The first parameter of a call should have been a valid select code, but a device address or an invalid select code was specified instead.
		- A device address was expected, but a select code was given; or a primary address outside the range 00 - 30, or a secondary address outside the range 00 - 31 was specified.
		- The device address of the interface was specified as a parameter in commands such as IOSPOLL, IOREMOTE and IOCLEAR.
3	ERANGE	Value out of range.
		A command parameter was specified outside its allowable range. This error would most likely occur in these commands:
		- IOSTATUS - The status specified was outside the range 0 - 7.
		- IOCONTROL - The specified value must be from 5 to 7. If 7 is selected, the valid address values are 0 - 30.
		- IOTIMEOUT - The specified timeout value must be greater than or equal to 0; a negative timeout value may not be entered.
		- IOPPOLLC - The configuration value must be between 0 and 15.
		- IOEOL - The specified length must be between zero and 8.

4	ETIME	Timeout. The time specified by IOTIMEOUT has elapsed since the last byte was transferred.
5	ECTRL	HP-IB must be controller. The interface must be active controller and/or system controller.
6	EPASS	Pass control not permitted. Pass control is not supported by the Library.
7	ENUM	Invalid number. No digit or an improperly formatted number was received during a real number read (IOENTER, IOENTERA). In this case, a value of zero is returned as the reading value.
8	EADDR	Improper addressing. An attempt was made to input/output data when the interface was not addressed to listen/talk. This error is likely to occur if a select code was specified for a read or write, but the interface was not properly addressed to talk or listen.

A HP-IB Review

Bus Description

The Hewlett-Packard Interface Bus (HP-IB) is HP's implementation of the IEEE-488 communication interface. It is used by a variety of instruments, disc drives and peripherals manufactured by Hewlett-Packard and other companies. The HP-IB is a 16-line bus that connects up to 15 devices in parallel on a communication link. Figure A-1 illustrates the HP-IB connector.

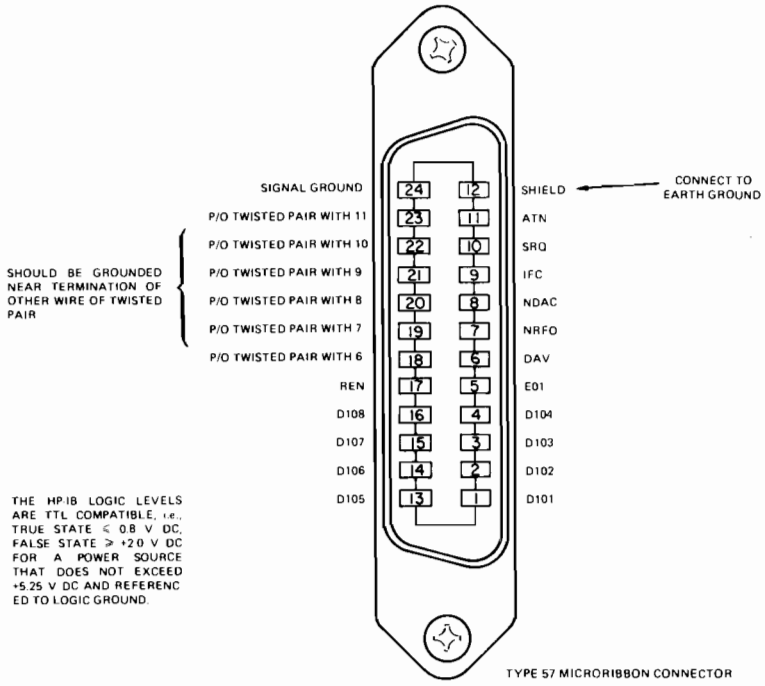


Figure A-1. HP-IB Connector

Of the 16 signal lines, 8 are data lines, 3 are for handshake purposes and the remaining 5 are control lines. Information is transferred across the 8 data lines in a bit-parallel, byte-serial fashion. Briefly, the 8 control and handshake lines are used as follows:

ATN - "Attention" - is used primarily to differentiate between Command mode and Data mode. When ATN is true, information on the data lines is interpreted as a bus command; when ATN is false, the information is treated as a data byte.

EOI - "End or Identify" - has two uses. In the first, EOI is asserted on the last byte of a data transfer. This signals all listening devices that no more data should be expected on the transfer. The second use is in combination with ATN and is used in performing a parallel poll.

IFC - "Interface Clear" - is under the exclusive control of the system controller. When it is pulsed true all device interfaces are returned to an idle state and the state of the bus is cleared.

REN - "Remote Enable" - may be set by the system controller to permit devices to operate in Remote mode, that is, under programmed HP-IB control instead of via the device's front panel.

SRQ - "Service Request" - can be set by a device on the interface to indicate it is in need of service. Examples where SRQ might be set are completion of a task such as taking a measurement, an error detected during device operation, or a request to be active controller.

DAV - "Data Valid" - this handshake line indicates that the active talker has placed data on the data lines (DIO1 - DIO8).

NRFD - "Not Ready for Data" - this handshake line indicates that one or more active listeners is not ready for more data, and the active talker should wait before sending new data on the bus.

NDAC - "Not Data Accepted" this handshake line indicates that one or more active listeners has not accepted the current data byte, and the active talker should leave the current byte asserted on the data lines.

Figure A-2 illustrates the HP-IB structure.

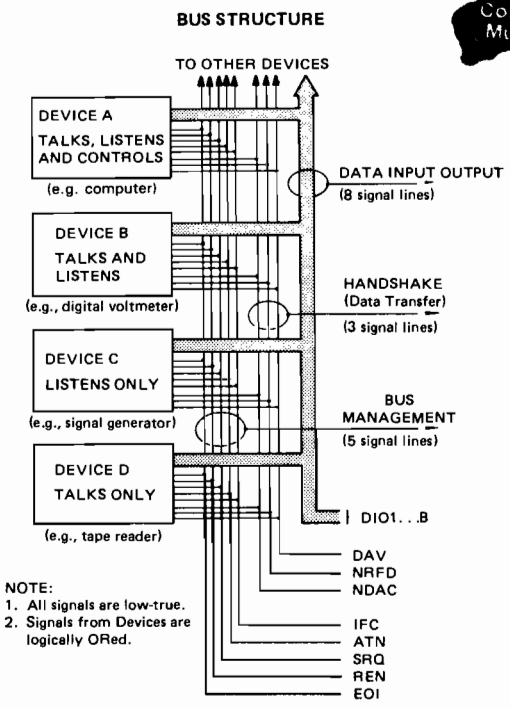


Figure A-2. HP-IB Structure

Commands and Data

There are two modes of communication on an HP-IB: Command mode and Data mode.

In Command mode, information transmitted across the 8 data lines is interpreted as talk or listen addresses, or universal, address or unaddress commands (explained later). In this mode, only 7 of the data lines are used. Some devices use the 8th line as a parity check for certain protocols.

In Data mode, any 8-bit value can be transmitted. The HP-IB can therefore be used for transmission of binary data as well as ASCII characters.

The 3-line handshake scheme has several advantages. First, data transfer is asynchronous - the data rate is limited only by the speed of the devices actively involved in the transfer. A second, related advantage is that devices with different I/O speeds can be interconnected without need for other synchronization mechanisms. Also, multiple devices can be addressed concurrently.

Controllers, Talkers and Listeners

To understand communication among devices, you should be familiar with the concepts of controller, talker and listener.

Controller

There are two types of controller within an HP-IB system: system controller and active controller.

There must be a single *system controller* capable of taking control of the interface at any time. The system controller has exclusive control over the IFC and REN lines.

Each system also has one or more device capable of being *active controller* (sometimes referred to as controller-in-charge), although there may only be one active controller at any given time. The active controller has the ability to establish listeners and talkers, send bus commands, perform serial polls, etc.

In most systems, a single computer will be both the system controller and the only active controller. Some non-system controller devices may request service indicating their desire to be active controller in order to perform some operation such as plotting data or directly accessing disc drives. The current active controller "passes control" to a requesting device to make it the active controller. In other systems a system controller may not be capable of operating as non-active controller, and therefore no pass control capabilities will exist. Note that system controller capabilities may not be transferred.

Talker

In each system there can be at most one device addressed as talker at any given time. A device becomes addressed as talker by receiving its talk address from the active controller. Each device on the bus must have a unique bus address, usually set at the manufacturing site or settable by switches on the instrument.

The addresses are in the range 0 to 30. A talk address is formed by adding the primary bus address to the talk address base of 64 and transmitting that value across the data lines while ATN is asserted. For example, talk address 9 would be formed by taking 64 + 9 = 73, asserting ATN and transmitting a byte whose value is 73 (ASCII character 'I').

