About this Manual

We've added this manual to the Agilent website in an effort to help you support your product. This manual is the best copy we could find; it may be incomplete or contain dated information. If we find a more recent copy in the future, we will add it to the Agilent website.

Support for Your Product

Agilent no longer sells or supports this product. Our service centers may be able to perform calibration if no repair parts are needed, but no other support from Agilent is available. You will find any other available product information on the Agilent Test & Measurement website, www.tm.agilent.com.

HP References in this Manual

This manual may contain references to HP or Hewlett-Packard. Please note that Hewlett-Packard's former test and measurement, semiconductor products and chemical analysis businesses are now part of Agilent Technologies. We have made no changes to this manual copy. In other documentation, to reduce potential confusion, the only change to product numbers and names has been in the company name prefix: where a product number/name was HP XXXX the current name/number is now Agilent XXXX. For example, model number HP8648A is now model number Agilent 8648A.

HP 8711C/12C/13C/14C/30A LAN Interface

HP part number: 08712-90062
Printed in USA August 1998 Supersedes October 1997

Notice

The information contained in this document is subject to change without notice. Hewlett-Packard makes no warranty of any kind with regard to this material, including but not limited to, the implied warranties of merchantability and fitness for a particular purpose. Hewlett-Packard shah not be liable for errors contained herein or for incidental or consequential damages in connection with the furnishing, performance, or use of this material.

Key Conventions

This manual uses the following conventions:

Front-Panel Key — This represents a key physically located on the instrument (a "hardkey").

Softkey — This indicates a "softkey, " a key whose label is determined by the instrument's firmware, and is displayed on the right side of the instrument's screen next to the eight unlabeled keys.

Firmware Revision

This manual documents analyzers with firmware revisions C.04.52 and above. Some features will not be available or will require different keystrokes in analyzers with earlier firmware revisions. For full compatibility, you can upgrade your firmware to the latest version. Contact your nearest Hewlett-Packard sales or service office for information.

Acknowledgments

Portions of the TCP/IP software are copyright Phil Karn, KA9Q.

GIF output routines are by John Silva (derived from Jef Poskanzer's PBMplus package).

Java™ is a U.S. trademark of Sun Microsystems, Inc.

Lotus@ 1-2-3® are U.S. registered trademarks of Lotus Development Corporation.

Microsoft@ is a U.S. registered trademark of Microsoft Corp.

MS@ and MS-DOS@ are U.S. registered trademarks of Microsoft Corporation.

MS Windows@, Windows@, Windows 95®, and Windows NT@ are U.S. registered trademarks of Microsoft Corporation.

Netscape is a U.S. registered trademark of Netscape Communications Corporation.

Pentium® is a U.S. registered trademark of Intel Corporation.

Postscript™ is a trademark of Adobe Systems Incorporated which may be registered in certain jurisdictions.

Reflection™ is a U.S. trademark of Walker, Richer & Quinn, Inc.

UNIX@ is a registered trademark in the United States and other countries, licensed exclusively through X/Open Company Limited.

Introducing the LAN Interface

The Option 1F7 LAN Interface lets you connect your analyzer to an Ethernet network that uses TCP/IP. With the LAN interface you can:

- Save and recall the analyzer's settings (instrument state).
- Transfer IBASIC programs between your computer and your analyzer.
- Transfer files between your computer and analyzer using FTP.
- Connect many analyzers to one computer.
- Automate the control of your analyzers.
- Program the analyzer using SCPI commands.
- Print hardcopy directly to an HP LaserJet printer.
- Access your analyzer's Web pages. Here you'll find links to:
  - o general information about the HP 87xx family of analyzers
  - o online documentation such as SCPI command references
  - o specific information about your analyzer such as your current firmware revision, installed options, even the analyzer's current screen image.
  - o general information about Hewlett-Packard, and how to obtain assistance if you need it.

Expert Assistance

Most sites will hire a network administrator to install the LAN hardware and manage the assignment of IP addresses.

> NOTE
>
> Hewlett-Packard offers professional consulting services to help increase your computer and manufacturing productivity. HP system engineers will work with your factory management, engineering, and production groups to evaluate various automation solutions. For more information contact the nearest HP sales office. Refer to Chapter 7 for a table of Hewlett-Packard sales and service offices.

Documentation Outline

This *User's Guide Supplement* describes how to connect, use and troubleshoot the LAN interface on your analyzer. This supplement contains the following chapters:

1. **Connecting and** Configuring the Analyzer	Provides information about connecting the analyzer to the network. To effectively use this chapter, you should be familiar with your network setup and operation.
2. **Accessing** the Analyzefs Web **Pages**	Describes how to use a Web browser to access built-in Web pages.
3. Printing	Describes how to configure-and dump a hardcopy to-a network printer.
4. Accessing the **Analyzer's** File System	Describes how to access the analyzer's file system using file transfer protocol (FTP). The directory structure of the analyzer is described here.
5. Accessing the **Analyzer's** Dynamic Data Disk	Describes the analyzer's "data' directory, the dynamic data disk. Includes an example program.
6. Controlling the Analyzer via the LAN	Shows you methods for programming the analyzer via the network connection.
7. General Troubleshooting	Describes what to do if you have a problem using the analyzer on your network.
Glossary	Definitions for networking and other terms used in this book.

Contents

Figures

Tables

1

Connecting and Configuring the Analyzer

Connecting and Configuring the Analyzer

This chapter describes:

- how to connect your analyzer to your network
- how to set up a network
- how to configure your analyzer
- how to verify connectivity

In order to complete the steps in this chapter, you'll need:

- Option 1F7 installed in your analyzer
- A computer with a LAN interface, running an operating system that supports TCP/IP. For example: UNIX@ or Microsoft Windows 95®. A typical computer would be an IBM-compatible Pentium®-based PC with a 10Base-T LAN card, or an HP 5210 PA-RISC workstation.
- A computer program that talks over TCP/IP. This might be the built-in FTP or telnet program, or a program that you write. This will be covered in detail in the following chapters.
- LAN cabling, and typically a LAN hub.

If you only wish to dump hardcopy to a LaserJet printer, you'll need:

- Option 1F7 installed in your analyzer
- An HP LaserJet printer with an HP JetDirect LAN interface card
- LAN cabling, and typically, a LAN hub

Note to Novell Netware users

Older versions of Novell Netware used IPX networking protocol exclusively IPX protocol is not compatible with TCP/IP protocol.

Newer versions of Novell Netware, such as version 3.1x and 4.xx allow for add-on products which provide a gateway to a TCP/IP network. Consult your Novell network administrator for the latest information on using Novell Netware with TCP/IP protocol.

Connecting the Analyzer to the LAN

Your analyzer has an RJ-45 connector (see Figure l-l), and connects to your network using 10Base-T twisted pair cabling, also called Ethertwist. Ethertwist cables resemble a standard modular phone line.

> NOTE
>
> If your network uses ThinLAN (10Base-2), you will need to purchase an adapter which converts the ThinLAN BNC connector to 10Base-T Ethertwist.

To connect the analyzer to your network:

1. Turn off the analyzer.
2. Connect the Ethertwist cable from your network to the LAN ETHERTWIST port on the rear of your analyzer and then turn on the analyzer.

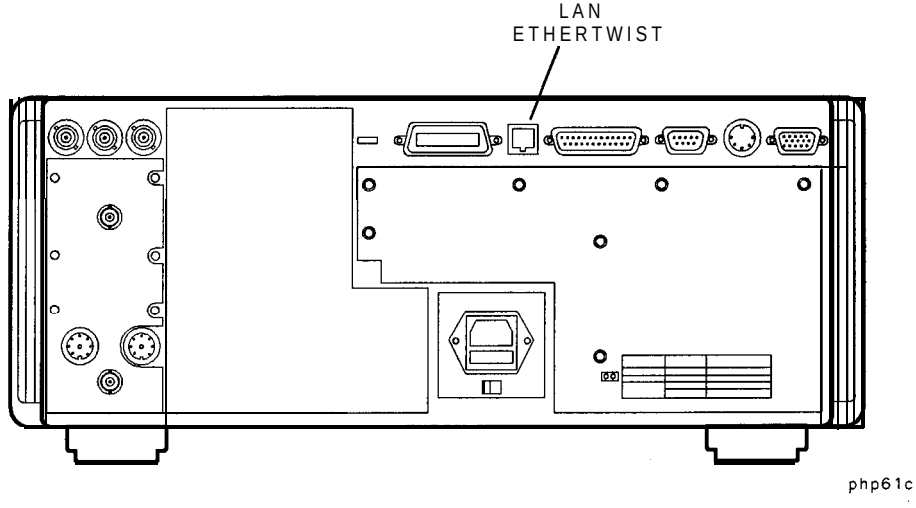


Figure 1-1. The LAN ETHERTWIST Port

If you do not already have a network, you will need to create one. A simple network consists of a central LAN hub with multiple Ethertwist cables, one connected to the LAN port of each network device. This is often called a star topology, with the LAN hub at the center.

Typical 8-port hub:

HP J2610B AdvanceStack 10Base-T Hub-8U

Typical 16-port hub:

HP J2611B AdvanceStack 10Base-T Hub-16U

Typical Ethertwist cables:

HP 92268A Twisted-pair "straight-through" cable, 4 meters

HP 92268B Twisted-pan "straight-through" cable, 8 meters

HP 922686 Twisted-pair "straight-through" cable, 16 meters

HP 92268D Twisted-pair "straight-through" cable, 32 meters

HP 92268N Twisted-pair "straight-through" cable, 300 meters

lb order cables, contact the nearest Hewlett-Packard sales or service office. See Chapter 7 for a table of sales and service offices.