Listener

Listen addresses are formed in a similar manner to talk addresses, except that listen addresses use a base of 32 (e.g., listen address 9 is formed as 32 + 9 = 41 = ASCII character ")") transmitted with ATN true.

Multiple devices may be addressed to listen at any time and data bytes will be received by all listeners in parallel. However, most devices cannot be addressed to both talk and listen at the same time.

(Refer to Table A-3 for talk and listen address codes.)

Extended Addressing

The descriptions of talk address and listen address refer to a device's primary address. Some devices also have extended talker or extended listener capabilities, sometimes referred to as secondary addresses. With extended addressing, talk and listen addresses are represented by two command bytes. The first byte is the primary talk or listen address as described above. The second byte is a secondary address command.

Secondary addresses may be in the range 0 to 31. The secondary commands transmitted are formed by adding the secondary address to the base of 96 and transmitting the byte with ATN true.

Extended addresses can be used, for example, to access a specific I/O card within an instrument that allows multiple I/O cards.

Bus Commands

There are five types of information transmitted when the bus is operating in Command mode (i.e. when ATN is asserted):

a. talk address
b. listen address
c. universal commands
d. addressed commands
e. unaddress commands

Talk address and listen address are discussed above.

Universal Commands

Universal commands are received by all responding devices on the bus whether addressed to listen or not. Table A-1 lists the universal commands:

MNEMONIC	COMMAND	DESCRIPTION
LLO	Local Lockout	Disables the front panel of the responding device. The REN line must be asserted in order for LLO to have any effect. If the instrument is already in Remote mode, the lockout will be immediate. Otherwise, the lockout will commence when the device receives its listen address.
DCL	Universal Device Clear	All devices capable of responding are returned to some known, device-dependent state. In some cases a device will perform a self-test in response to a Universal Device Clear.
PPU	Parallel Poll Unconfigure	Directs all devices on the HP-IB that have parallel poll configure capabilities to not respond to a parallel poll.
SPE	Serial Poll Enable	Enables Serial Poll mode on the interface.
SPD	Serial Poll Disable	Disables Serial Poll mode on the interface.

Table A-1. Universal Commands

Addressed Commands

Addressed commands are executed only by those devices that are currently addressed as listeners. They allow the controller to initiate a simultaneous action by a selected group of devices on the bus, such as triggering them to take readings at the same time. Table A-2 lists the addressed commands:

MNEMONIC	COMMAND	DESCRIPTION
SDC	Selected Device Clear	Similar to a Universal Device Clear (DCL) with only those devices addressed to listen responding.
GTL	Go To Local	Returns devices that are addressed to listen to Local mode (i.e. re-enables front panel programming). REN stays asserted when a GTL is sent, and devices will be returned to Remote upon receipt of their listen address.
GET	Group Execute Trigger	Initiates some pre-programmed action by responding devices. This may be used to simultaneously start action in a group of devices that are addressed to listen.
PPC	Parallel Poll Configure	Configures a device to respond to a parallel poll on a specified data line with either a positive or negative signal. A secondary command sent after PPC contains the data that configures the device.
TCT	Take Control	Transfers active controller status to another device on the bus. (Note: This command is not permitted in the HP-IB Command Library for MS-DOS)

Table A-2. Addressed Commands

Unaddress Commands

The two unaddress commands can be considered as extensions of talk and listen addresses. The first, UNL (Unlisten), causes all devices on the bus (except those that have a built-in switch set to Listen Only) to stop being listeners.

Similarly, UNT (Untalk), directs any device on the interface to no longer be addressed as talker. Since there may only be one device addressed to talk at any time, receipt of another device's talk address is equivalent to receiving an UNT. UNL and UNT are logically equivalent to listen address 31 and talk address 31, respectively.

Service Requests

Some devices that operate on the interface have the ability to request service from the system controller. A device may request service when it has completed a measurement, when it has detected a critical condition, or under many other circumstances.

A service request (SRQ) is initiated when the device sets the SRQ line true. The controller, sensing SRQ has been set (typically either by polling the status of the line or by enabling an SRQ interrupt), can poll devices in one of two ways: serial poll or parallel poll.

Serial Poll

A typical sequence of events in performing a serial poll is: establish a device as a talker, send SPE to set up Serial Poll mode, wait for the addressed device to send its serial poll response byte, then send an SPD and UNT to disable the Serial Poll mode.

The meaning of the serial poll response byte depends upon the individual device. However, if bit 6 of the response byte (bit value 64) is 1, the device is indicating it has requested service. If bit 6 is 0, the polled device was not the one that requested service. Individual device manuals provide additional information on the meanings of serial poll response bytes.

Parallel Poll

Parallel polling permits the status of multiple devices on the HP-IB to be checked simultaneously. Each device is assigned a data line (DI01 through DI08) that is set true by the device during the parallel poll routine if it requires service.

More than one device can be assigned to a particular data line. If a shared line is sensed true, a serial poll can typically be performed to determine which device requested service. A parallel poll is started when the controller asserts ATN and EOI together. After a short period of time the controller reads the poll byte and begins its interpretation thereof.

Some devices can be configured (by the PPC command) to respond on specific data lines. Other devices may respond on lines selected by switches or jumpers in the devices. Some devices do not have parallel poll capability.

System Configuration

HP-IB systems can be configured in three ways:

No Controller

This mode of data transfer is limited to a direct transfer between one device manually set to talk only, and one or more devices manually set to Listen Only.

Single Controller

In this configuration, data transfer can be:

- from controller to device(s) (Command or Data mode).
- from device to controller (Data mode only).
- from a device to other device(s) (Data mode).

Multiple Controllers

This mode of data transfer is similar to that of a single controller, with the requirement that active controller status be passable from one controller to another. In this configuration, one controller must be designated as the system controller. This controller is the only one that can control the IFC and REN lines.

Control is passed to another controller by addressing it as a talker, and commanding it to "take control" (TCT).

Note that the HP-IB Command Library does not support more than one controller. Therefore, control may not be passed.

B7 B6 B5 BITS / B4 B3 B2 B1	0 0 0	0 0 1	0 1 0	0 1 1	1 0 0	1 0 1	1 1 0	1 1 1
	CONTROL		NUMBERS SYMBOLS		UPPER CASE		LOWER CASE	
0 0 0 0	0 NUL 0 0	20 DLE 10 16	40 L0 SP 20 32	60 L16 0 30 48	100 T0 @ 40 64	120 T16 P 50 80	140 ` 60 96	160 p 70 112
0 0 0 1	1 GTL SOH 1 1	21 LLO DC1 11 17	41 L1 ! 21 33	61 L17 1 31 49	101 T1 A 41 65	121 T17 Q 51 81	141 a 61 97	161 q 71 113
0 0 1 0	2 STX 2 2	22 DC2 12 18	42 L2 " 22 34	62 L18 2 32 50	102 T2 B 42 66	122 T18 R 52 82	142 b 62 98	162 r 72 114
0 0 1 1	3 ETX 3 3	23 DC3 13 19	43 L3 # 23 35	63 L19 3 33 51	103 T3 C 43 67	123 T19 S 53 83	143 c 63 99	163 s 73 115
0 1 0 0	4 SDC EOT 4 4	24 DCL DC4 14 20	44 L4 $ 24 36	64 L20 4 34 52	104 T4 D 44 68	124 T20 T 54 84	144 d 64 100	164 t 74 116
0 1 0 1	5 PPC ENQ 5 5	25 PPU NAK 15 21	45 L5 % 25 37	65 L21 5 35 53	105 T5 E 45 69	125 T21 U 55 85	145 e 65 101	165 u 75 117
0 1 1 0	6 ACK 6 6	26 SYN 16 22	46 L6 & 26 38	66 L22 6 36 54	106 T6 F 46 70	126 T22 V 56 86	146 f 66 102	166 v 76 118
0 1 1 1	7 BEL 7 7	27 ETB 17 23	47 L7 ' 27 39	67 L23 7 37 55	107 T7 G 47 71	127 T23 W 57 87	147 g 67 103	167 w 77 119
1 0 0 0	10 GET BS 8 8	30 SPE CAN 18 24	50 L8 (28 40	70 L24 8 38 56	110 T8 H 48 72	130 T24 X 58 88	150 h 68 104	170 x 78 120
1 0 0 1	11 TCT HT 9 9	31 SPD EM 19 25	51 L9) 29 41	71 L25 9 39 57	111 T9 I 49 73	131 T25 Y 59 89	151 i 69 105	171 y 79 121
1 0 1 0	12 LF A 10	32 SUB 1A 26	52 L10 ★ 2A 42	72 L26 : 3A 58	112 T10 J 4A 74	132 T26 Z 5A 90	152 j 6A 106	172 z 7A 122
1 0 1 1	13 VT B 11	33 ESC 1B 27	53 L11 + 2B 43	73 L27 ; 3B 59	113 T11 K 4B 75	133 T27 [5B 91	153 k 6B 107	173 { 7B 123
1 1 0 0	14 FF C 12	34 FS 1C 28	54 L12 , 2C 44	74 L28 < 3C 60	114 T12 L 4C 76	134 T28 \ 5C 92	154 l 6C 108	174 ¦ 7C 124
1 1 0 1	15 CR D 13	35 GS 1D 29	55 L13 − 2D 45	75 L29 = 3D 61	115 T13 M 4D 77	135 T29] 5D 93	155 m 6D 109	175 } 7D 125
1 1 1 0	16 SO E 14	36 RS 1E 30	56 L14 . 2E 46	76 L30 > 3E 62	116 T14 N 4E 78	136 T30 ^ 5E 94	156 n 6E 110	176 ~ 7E 126
1 1 1 1	17 SI F 15	37 US 1F 31	57 L15 / 2F 47	77 UNL ? 3F 63	117 T15 O 4F 79	137 UNT — 5F 95	157 o 6F 111	177 RUBOUT (DEL) 7F 127
	ADDRESSED COMMANDS	UNIVERSAL COMMANDS	LISTEN ADDRESSES		TALK ADDRESSES		SECONDARY ADDRESSES OR COMMANDS	