Point-to-Point Connections

It is possible to connect a single computer to a single analyzer, and avoid using a LAN hub. lb do this, you must use a special "cross-over" cable or adapter, which acts like a LAN hub. If you try to create a point-to-point connection using a standard "straight through" cable, it will not work. For most applications, use of a LAN hub is simpler, and provides expandability.

NOTE

Point-to-point connections may not work when connecting to older laser printers. Older printers typically require a boot server for network use. To use a point-to-point connection with a printer, use an HP LaserJet 4 or newer.

Configuring the Analyzer

Before you configure your analyzer, you will need to contact your network administrator to obtain an IP address for your analyzer. You will also want to ask your network administrator about a unique hostname for your analyzer, a gateway IP address, and a subnet mask.

The Analyzer's IP Address and Hostname

Each device on your network must have a unique address so that all devices can communicate simultaneously over the same network. These unique addresses are called IP addresses, and are assigned by your network administrator. IP addresses are a set of 4 decimal numbers, separated by periods.
For example: 192.170.128.21

You may also receive (or request) from your network administrator a hostname for your analyzer.
For example: **my87** 11

The hostname is not required, but can be used on your computer so that you don't have to remember the IP address. Typically, the hostname is found in the hosts file on your computer or returned by a name server.

Your network administrator will apply for a range of IP addresses from the Internet Network Information Center (InterNIC). InterNIC is responsible for registering domain names and assigning TCP/IP network numbers to networks that connect to the Internet. You may contact InterNIC via e-mail at hostmaster@internic.net, or by accessing their Web site at http://rs.internic.net/.

CAUTION

It is important that no two devices are assigned the same IP address, or else conflicts will occur.

The Gateway Address

If your analyzer will be communicating with devices on different physical networks, you may need to have your network administrator assign a gateway IP address for you. The gateway IP address is the address of a routing device that connects your analyzer's LAN with other LANs.

NOTE

See "Subnets and Gateways" in Chapter 7 for more information on gateway addresses.

The Subnet Mask

If your analyzer will be communicating with devices on different physical networks, you may need to have your network administrator assign a subnet mask number for you. The subnet mask tells your analyzer whether a remote device is on the same LAN as your analyzer. If your analyzer is attempting to communicate with another device, the subnet mask defines whether your analyzer needs to route communications through the gateway.

NOTE

See "Subnets and Gateways" in Chapter 7 for more information on subnet masks.

The Ethernet Address

Your analyzer has a unique built-in Ethernet address associated with the LAN hardware inside the analyzer. The Ethernet address is a 48-bit number assigned at the factory. The Ethernet address cannot be changed, and you should not have to be concerned about it. In this document, the term "LAN address" refers to the IP address.

To Configure the Analyzer

1. Press (SYSTEM OPTIONS) LAN to access the LAN menu.

> NOTE
>
> After each of the following steps, the analyzer will prompt you to cycle power for the new setting to take effect. It is not necessary to cycle the power after each step. It only needs to be done **once**— when you are finished entering all of the settings.

2. Set the **LAN State** setting to ON if it isn't already. If you do not wish to use your analyzer's LAN, you can turn this to OFF. When OFF, networking is disabled, and more memory is available to IBASIC and the volatile RAM disk.
3. Press **LAN Port Setup 87xxx IP Address** , and enter the IP address that your network administrator assigned to your analyzer. You may have also received a hostname (for example: my8711). You cannot enter the hostname into your analyzer, just the IP address. The hostname can be used on your computer, so that you don't have to remember the IP address.
4. Press **Gateway IP Address** , and enter the numbers assigned to you by your network administrator. If you were not assigned a gateway IP address, leave the setting at 0.0.0.0 (default value) to disable gateway routing.
5. Press **Subnet Mask** , and enter the numbers assigned to you by your network administrator. If you were not assigned a subnet mask, leave the setting at 0.0.0.0 (default value) to disable subnet masking.
6. Once you have entered these settings, cycle the power on your analyzer, so that it can re-initialize its LAN interface with these new values.

Verifying Connectivity

You should now verify connectivity between your computer and your analyzer.

The ping utility is typically used to test connectivity.

Running Ping under Windows 95

At the command prompt of a DOS window, type:

```
ping <IPaddress>
```

or

```
ping <hostname>
```

where **<IPaddress>** is the number you entered into your analyzer in the steps above, and **<hostname>** is the hostname assigned to your IP address. For example, type ping **my87** 11.

You should see something like this:

```
Pinging my8711 [15.4.43.5] with 32 bytes of data:

Reply from 15.4.43.5: bytes=32 time=37ms TTL=252
Reply from 15.4.43.5: bytes=32 time=30ms TTL=252
Reply from 15.4.43.5: bytes=32 time=30ms TTL=252
Reply from 15.4.43.5: bytes=32 time=31ms TTL=252
```

If you see something like this:

```
Request timed out.
Request timed out.
Request timed out.
Request timed out.
```

your connection may have a problem. Refer to Chapter 7 for troubleshooting help and information.

Running Ping under UNIX

The ping program is typically found in the "/etc" or "/usr/etc" directory, so you must add the appropriate directory to your PATH, or type the full path:

```
/etc/ping <IP address> 64 5
```

or

```
/etc/ping <hostname> 64 5
```

This command tells ping to send 5 packets of 64 bytes each.

The output should look something like this:

```
PING hostname: 64 byte packets
64 bytes from 15.4.43.5: icmp_seq=0. time=8. ms
64 bytes from 15.4.43.5: icmp_seq=1. time=4. ms
64 bytes from 15.4.43.5: icmp_seq=2. time=4. ms
64 bytes from 15.4.43.5: icmp_seq=3. time=3. ms
64 bytes from 15.4.43.5: icmp_seq=4. time=3. ms

---- hostname PING Statistics----
5 packets transmitted, 5 packets received, 0% packet loss
round-trip (ms) min/avg/max = 3/4/8
```

If you do not see any output after about 20 seconds, interrupt the ping command using ^C (hold down the "Ctrl" key, and press "c"). Once you do this, ping should provide some statistics on how many packets were sent and received. If the statistics look like this:

```
---- hostname PING Statistics----
4 packets transmitted, 0 packets received, 100% packet loss
```

your connection may have a problem. Refer to Chapter 7 for troubleshooting help and information.

2

Accessing the Analyzer's Web Pages

Accessing the Analyzer's Web Pages

Your analyzer has built-in Web pages that are accessible with a Web browser such as Netscape or Microsoft@ Internet Explorer. These Web pages contain links to general product information, selected online documentation, benchmarks, specific information about your analyzer, and general Hewlett-Packard sites. You can also E-mail us with your comments and feedback on the HP 87xx family of analyzers.

Before you can access your analyzer with a Web browser, you need to connect and configure your analyzer as described in Chapter 1.

Accessing the Analyzer with your Web Browser

lb access your analyzer, start your Web browser and connect to "http://my8711", where "my8711" is the hostname that has been assigned to your analyzer.

When you are connected to your analyzer, a Web page will appear with the following information on it:

- Get a current screen snapshot.
- Examine your analyzer's conhguration.
- Browse selected product documentation.
- Review the Product Summary.
- Other links

Click on the hyperlinks (any underlined words) to browse through the analyzer's pages.

The rest of this chapter explains some of the areas you can browse in further detail.

Screen Snapshot

Clicking on "Get a current screen snapshot" shows an exact copy of your analyzer's current screen image. Using your Web browser's "reload" function assures you of having the most current image.

CAUTION

The screen image takes a few seconds to load. Do not push any buttons on the analyzer or send any programming commands to it while the snapshot is loading, or an inaccurate image may result.

NOTE

Before capturing the screen image with your Web browser, you may wish to customize the look of the image using the **Color Options** menu on your analyzer. (See your analyzer's *User's Guide* for more information.) In particular, you may want to choose **Inverse Video** to create a white background, especially if you plan to print the page from your Web browser.

Analyzer Configuration

Clicking on "Examine your analyzer's configuration" brings up a screen of information that is equivalent to pressing (SYSTEM OPTIONS) **Service Instrument Info** on the analyzer. This screen provides the model and serial number of your analyzer, Ermware revision, installed options, and amount of memory.

Product Documentation

This section provides selected portions of your analyzer's documentation online, as well as benchmark information and information about product upgrades and options.

The following list shows the links currently available on this page:

- Optimizing your Measurements
- Accessing Built-in Disks
- Controlling I/O Ports

- Accessing the Analyzer's file system via the LAN
- Accessing the Dynamic Data Disk via the LAN
- Controlling the Analyzer via the LAN

- HP 871xC SCPI command reference
- HP 8730A SCPI command reference
- IEE 488.2 common commands

- Product Upgrades and Options

- Transfer Speeds using HP-IB
- Transfer Speeds using LAN
- Printing Speed

- List of printed manuals

If there are additional portions of the analyzer's documentation that you think would be helpful to have online, please contact us via E-mail. Click on "Contact HP" and "Send us your feedback!' from your analyzer's Web page.

Product Overview

The links in this area provide generic information about the HP 871xC and HP 8730A family of analyzers. New features, compatibility issues, and available options are among the information provided here.