KEY Octal 25 PPU Message
NAK ASCII/ISO character
hex 15 21 decimal

Table A-3. ASCII Codes

B HP-IB Card Installation

The HP-IB Command Library for an HP Vectra (or IBM) - Part No. 61062AA - includes an HP-IB Card. This appendix describes how to configure the card for use with the Library, and how to install it in an HP Vectra or an IBM. If you use an IBM-compatible computer, use this appendix along with your particular computer manual to learn how to install the card.

Handling the Card

- HANDLE GENTLY. Do not drop the card or handle it roughly. Be careful when unpacking the card and during installation.
- HOLD ONLY BY THE EDGES. Never touch any part of the card except the edges. Do not put fingerprints on the connector.
- PROTECT FROM STATIC ELECTRICITY. The interface card can be damaged by static electricity. For protection, the card is packed in an anti-static bag.

  Leave the card in its anti-static bag until you are ready to install it.

  Save the anti-static bag so you can protect the card if you have to remove it from the computer.

Configuring the Card

The HP-IB card configuration switches set the card's operating parameters. They should be set as shown in Figure B-1 to use the Library:

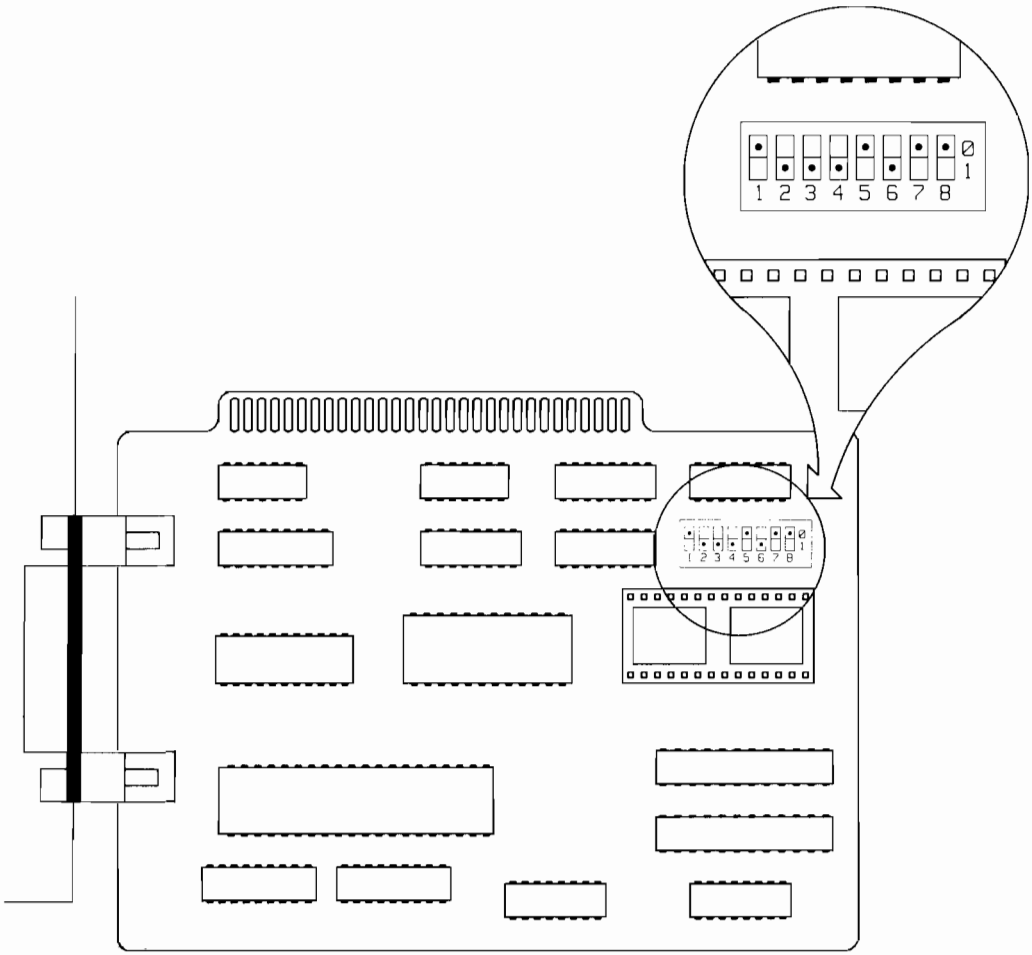
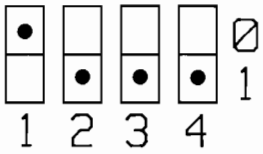


Figure B-1. HP-IB Card Configuration Switches

To check the configuration, hold the card with the edge connector up, then locate the switch block from Figure B-1. Set the switches as described in the following sections.

Card Address

The card address switches (1 - 4) determine the interface select code:



	Switch				
Address	*1*	*2*	*3*	*4*	*Select Code*
C0000	0	0	0	0	16
C4000	0	0	0	1	1
C8000	0	0	1	0	2 - reserved for Fixed Disc Drive Controller
CC000	0	0	1	1	3
D0000	0	1	0	0	4 - reserved for PC Cluster Adapter (LAN)
D4000	0	1	0	1	5
D8000	0	1	1	0	6
DC000	0	1	1	1	7 <————
E0000	1	0	0	0	8
E4000	1	0	0	1	9
E8000	1	0	1	0	10
EC000	1	0	1	1	11
F0000	1	1	0	0	12 - reserved for system ROM
F4000	1	1	0	1	13 - reserved for system ROM
F8000	1	1	1	0	14 - reserved for system ROM
FC000	1	1	1	1	15 - reserved for system ROM

Table B-1. Card Addressing

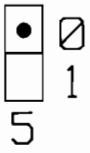
Select a value that does not conflict with any other installed interface card — select code 7 is convenient for HP computers.

If you have multiple HP-IB cards installed, each must be at a separate address. And if one of them is being used as a mass storage interface, it must be the highest addressed card.

Use Table B-1 along with your computer documentation to help you set the card address.

Boot Device

Set switch 5 to zero:



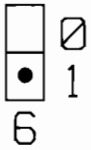
To boot from:	Set Switch 5 to:
- the internal disc drive	0 <————
- an external disc drive	1

Table B-2. Boot Device Switch

This switch selects the boot device which can be an internal or an external disc drive. You must select the internal disc drive as the boot source to use the Library.

System Controller

Set switch 6 to position 1:



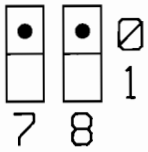
System Controller:	Switch 6:
- not system controller	0
- system controller	1 <————

Table B-3. System Controller

This switch determines whether or not the HP-IB card is the system controller. The card must be system controller to work with the Library.

Interrupt Level

Set switches 7 and 8 to zero:



Interrupt Level	Switch 7	8	
3	0	0	<————
4	0	1	
5	1	0	
6	1	1	

Table B-4. Interrupt Level

These switches let you determine the priority level at which the HP-IB card interrupts the CPU. Interrupts are not supported by the Library, so these switches must be set to zero.

Installing the Card

1. Turn off and unplug your computer.
2. In the case of an HP Vectra PC (or IBM PC/AT), unlock the front cover.
3. Remove the rear panel periphery screws. Then, remove the cover by sliding it forward.

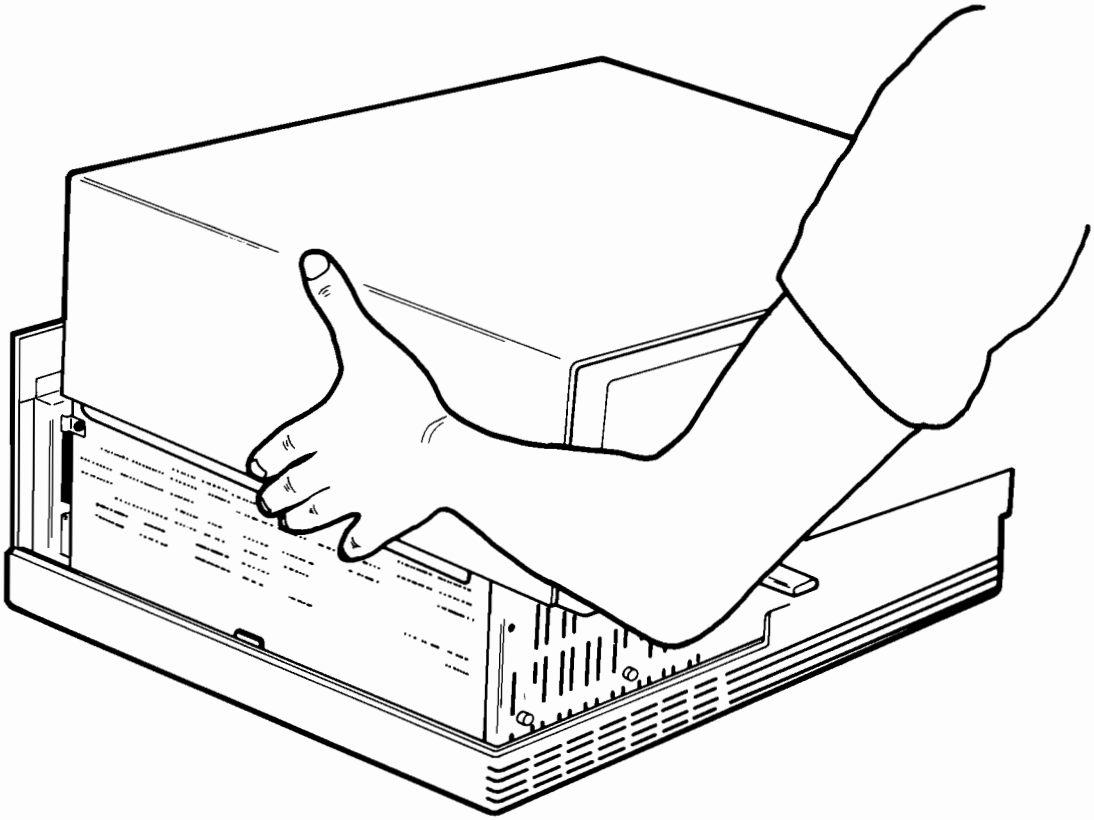


Figure B-2. Removing the Cover

4. Select a suitable empty slot. You can install the card in any slot of an HP Vectra PC or an IBM PC or PC/AT. You can install it in any slot of an IBM PC/XT *except* slot 8.

5. Remove the cover plate of the slot by removing its mounting screw and lifting it from its location (see Figure B-3).

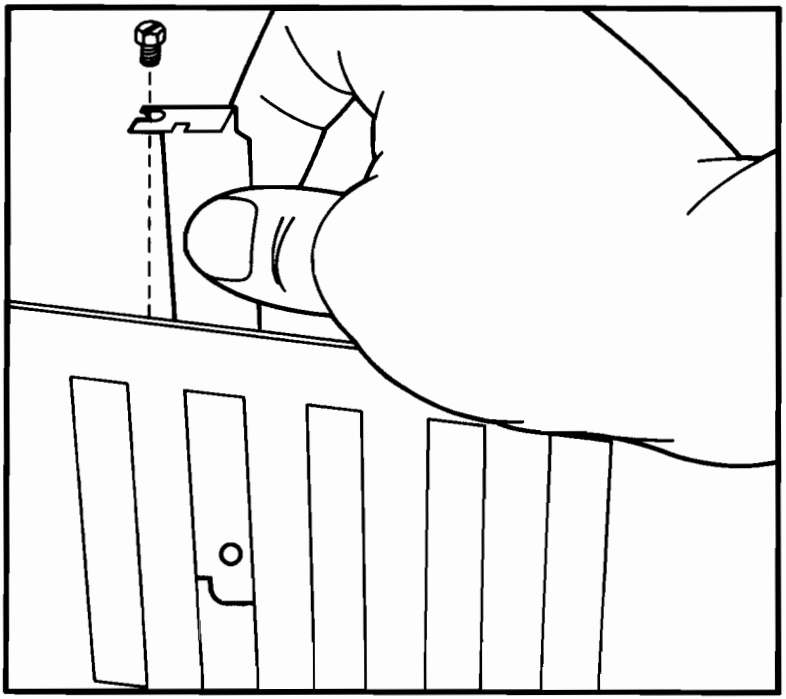


Figure B-3. Removing the Slot Cover Plate

6. Remove the HP-IB card from the anti-static bag, holding it by its edges. Check the settings of the switches on the card (see previous sections).

7. Hold the card with the edge connector down and the cable connector toward the rear of the computer. Insert the plug-in connector into the slot and press the card down firmly to make sure the connector is fully seated. (Figure B-4.)

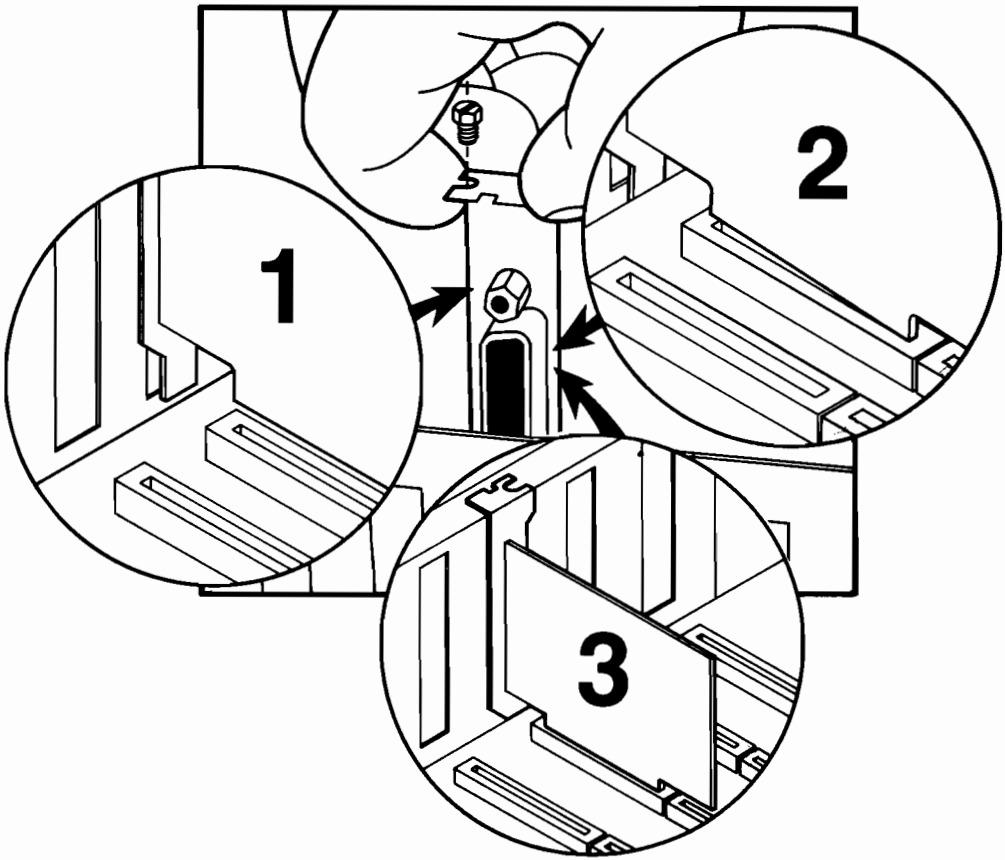


Figure B-4. Installing the Card

8. Replace the slot cover screw to hold the HP-IB card in place. (Store the slot cover for use if the card is later removed from the computer.)