Other Links

At the bottom of every Web page in the analyzer, you'll find the following links:

Top	takes you to the top of the current page.
Search	takes you to the "Product Documentation" page.
Contact HP	takes you to a page that provides links to HP Web sites, as well as the opportunity to provide HP with feedback on your analyzer and its documentation.
Copyright	takes you to copyright information.

3

Printing

Printing

Your analyzer can send hardcopy output directly to an HP LaserJet printer on your network. In order to print to a LAN printer, your analyzer must have the LAN state turned on, it must have its IP address set, and it must be connected to your network. Refer to Chapter 1 if you have not yet configured your analyzer as described.

Compatible Printers

The HP LaserJet 4 and HP LaserJet 5 families of printers are compatible with your analyzer for printing directly via a point-to-point connection or over your network. These newer printers allow you to enter the printer's IP address directly from the front panel and do not require a boot server computer. Your printer should have a JetDirect LAN card installed.

Some older printers, such as an HP LaserJet III, do not allow you to enter an IP address from the front panel. They require a boot server computer on the network that configures (sets) the printer's IP address.

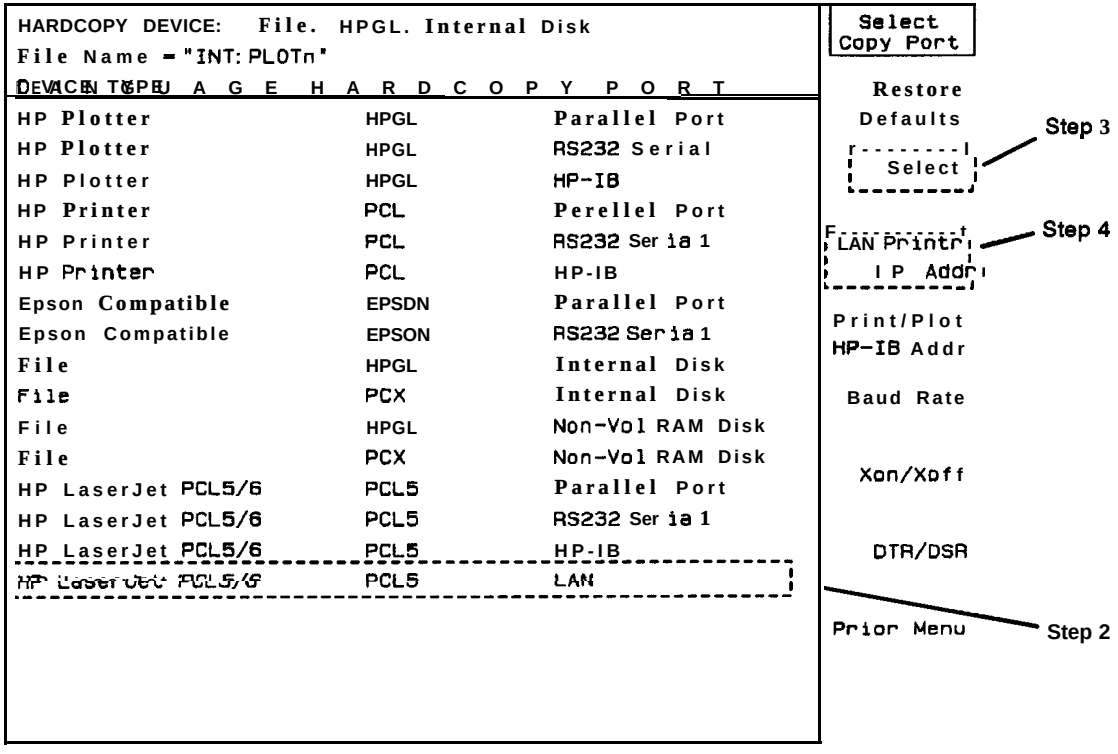
Configuring the Printer

Refer to your printer's documentation for instructions on how to set up your printer for LAN usage. Typically, you will need to contact your network administrator to assign a unique IP address for your printer. Your printer software will configure the printer with the assigned IP address each time it is turned on.

Configuring the Analyzer for Printing to a LAN Printer

To set up your analyzer to print hardcopies to a LAN printer:

1. Press (HARD COPY) Select **Copy** Port .
2. Use the front panel knob, or the ⇧ ⇩ keys to highlight the LaserJet LAN printer in the table. See Figure 3-1.
3. Press **Select** . See Figure 3-1.



php85c

Figure 3-1. Selecting and Configuring the LAN Printer

4. Press **LAN Printr IP Addr** . See Figure 3-1. Enter the IP address of the network printer you wish to use. Use the **Clear Entry** key to clear the current or default setting, and then input the IP address using the analyzer's numeric keypad. (You can also use a keyboard connected to the rear panel DIN KEYBOARD connector to enter the IP address.)
5. Press **Prior Menu** and use the **Def ine PCL5** key to set up the printer configuration, and the **Define Hardcopy** key to define the output. See your analyzer's User's *Guide* for information on configuring printers and defining output.

NOTE

You can print color screen dumps if you send the output to an HP Color LaserJet or HP Color LaserJet 5 printer. Press **Define PCL5 Color** .

6. After you have completed the previous steps, you can send a hardcopy to your LAN printer by simply pressing (HARD COPY) **Start** .

If You Have Trouble Printing

- Make sure the LAN state is set to ON (see Chapter 1).
- Make sure the analyzer's LAN IP address has been set (see Chapter 1).
- Make sure the printer is configured properly. Refer to your printer's documentation or your network administrator.
- Verify connectivity to printer using the analyzer's built-in ping diagnostic utility (see Chapter 7).

4

Accessing the Analyzer's File System

Accessing the Analyzer's File System

This chapter shows you how to access the analyzer's file system using file transfer protocol (FTP). This chapter also contains two simple examples: one for copying a file to the analyzer from your computer, and one for copying a file from the analyzer to your computer. The last section of this chapter contains a summary of commonly used ftp commands.

This chapter assumes that your analyzer is physically connected to your local area network. If it is not connected, refer to Chapter 1 for information on how to connect the system.

When you access the analyzer, you will have read and write access to the analyzer's file system (except for some files in the dynamic "data" disk, which are described in Chapter 5).

CAUTION

It is possible to have multiple FTP sessions open to your analyzer simultaneously. Files can be corrupted when more than one session is accessing the analyzer at a time. Take care to not have more than one session accessing the analyzer's file system at a time.

This caution also applies to file system access performed via SCPI commands over the LAN, HP-IB, or IBASIC.

CAUTION

There is no password protection built into the analyzer. This means that files are not protected against either deletion or being written over.

Using FTP to Access the Analyzer

If you are using a UNIX workstation, you have built-in networking software that includes ftp. The same is true if you are operating under Windows 95. If you are operating under Windows 3.1, you will need to have additional networking software that includes ftp.

> NOTE
>
> There are versions of FTP programs available with a graphical user interface (GUI). See "Using GUI FTP Software" later in this chapter for information on using these types of programs.

To access the analyzer's file system using ftp:

1. Enter the following command on your computer or workstation:

   ftp **<hostname>**

   or

   ftp **<IP** address>

   For example, type f tp **my87** 11.

2. When the connection is made, you will be prompted for a login name. Just press the Return or Enter key on your computer.
3. If you are prompted for a password, just press the Return or Enter key on your computer. There is no password protection built into your analyzer.
4. You should now have a prompt on your computer display that looks like this:

   ftp>

5. Type dir at the prompt. Your computer display should return something that looks like this:

```
200 Port command okay
150 Opening data connection for LIST /
drwxrwxrwx   2 root      sys          1024 Oct 9   1997 int
drwxrwxrwx   2 root      sys          1024 Oct 9   1997 nvram
drwxrwxrwx   2   root    sys          1024 Oct 9   1997 ram
drwxrwxrwx   2 root      sys          1024 Oct 9   1997 data
226 File sent OK
```

   You can read and write files to:

   int-a DOS formatted floppy disk in the analyzer's 3.5" floppy disk drive

   nvram-the analyzer's internal non-volatile memory

   ram-the analyzer's internal volatile memory

   The data directory is a dynamic data disk with files that are linked directly to analyzer operations. See Chapter 5 for information on accessing and using this directory.

6. Use the examples in this chapter to copy a file to the analyzer and to retrieve a file from the analyzer. Also see "Commonly Used FTP Commands" in this chapter for commonly used ftp commands.

Example 1: Copy a File to the Analyzer

You can copy files from your computer to your analyzer. For instance, you may want to develop an IBASIC program on your computer and then copy it to the analyzer so that you can run it from the front panel of the analyzer.

This example copies a file, "ib_prog", from your computer to the analyzer's nvram disk.

1. On your computer or workstation, change directories to the directory that contains the file "ib_prog."
2. On your computer or workstation, access the analyzer by typing ftp **hostname**. Press Return or Enter twice (remember you don't need to enter a login name or password).
3. Change to the non-volatile RAM disk in the analyzer by typing cd nvram at the ftp prompt.
4. Specify the type of file you will be transferring by typing either binary or ascii at the ftp prompt.

CAUTION

Binary files can be corrupted if you attempt to transfer them in "ascii" mode.

5. Type put ib,prog at the ftp prompt.
6. Type bye at the ftp prompt to exit ftp.

You can can now recall and run the program from the front panel of your analyzer.

1. Press [SAVE RECALL] **Select Disk Non-Vol RAM Disk** .
2. Press **Prior Menu Programs** . Use the front panel knob to highlight the **IB_PROG** file.
3. Press Recall **Program Run** .