9. Replace the computer cover and install the cover screws.

Connecting the Cable

An HP-IB system can accommodate up to 14 devices in addition to the controller. To set up an instrument and connect it to the interface, follow these instructions:

1. Refer to your particular instrument manual to set up the instrument, connect its power cables, and run its self-test before connecting it to the computer.
2. Turn off the instrument.
3. Determine a bus address for the instrument that does not conflict with the addresses of other devices. Set its HP-IB switches accordingly. (The HP-IB Command Library uses address 30 for the controller address.)
4. Press the HP-IB cable plug into the HP-IB socket on the interface card and tighten the connector screws. The plug and socket fit only one way; if you're having trouble connecting them, rotate the plug 180 degrees and try again.
5. Connect the other end of the HP-IB cable to the HP-IB socket of the instrument. Tighten the connector screws.

Here are some points to remember when connecting instruments to the interface:

- You can connect up to 14 devices to a single interface.
- You can interconnect devices in any scheme as long as there is an unbroken path between each instrument and the controller:

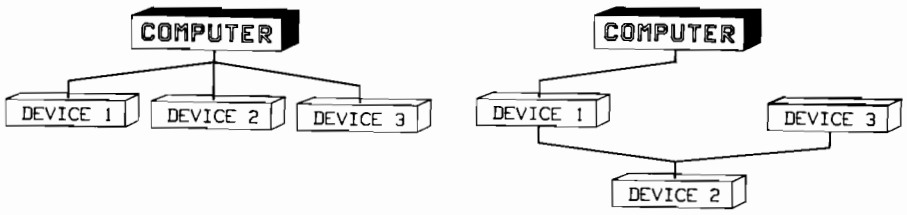


Figure B-5 Connecting Devices

- You can connect HP-IB Cables in piggyback fashion. However, don't stack more than three on a device as the weight of the connectors and cables could damage the socket.

- The total cable length on one interface should not exceed the lesser of 20 meters, or two meters times the number of connected devices (the interface is considered as one device).

C Library Files

This appendix lists the contents of the SETUP.BAS file for Interpretive BASIC programming, and the IODECL.EX and IOPROC.EX files for Pascal Programming.

SETUP.BAS:

```
5   ' Copyright Hewlett-Packard 1984, 1985
10  '
15  ' Set up program for MS-DOS HP-IB I/O Library
20  ' For use independent of the PC instrument bus system
25  '
30  DEF SEG
35  CLEAR ,&HFE00
40  I=&HFE00
45  '
50  ' PCIB.DIR$ represents the directory where the library files
55  '    are located
60  ' PCIB is an environment variable which should be set from MS-DOS
65  '    i.e. A:> SET PCIB=A:\LIB
70  '
75  ' If there is insufficient environment space a direct assignment
80  ' can be made here, i.e
85  '    PCIB.DIR$ = "A:\LIB"
90  ' Using the environment variable is the preferred method
95  '
100 PCIB.DIR$ = ENVIRON$("PCIB")
105 I$ = PCIB.DIR$ + "\PCIBILC.BLD"
110 BLOAD I$,&HFE00
115 CALL I(PCIB.DIR$, I%, J%)
120 PCIB.SEG = I%
125 IF J%=0 THEN GOTO 160
130 PRINT "Unable to load.";
135 PRINT "    (Error #";J%;")"
140 STOP
145 '
```

```
150 ' Define entry points for setup routines
155 '
160 DEF SEG = PCIB.SEG
165 O.S       = 5
170 C.S.      = 10
175 I.V.      = 15
180 I.C.      = 20
185 L.P       = 25
190 LD.FILE   = 30
195 GET.MEM   = 35
200 L.S       = 40
205 PANELS    = 45
210 '
215 ' Establish error variables and ON ERROR branching
220 '
225 DEF.ERR = 50
230 PCIB.ERR$ = STRING$(64,32)
235 PCIB.NAME$ = STRING$(16,32)
240 CALL DEF.ERR(PCIB.ERR,PCIB.ERR$,PCIB.NAME$,PCIB.GLBERR)
245 PCIB.BASERR = 255
250 ON ERROR GOTO 410
255 '
260 J=-1
265 I$=PCIB.DIR$+"\HPIB.SYN"
270 CALL O.S(I$)
275 IF PCIB.ERR<>0 THEN ERROR PCIB.BASERR
280 '
285 ' Determine entry points for HP-IB Library routines
290 '
295 I=0
300 CALL I.V(I,IOABORT,IOCLEAR,IOCONTROL,IOENTER)
305 IF PCIB.ERR<>0 THEN ERROR PCIB.BASERR
310 CALL I.V(I,IOENTERA,IOENTERS,IOEOI,IOEOL)
315 IF PCIB.ERR<>0 THEN ERROR PCIB.BASERR
320 CALL I.V(I,IOGETTERM,IOLLOCKOUT,IOLOCAL,IOMATCH)
325 IF PCIB.ERR<>0 THEN ERROR PCIB.BASERR
330 CALL I.V(I,IOOUTPUT,IOOUTPUTA,IOOUTPUTS,IOPPOLL)
335 IF PCIB.ERR<>0 THEN ERROR PCIB.BASERR
340 CALL I.V(I,IOPPOLLC,IOPPOLLU,IOREMOTE,IORESET)
345 IF PCIB.ERR<>0 THEN ERROR PCIB.BASERR
```

```
350 CALL I.V(I,IOSEND,IOSPOLL,IOSTATUS,IOTIMEOUT)
355 IF PCIB.ERR< >0 THEN ERROR PCIB.BASERR
360 CALL I.V(I,IOTRIGGER,J,J,J)
365 IF PCIB.ERR< >0 THEN ERROR PCIB.BASERR
370 CALL C.S
375 I$=PCIB.DIR$+"\HPIB.PLD"
380 CALL L.P(I$)
385 IF PCIB.ERR< >0 THEN ERROR PCIB.BASERR
390 GOTO 475
395 '
400 ' Error handling routine
405 '
410 IF ERR=PCIB.BASERR THEN GOTO 425
415 PRINT "BASIC error #";ERR;" occurred in line ";ERL
420 STOP
425 TMPERR = PCIB.ERR
430 TMPERR = 0 THEN TMPERR =PCIB.GLBERR
435 PRINT "PC Instrument error #";TMPERR;" detected at line ";ERL
440 PRINT "Error: ";PCIB.ERR$
445 STOP
450 '
455 ' COMMON declarations are needed if your program is going to chain
460 '     to other programs. When chaining, be sure to call DEF.ERR as
465 '     well upon entering the chained-to program
470 '
475 COMMON PCIB.DIR$,PCIB.SEG
480 COMMON LD.FILE,GET.MEM,PANELS,DEF.ERR
485 COMMON PCIB.BASERR,PCIB.ERR,PCIB.ERR$,PCIB.NAME$,PCIB.GLBERR
490 COMMON IOABORT,IOCLEAR,IOCONTROL,IOENTER,IOENTERA,IOENTERS,IOEOI,
IOEOL,IOGETTERM,IOLLOCKOUT,IOLOCAL,IOMATCH,IOOUTPUT,IOOUTPUTA,
IOOUTPUTS,IOPPOLL,IOPPOLLC,IOPPOLLU,IOREMOTE,IORESET,IOSEND,IOSPOLL,
IOSTATUS,IOTIMEOUT,IOTRIGGER
495 '
500 FALSE    = 0
505 TRUE     = NOT FALSE
510 NOERR    = 0
515 EUNKNOWN = 100001!
520 ESEL     = 100002!
525 ERANGE   = 100003!
530 ETIME    = 100004!
```

```
535 ECTRL       = 100005!
540 EPASS       = 100006!
545 ENUM        = 100007!
550 EADDR       = 100008!
555 COMMON FALSE, TRUE, NOERR, EUKNOWN, ESEL, ERANGE, ETIME, ECTRL,
EPASS, ENUM, EADDR
560 '
565 ' End Program Set-up
570 ' User program can begin anywhere past this point
```

IODECL.EX:

```
CONST
  noerr     = 0
  eunknown  = 1;
  esel      = 2;
  erange    = 3;
  etime     = 4;
  ectrl     = 5;
  epass     = 6;
  enum      = 7;
  eaddr     = 8;
  UNL     = chr(63);
  UNT     = chr(95);
  TAD     = chr(64);
  LAD     = chr(32);

TYPE
  xxstr40 = string(40);
  realarray = super array [1..*] of real;
```