NOTE

You can also download and automatically run IBASIC programs by accessing the data disk. See Chapter 5 for information on accessing this disk.

Copying Files from UNIX

When copying files from a UNIX environment to the analyzer, files that do not meet the DOS file-naming criteria (≤8 characters in filename, with ≤3 characters in extension) will be truncated to comply For example, if You copy a file from UNIX named "ibasic_program.abcd" it will appear as "ibasic_p.abc" on the analyzer. There will be no feedback or indication from ftp that this has occurred.

Example 2: Retrieve a File from the Analyzer

You can copy files from your analyzer to your computer. For instance, you may want to retrieve saved measurement data from your analyzer (or a group of analyzers) for statistical analysis on your computer. In another scenario, you may have automated your measurement system with an IBASIC program saving data and/or instrument states to the analyzer's RAM disk. Your remote computer could, asynchronously, be copying and then deleting the files from the RAM disk, assuring backup of data, and preventing the RAM disk from filling up.

You may also want to copy instrument states and calibrations to your computer as a backup, eliminating the need for backups on floppy disks.

This example copies a file "STATE2.STA" from your analyzer's nvram disk to a directory on your computer or workstation.

1. On your computer or workstation, access the analyzer by typing ftp hostname. Press Return or Enter twice (remember you don't need to enter a login name or password).
2. Change to the non-volatile RAM disk in the analyzer by typing cd nvram at the ftp prompt.
3. If necessary, use the **lcd** command to change to the local directory on your computer where you want to put the file. For example: type **lcd /users/myname/871x_data**.
4. Specify the type of file you will be transferring by typing either binary or ascii at the ftp prompt.

CAUTION

Binary files can be corrupted if you attempt to transfer them in "ascii" mode.

5. Type get **state2.** sta at the ftp prompt.
6. Type bye at the ftp prompt to exit ftp.
7. Verify the file was copied by listing the contents of the directory it was copied to.

Commonly Used FTP Commands

The exact commands you use within ftp depend on the software. If you are not familiar with your ftp software, type "?" or "help" at the ftp prompt to see a list of commands.

The following table lists and provides a brief description of some commonly used ftp commands.

Table 4-1. FTP Commands

Command	Description
ascii	Sets the file transfer type to ASCII.
binarv	Sets the file transfer **type** to binarv.
bye	Closes the connection to the host and exits ftp.
cd *remote_directory*	Sets the working directory on the host to *remote_directory.*
delete *remote_file*	Deletes *remote-fib* or empty *remote-direaory.*
dir [*remote_directory*]	lists the contents of the specified *remote_directory.* If *remote_directory* is unspecified, the contents of the current remote directory ara listed.
get *remote_file* [*local_file*]	Copies *remote_file* to *local_file.* If *local_file* is unspecified, ftp uses the *remote_file* name as the *local_file* name.
help	Provides a list of ftp commands.
help *commend*	Provides a brief description of *commend.*
imaoe	Sets the file transfer tvoe to binarv.
lcd [*local_directory*]	Sets the local working directory to *local_directory.*
ls [*remote_directory*]	lists the contents of the specified *remote_directory.* If the *remote_directory* is unspecified, the contents of the current remote directory **are** listed.
mget [*remote_files*]	Copies multiple *remote_files* to the local system. Supports metacharacters such as **"*".**
mput [*local_files*]	Copies multiple *local_files* to the remote system. Supports metacharacters such as **"*".**
put *local_file* *[remote-fib]*	Copies *local_file* to *remote file.* If *remote-fib* is unspecified, ftp uses the *local_file* name es the *remote_file* name.
quit	Closes the connection to the host and exits ftp.

Using GUI FTP Software

There are versions of FTP programs available with a graphical user interface (GUI). These programs can make transferring files between the analyzer and your PC a simple "drag and drop" function.

> NOTE
>
> The procedures in this section were developed using Reflection™ FTP for Windows NT®. They are intended as examples only Other GUI FTP software may not be able to understand the analyzer's directory format, and will probably have different steps.

Example: Transferring Files between the Analyzer and your PC

This example copies a file, "ib-prog" , from your computer to the analyzer's nvram disk.

1. Start the FTP program and configure as follows:
   - Set View to Split Window. (View both the command window and the normal window.)
   - Under the Options menu, set Server Directory Format to List Filenames Only.
2. Type your analyzer's hostname in the Server Name box.
3. Click on Open.

4. You will be prompted for a user name. Since the analyzer does not require this (but the program does), you can type anything in the box. One character will suffice.
5. You will be prompted for a password. This is not required. Just click on OK.
6. To change to the non-volatile RAM disk in the analyzer, click inside the command window and then type cd nvram at the **ftp>** prompt.
7. Use the Client side of the window to change directories on your PC to the directory that contains the file "ib_prog".
8. Click on the file "ib_prog" and "drag" it over to the Server side of the window and "drop" it.
9. The file has been transferred to the non-volatile RAM disk on your analyzer.
10. lb drag and drop multiple files, just hold down the Ctrl key on your PC while selecting files with the mouse. Then when you drag and drop, your entire selection will be transferred.
11. You can also transfer files from the analyzer to your computer by dragging files in the other direction.