IOPROC.EX:

```
procedure errstr (error: integer; var estring: xxstr40);

begin
  case error of
    NoErr:      estring := ' No error                                    ';
    EUnknown:estring := ' Unknown error                                  ';
    Esel:       estring := ' Invalid select code or device address       ';
    ERange:     estring := ' Value out of range                          ';
    ETime:      estring := ' Timeout                                     ';
    ECtrl:      estring := ' HP-IB must be controller                    ';
    EPass:      estring := ' Pass control not permitted                  ';
    ENum:       estring := ' Invalid number                              ';
    EAddr:      estring := ' Improper addressing                         ';
    otherwise   estring := ' What?                                       ';
    end;
end;

function IOABORT (isc: integer4): integer; extern;

function IOCLEAR (dev_or_isc: integer4): integer; extern;

function IOCONTROL (isc: integer4;
                    condition: integer;
                    status: integer): integer; extern;

function IOENTER (dev_or_isc: integer4;
                  VAR realval: real): integer; extern;

function IOENTERA (dev_or_isc: integer4;
                   VAR readings: realarray;
                   VAR elements: integer): integer; extern;

function IOENTERS (dev_or_isc: integer4;
                   VAR str: string;
                   VAR length: integer): integer; extern;

function IOEOI (isc: integer4;
                eoistate: integer): integer; extern;
```

```
function IOEOL (isc: integer4;
                VAR endline: string;
                length: integer): integer; extern;

function IOGETTERM (isc: integer4;
                    VAR reason: integer): integer; extern;

function IOLLOCKOUT (isc: integer4): integer; extern;

function IOLOCAL (dev_or_isc: integer4): integer; extern;

function IOMATCH (isc: integer4;
                  matchchar: char;
                  enable: integer): integer; extern;

function IOOUTPUT (dev_or_isc: integer4;
                   realval: real): integer; extern;

function IOOUTPUTA (dev_or_isc: integer4;
                    VAR values: realarray;
                    elements: integer): integer; extern;

function IOOUTPUTS (dev_or_isc: integer4;
                    VAR outstr: string;
                    length: integer): integer; extern;

function IOPPOLL (isc: integer4;
                  VAR response: integer): integer; extern;

function IOPPOLLC (dev_or_isc: integer4;
                   configuration: integer): integer; extern;

function IOPPOLLU (dev_or_isc: integer4): integer; extern;

function IOREMOTE (dev_or_isc: integer4): integer; extern;

function IORESET (isc: integer4): integer; extern;
```

```
function IOSEND (isc: integer4;
                 VAR commands: string;
                 length: integer): integer; extern;

function IOSPOLL (dev_or_isc: integer4;
                  VAR response: integer): integer; extern;

function IOSTATUS (isc: integer4;
                   condition: integer;
                   VAR status: integer): integer; extern;

function IOTIMEOUT (isc: integer4;
                    timeout: real): integer; extern;

function IOTRIGGER (dev_or_isc: integer4): integer; extern;
```

D

HP-IB/PCIB Programming Example

Introduction

You can combine HP-IB compatible instruments with HP PC Instruments in the same BASIC application and control them from your personal computer. This appendix presents an example program that controls a PC Instruments function generator and an HP-IB voltmeter. The program is run on an HP Series 150 computer.

The example includes commands from the HP-IB Command Library and programming statements from the PC Instruments Library. A description of each line is provided to help you understand the program. Refer to the HP-IB Command Library and PC Instruments manuals for details on programming with these products.

(The HP-IB Command Library for an HP Series 150 is part number 14857AA. The PC Instruments users' manual for an HP Series 150 is part number 61060-90001.)

Note: For brevity, the example program has no error checking lines. You should include these, however, in your applications.

Programming Example

This example program tests op amps with an HP 61014A PC Instruments Function Generator and an HP 3456A Digital Voltmeter (HP-IB compatible). It checks the response of an op amp to a 2V peak-to-peak signal, swept from 1 kHz to 10 kHz.

Hardware Configuration

Figure D-1 illustrates the hardware configuration required for the example. Connect the HP 61014A PC Instruments Function Generator to the computer via the PC Instruments Interface card which must be installed in the computer. (Complete installation instructions for PC Instruments including power connections, PC Instruments bus connections, and address switch settings are given in Chapter 2 of the PC Instruments manual.)

Connect the HP 3456A Digital Voltmeter (DVM) to the HP-IB connector on the rear of the computer with an HP-IB cable. Complete installation instructions for the voltmeter including power connections, bus connections, and HP-IB address switch settings are given in the Operating Manual for the HP 3456A DVM.

To simulate the fixture/op amp combination, use a coaxial cable with meter lead adapters to connect the signal (OUT 50 ohms) from the Function Generator to the voltmeter's VOLTS input.

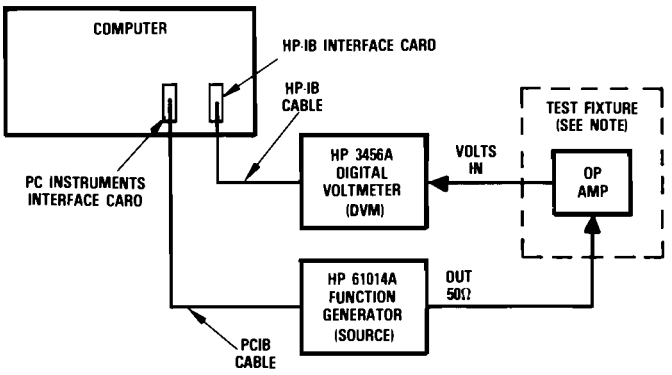


Figure D-1 Hardware Configuration

This example uses a select code of 7, and a DVM device address of 22. (To run the example, make sure that your hardware address settings agree with the values you use in the program.)

Copying HP-IB Command Library Files

Before you can write a BASIC program for this configuration, you must copy two files (HPIB.SYN and HPIB.PLD) from the HP-IB Command Library disc onto your PC Instruments work disc. The work disc must already contain the PCIB subdirectory (Refer to Chapter 3 of the PC Instruments manual).

The following steps assume the PC Instruments disc is in drive A, and the HP-IB Command Library disc is in drive B. If you use different drives, substitute them in the procedure.

To copy the files:

1. Boot up the system.

2. Insert the HP-IB Library disc into drive B.

3. Use file manager from PAM to copy the files HPIB.SYN and HPIB.PLD from B: to A:\PCIB.

Creating the Program Shell

After you copy the HP-IB Library files onto your PC Instruments work disc, create a program shell to accommodate the DVM and the function Generator:

1. Run the Configure program described in Chapter 3 of the PC Instruments manual. This displays the System Configuration menu on the computer screen.

2. Answer the questions in the menu by making the appropriate entries from the keyboard. Answer "yes" to the HP-IB Option question.

3. Run the PANELS program and set up your PC Instrument. Use the Soft Front Panel and Soft Rear Panel displays on the screen. (This is described in Chapter 3 and 4 of the PC Instruments manual, and in Chapter 3 of the Function Generator manual.)

This example assumes you have assigned the label "SOURCE" on the Soft Rear Panel. It also assumes you did not change any of the Function Generator's default settings (Function = Sine, Frequency = 1.0 Khz, Offset = OV, Amplitude = 10 mV, Symmetry = 50%, Mode = Continuous, and Output = Disabled) on the Soft Front Panel. See Chapter 3 in the Function Generator Manual for details.

4. Use Soft Front Panel system softkeys to save the program shell as "OPAMP". (See Chapter 4 of the PC Instruments manual.) Then, exit the PANELS program.

Writing the BASIC Program

First, load the program shell OPAMP into the BASIC workspace as described in Chapter 5 of the PC Instruments manual. Then, write your BASIC program by inserting PC Instruments programming statements, HP-IB commands, and BASIC statements. Be sure to preface each PC Instrument statement and HP-IB command with CALL.

Begin your application program at line 1000. Remember that the label SOURCE was assigned to the Function Generator and must be used in your PC Instrument programming statements.

Note

If the PC Instrument programming statements or HP-IB commands fail to execute, you may have a program error. Refer to Chapter 5 of the PC Instruments manual, or Chapter 2 of the HP-IB Command Library manual for more information on errors and error handling.

First, Initialize some working variables:

```
1000 'User program starts at this line.
1010 '
1020 CODES$=SPACE$(50)
1030 MIN.FREQ=1000
1040 MAX.FREQ=10000
1050 STEP.FREQ=1000
1060 AMPL=2
1070 '
```

1020 Initialize the string (CODES$) to hold a sufficient number of HP-IB instrument commands.

1030 Set the minimum (or start) frequency to 1 KHz.

1040 Set the maximum (or stop) frequency to 10 KHz.

1050 Define frequency increment (or step frequency) as 1 kHz.

1060 Define the SOURCE output amplitude of 2V peak-to-peak.

Define the HP-IB address variables:

```
1080 ISM=7
1090 DVM=722
1100 '
```

1080 Set the HP-IB select code variable ISC to 7.

1090 Set the Digital Voltmeter HP-IB address variable DVM to 722.

Define some initialization characteristics:

```
1110 CALL IORESET(ISC)
1120 TIMEOUT=5
1130 CALL IOTIMEOUT(ISC,TIMEOUT)
1140 ENABLE=1
1150 CALL IOEOI(ISC,ENABLE)
1160 CALL IOREMOTE(ISC)
1170 CALL IOCLEAR(ISC)
1180 '
```

1110 Reset the interface to clear all previous activity.

1120/1130 Define an HP-IB timeout of 5 seconds.

1140/1150 Execute the IOEOI command with ENABLE of 1. When OUTPUTing, this enables the controller to set the EOI line true on the last byte of the write cycle. When ENTERing, it enables the read cycle to terminate when the EOI line is sensed true.

1160 Set the interface Remote Enable line (REN); this places any device on the bus in the Remote mode as soon as it's addressed to listen.

1170 Set all HP-IB instruments into a known device-dependent state.

Program the PC Instruments signal source:

```
1190 CALL SET.MODE(SOURCE,CONTINUOUS)
1200 CALL SET.FUNCTION(SOURCE,SINE)
1210 CALL SET.AMPLITUDE(SOURCE,AMPL)
1220 '
```

1190 Set the SOURCE to output a waveform continuously.

1200 Set the output waveform to be a sine wave.

1210 Set the amplitude of the output sine wave to be 2V peak-to-peak.

Program the Voltmeter:

```
1230 CODES$="H F2 R4 Z0 T3"
1240 LENGTH=LEN(CODES$)
1250 CALL IOOUTPUTS(DVM,CODES$,LENGTH)
1260 '
```

1230 Define the programming codes string CODES$ to hold:

H - software reset the voltmeter
F2 - select the AC volts function
R4 - select the 10 volt range
Z0 - turn off auto zero
T3 - select single trigger.

1240 Determine the length of the programming codes string for use in the next line.

1250 Send the programming codes string to the voltmeter.

Take a measurement:

```
1270 CALL ENABLE.OUTPUT(SOURCE)
1280 '
1290 FOR FREQ=MIN.FREQ TO MAX.FREQ STEP STEP.FREQ
1300 CALL SET.FREQUENCY(SOURCE,FREQ)
1310 CALL IOTRIGGER(DVM)
1320 CALL IOENTER(DVM,READING)
1330 PRINT FREQ,READING
1340 NEXT FREQ
1350 '
1360 END
```

1270 The output sine wave from the SOURCE is started.

1290 A FOR/NEXT loop starts to increment the SOURCE output frequency from 1 kHz to 10 kHz in 1 kHz steps.

1300 Set the output frequency of the SOURCE.

1310 Trigger the DVM to take a reading.

1320 Get the reading from the DVM.

1330 Print each frequency and the measured voltage

1340 Complete the FOR/NEXT loop.

1360 End the program.

You can save the program on a work disc and run it as described in Chapter 5 of the PC Instruments manual.

E Binary Data Transfer

Some instruments represent data in a form other than ASCII characters and digits. This is known as binary data. The HP-IB Command Library lets you enter binary data in the computer as character strings. But without further processing, it may still be unusable. In this case, you can use the ASC() function in BASIC and the ORD() function in Pascal to convert the data to a suitable form.

Depending on the particular instrument, individual characters in the binary data stream may have special meanings. The instrument may code data as two byte (16 bit) binary numbers, four byte binary numbers, hexadecimal numbers, octal numbers, etc.

For example, assume an instrument sends the value 31614 to the computer. This could be represented as the binary number 0111011001111110.

Decimal:	Binary:
31614	0111011001111110

This would be entered into the computer with the IOENTERS command as ASCII characters { ~ .

Binary:	ASCII Characters:	Decimal equivalent:
0111011001111110	{ ~	123 126

You could convert these characters back into a number with the ASC() function in BASIC. This would return the value 123 for the first character and 126 for the second. Then, by multiplying the place value of 256 (2^8) times the first character (123), you get 31488. Adding the second character (126) results in 31614, the original number. You can apply similar conversions to other binary formats.

The following example reads binary data from an HP 3457 digital multimeter. It converts the data into two different formats. The first is a conversion from binary to decimal. The second is a conversion to a scaled decimal number using a scaling factor provided by the meter.

The program enters the data as an ASCII string. It then converts the individual characters to their decimal equivalent with the ASC() function. Finally, it displays the results.

Note that BASIC for MS-DOS allows the transfer of only 255 bytes to a string variable. If your binary data includes longer strings, you must use MS-Pascal or enter the data into several strings.

```
1000 ISC=7
1010 DVM=722
1020 MAXLEN=4
1030 CODES$=SPACE$(100)
1040 '
1050 CALL IORESET (ISC)
1060 TIMEOUT=5
1070 CALL IOTIMEOUT (ISC,TIMEOUT)
1080 CALL IOREMOTE(ISC)
1090 CALL IOCLEAR(ISC)
1100 IF PCIB.GLBERR <> NOERR THEN ERROR PCIB.BASERR
1110 '
1120 'Set DVM for 30k Ohms Range, double integer binary output
1130 CODES$="OHM 3.1E3;OFORMAT DINT;TARM SYN"
1140 LENGTH=LEN(CODES$)
1150 CALL IOOUTPUTS(DVM,CODES$,LENGTH)
1160 IF PCIB.ERR <> NOERR THEN ERROR PCIB.BASERR
1170 '
1180 'Get Binary Data from DVM
1190 ACTLEN=0
1200 BIN$=SPACE$(20) 'allocate room for data
1210 CALL IOENTERS(DVM,BIN$,MAXLEN,ACTLEN)
1220 IF PCIB.ERR <> NOERR THEN ERROR PCIB.BASERR
1230 '
1240 PRINT "String entered of length: ";ACTLEN
```

```
1250  '
1260  'Convert string to number
1270  DAT=0
1280  FOR I=1 TO ACTLEN
1290  DAT=DAT*256
1300  DAT=ASC(BIN$)+DAT
1310  LENGTH =LEN(BIN$)
1320  BIN$=RIGHT$(BIN$,LENGTH-1)
1330  NEXT I
1340  '
1350  PRINT "Data =";DAT
1360  '
1370  'Get the Scale Factor from the DVM
1380  CODES$="ISCALE?"
1390  LENGTH=LEN(CODES$)
1400  CALL IOOUTPUTS(DVM,CODES$,LENGTH)
1410  IF PCIB.ERR <> NOERR THEN ERROR PCIB.BASERR
1420  '
1430  CALL IOENTER(DVM,SCALE)
1440  IF PCIB.ERR <> NOERR THEN ERROR PCIB.BASERR
1450  '
1460  DAT=DAT*SCALE
1470  PRINT "Scaled Data=";DAT
1480  PRINT
1490  GOTO 1190 'Get next reading.
1500  END
```

Index

Worldwide Field Repair Centers

International

ARGENTINA

BUENOS AIRES
Hewlett-Packard Argentina S.A.
Avda Santa Fe 2035
Martinez, 1640
Phone: /792-1293

AUSTRALIA

New South Wales
SYDNEY
Hewlett-Packard Australia Ltd.
17-23 Talavera Road
North Ryde,
New South Wales 2113
Phone: 02 /888-4444

Victoria
MELBOURNE
Hewlett-Packard Australia Ltd.
31-41 Joseph Street
Blackburn, Victoria 3130
Phone: 03 /895-2895

AUSTRIA

VIENNA
Hewlett-Packard GmbH
Lieblgasse 1
Posifach 72
Vienna, A-1222
Phone: 222/3516210

BELGIUM

BRUSSELS
Hewlett-Packard Belgium SA/NV
Boulevard de la Woluwe 100
Woluwedal 100
Brussels, B-1200
Phone: 2 /762-3200

BRAZIL

SAO PAULO
Hewlett-Packard Brasil I.e.C.
Alameda Rio Negro 750
Alphaville
Barueri, SP-06400
Phone: 011/421-1311

CANADA

Alberta
EDMONTON
Hewlett-Packard Canada Ltd.
11120 178th Street
Edmonton, Alberta T5S 1P2
Phone: 403/486-6666

Alberta
CALGARY
Hewlett-Packard Canada Ltd.
3030 3rd Avenue. N.E.
Calgary, Alberia T2A 6T7
Phone: 403/235-3100

British Columbia
VANCOUVER
Hewlett-Packard Canada Ltd.
10691 Shellbridge Way
Richmond, British Columbia
V6X 2W8
Phone: 604/270-2277

Ontario
TORONTO WEST
Hewlett-Packard Canada Ltd.
5877 Goreway Drive
Mississauga, Ontario l4V 1M8
Phone: 416/678-9430

DENMARK

COPENHAGEN
Hewlett-Packard A/S
Datavej 52
Birkeroed, DK-3460
Phone: 2 /81-66-40

FINLAND

HELSINKI
Hewlett-Packard Oy
Revontulentie 7
Espoo, SF-02100
Phone: 90 /455-0211

JYVASKYLA
Hewlett-Packard Oy
Valnonkatu 9c
Jyvaskyla, SF-40100
Phone: 41 /216318

OULU
Hewlett-Packard Company
Kainuuntieic Oulu, SF-90140
Phone: 81 /338785

FRANCE

ORSAY
Hewlett-Packard France
Les Ulis Avenue De Tropiques
Z. Industrielle de Courtaboeuf
Les Ulis, F-91947
Phone: 6 /9077825

GERMANY

BOEBLINGEN
Hewlett-Packard GmbH
Herrenberger Strasse 110
Boeblingen, D-7030
Phone: 703/667750

HONG KONG

HONG KONG
Hewlett-Packard Hong Kong Ltd.
5 Floor Sun Hung Kai Centre
30 Harbour Road
Wanchai
Phone: 5 /832-3211

ITALY

MILANO
Hewlett-Packard Italiana S.p.A
Via G. Di Vittorio 9
Cernusco Sui Navigl, I-20063
Phone: 2 /903691

JAPAN

Kanagawa
SAGAMIHARA
Yokogawa-Hewlett-Packard Ltd.
27-15 Yabe
Sagamihara, Kanagawa 229
Phone: 427/59-1311

Osaka-Shi
OSAKA
Yokogawa-Hewlett-Packard Ltd.
Chuo Bldg. Nishinakajima
4-20, Yodogawa-ku
Osaka, Osaka-Shi 532
Phone: 6 /304-6021

Suginami-ku
TOKYO
Yokogawa-Hewlett-Packard Ltd.
29-21 Takaido-Higashi
Tokyo, Suginami-ku 168
Phone: 3 /331-6111

MEXICO

MEXICO CITY
Hewlett-Packard Mexicana SACV
Avenida Periferico Sur 6501
Tepepan, Xochimilco
Mexico City, DF16020
Phone: 905/676-4600

NETHERLANDS

AMSTELVEEN
Hewlett-Packard Nederland B.V.
Van Heuven Goedhartlaan 121
Amstelveen, NL-1181KK
Phone: 20 /472021

NORWAY

OSLO
Hewlett-Packard Norge A/S
Oesterndalen 16
Oesteraas, N-1345
Phone: 2 /17-11-80

SINGAPORE

SINGAPORE
Hewlett-Packard Singapore Ltd.
6th Floor, Inchcape House
450-452 Alexandra Road
Singapore, 9115
Phone: 5 /631-788

SOUTH AFRICA

Transvaal
JOHANNESBURG
Hewlett-Packard South Africa
Private Bag Wendywood
Sandton, Transvaal 2144
Phone: 11 /802-511

SPAIN

MADRID
Hewlett-Packard Espanola S.A.
Cosia Brava, 13-2
Mirasierra Madrid, E-34
Phone: 91 /734-1162

SWEDEN

STOCKHOLM
Hewlett-Packard Sverige AB
Skalholtsgatan 9, Kista
Box 19 Spanga, S-16393
Phone: 8 /750-2000

SWITZERLAND

ZUERICH
Hewlett-Packard (Schweiz) AG
Allmend 2 Widen, CH-8967
Phone: 57 /312111

TAIWAN

TAIPEI
Hewlett-Packard Taiwan Ltd.
8th Floor
337 Fu-Hsing North Road
Taipei
Phone: 2 /712-0404

UNITED KINGDOM

England
MANCHESTER
Hewlett-Packard Ltd.
Trafalgar House Navigation Road
Alltrincham, England WA14 1NU
Phone: 61 /928-6422

WINNERSH
Hewlett-Packard Ltd.
King Street Lane
Winnersh
Wokingham, England RG11 5AR
Phone: 734/784-774

VENEZUELA

CARACAS
Hewlett-Packard Venezuela C.A.
Edificio Segre 1, 2, & 3
3a Transversal Los Ruices Nort
Caracas, 1071
Phone: 2 /239-4133

United States

California
LOS ANGELES WEST
Hewlett-Packard Company
5400 W. Rosecrans Blvd.
Lawndale California 90260
Phone: 213/643-7500

SANTA CLARA
Hewlett-Packard Company
3003 Scott Boulevard
Santa Clara, California 95050
Phone: 408/988-7000

Colorado
DENVER
Hewlett-Packard Company
24 Inverness Place, East
Englewood, Colorado 80112
Phone: 303/649-5000

Georgia
ATLANTA
Hewlett-Packard Company
2000 South Park Place
Atlanta, Georgia 30339
Phone: 404/955-1500

Illinois
CHICAGO WEST
Hewlett-Packard Company
5201 Tollview Drive
Rolling Meadows, Illinois 60008
Phone: 312/255-9800

Maryland
WASHINGTON D.C.
Hewlett-Packard Company
2 Choke Cherry Road
Rockville, Maryland 20850
Phone: 301/948-6370

Massachusetts
ANDOVER
Hewlett-Packard Company
1775 Minuteman Road
Andover, Massachusetts 01810-0906
Phone: 617/682-1500

Michigan
DETROIT
Hewlett-Packard Company
39550 Orchard Hill Drive
Novi, Michigan 48050
Phone: 313/349-9200

New Jersey
PARAMUS
Hewlett-Packard Company
W120 Century Road
Paramus, New Jersey 07652
Phone: 201/265-5000

Pennsylvania
PHILADELPHIA
Hewlett-Packard Company
Valley Forge Corporate Center
2750 Monroe Boulevard
Valley Forge, Pennsylvania 19482
Phone: 215/265-7000

Texas
DALLAS
Hewlett-Packard Company
930 East Campbell Road
Richardson, Texas 75801
Phone: 2144/231-6101

Washington
SEATTLE
Hewlett-Packard Company
15815 S.E. 37th Street
Bellevue, Washington 98006
Phone: 206/643-4000

Headquarter Offices

HEADQUARTERS OFFICES

If there is no office listed for your area contact one of these headquarters offices.

AFRICA AND MIDDLE EAST

Hewlett-Packard S.A.
Mediterranean and Middle East
Operations Atrina Centre
32 Kifissias Ave.
Paradissos-Amarousion,
ATHENS Greece
Tel: 682 88 11
Telex: 21-6588
HPAT GR
Cable: HEWPACKSA Athens

NORTH/CENTRAL AFRICA

Hewlett-Packard S.A.
7, Rue du Bois-du-Lan
CH-1217 **MEYRIN** 2,
Switzerland
Tel: (022) 83 12 12
Telex: 27835 hpse
Cable: HEWPACKSA
Geneve

ASIA

Hewlett-Packard Asia Ltd.
47/F, 26 Harbour Rd. Wanchai,
HONG KONG
G.P.O. Box 863, Hong Kong
Tel: 5-8330833
Telex: 767793
HPA HX
Cable: HPASIAL TD

CANADA

Hewlett-Packard (Canada) Ltd.
6877 Goreway Drive **MISSISSAUGA,**
Ontario L4V 1M8
Tel: (416) 678-9430
Telex: 610-492-4246

EASTERN EUROPE

Hewlett-Packard Ges.m.b.h.
Liebgasse 1 P.O. Box 72
A-1222 **VIENNA,** Austria
Tel: (222) 2365110
Telex: 1 3 4425
HEPA A

NORTHERN EUROPE

Hewlett-Packard S.A.
Uilenstede 475 P.O. Box 999
NL-1180 AZ **AMSTELVEEN**
The Netherlands
Tel: 20 437771

SOUTH EAST EUROPE

Hewlett-Packard S.A.
World Trade Center 110 Avenue
Louis Carol 1215 Cointrin,
GENEVA, Switzerland
Tel: (022) 98 96 5 1
Telex: 27225 hpse.

EASTERN USA

Hewlett-Packard Co.
4 Choke Cherry Road
ROCKVILLE, MD 20850
Tel: (301) 258-2000

MIDWESTERN USA

Hewlett-Packard Company
5201 Tollview Drive
ROLLING MEADOWS, IL 60008
Tel: (312) 255-9800

SOUTHERN USA

Hewlett-Packard Company
2000 South Park Place
P.O. Box 105005
ATLANTA, GA 30348
Tel: (404) 955-1500

WESTERN USA

Hewlett-Packard Company
3939 Lankershim Blvd.
P.O. Box 3919
LOS ANGELES, CA 91604
Tel: (213) 506-3700

Other International Areas

Hewlett-Packard Company
Intercontinental Headquarters
3495 Deer Creek Road
PALO ALTO, CA 94304
Tel: (415) 857-1501
Telex: 034-8300
Cable: HEWPACK