CAUTION

Be sure to use the appropriate file transfer method (binary or ASCII) for the file(s) you are transferring. If you are transferring files to or from the analyzer's dynamic data disk, check Table 5-1 for file types.

5

Accessing the Analyzer's Dynamic Data Disk

Accessing the Analyzer's Dynamic Data Disk

Your analyzer has an ftp directory called "data," which is a dynamic data disk. The files in this directory trigger analyzer operations. So, for example, you can put another analyzer's instrument state into this directory and the analyzer will automatically recall this state. You can do the same with an IBASIC program: copy it to the analyzer's data directory and it will automatically run. You can also pull a screendump from the analyzer directly into a file in either PCX or HP-GL format.

The following files make up the contents of the dynamic data disk:

Table 5-1. The Dynamic Data Disk Contents

File	File Type	Description
state.sta[1]	binary	This file contains the analyzer's currant instrument state settings. Instrument state settings consist of all the stimulus and response parameters that sat up the analyzer to make a specific measurement including markers, limit lines, and memory traces. Instrument state information is saved and recalled for both measurement channels. You can either retrieve this information from the analyzer, or you can put another analyzer's instrument state information into this file, which will cause the analyzer to immediately enter the new instrument state.
cal.sta[1]	binary	This file contains the analyzer's currant calibration and instrument state settings. The measurement calibration information is the measurement correction data that the analyzer creates when you make a calibration. Measurement calibration information is saved and recalled for both measurement channels. You can either retrieve this information from the analyzer, or you can put another analyzer's calibration and instrument state information into this file, which will cause the analyzer to immediately enter the new cal and instrument state.
data.sta[1]	binary	This file contains the measurement data for both measurement channels. You can either retrieve this information from the analyzer, or you can put data trace information from another analyzer into this file.
tset_cal.cal[1]	binary	*For* use *with 87075C* rest *sets only.* This file contains the test set calibration data that currently resides on the analyzer's non-volatile RAM disk. You can either retrieve this information from the analyzer, or you can put test sat calibration data into this file.

1 See 'Saving and Recalling Analyzer States' in this chapter for information on how to use this file

Table 5-1. The Dynamic Data Disk Contents (continued)

File	File Type	Description
prog.bas[1,2]	ASCII	This file contains the currently loaded IBASIC program. You can either retrieve the program that is currantlv in this file or copy a new program to this file.
prog_run.bas[1,2]	ASCII	This file accepts a copy of an IBASIC program and immediately runs it.
prog_run.scp[1,2]	ASCII	This file accepts a copy of a file containing SCPI commands and immediately executes the commands.
screen.hgl[3]	ASCII	This file contains the current screen image in HP-GL format. It is available for uploading to a file on vour comoutar.
screen.pcx[3]	binary	This file contains the currant screen image in PCX format. It is available for uploading to a file on your computer.
screen_m.hgl[3]	ASCII	This file contains the current screen image, as wall as the currant softkey menu, in HP-GL format. It is available for uploading to a file on your computer.
screen_m.pcx[3]	binary	This file contains the currant screen imaga, as well as the currant softkey menu, in PCX format. It is available for uploading to a file on your computer.
screen_m.gif[3]	binary	This file contains the currant screen image, as well as the currant softkey menu, in GIF format. It is available for uploading to a file on your computer.
parm_all.txt[4]	ASCII	This file contains a listing of all of the instrument's operating parameters in ASCII text format.
parm_screen.txt[4]	ASCII	This file contains the information in the current operating parameters screen in ASCII text format.
trace1.prn[5]	ASCII	This file contains the measurement channel 1 measurement date in ASCII spreadsheet format.
trace2.prn[5]	ASCII	This file contains the measurement channel 2 maesurament data in ASCII spreadsheet format.
trace1.s1p[5]	ASCII	This file contains the measurement channel 1 measurement data in Touchstone format.
trace2.s1p[5]	ASCII	This file contains the measurement channel 2 measurement data in Touchstone format.

1 See "Copying Programs to and from the Analyzer" in this chapter for information on how to use this file

2 Your analyzer must have IBASIC, Option 1C2, to use this file

3 See "Copying a Screen Image to a Local File" in this chapter for information on how to use this fila

4 See "Copying Instrument Parameters in ASCII Text Format" in this chapter for information on how to use this file.

5 See "Retrieving Measurement Data in ASCII Format" in this chapter for information on how to use this file.

This section describes how to use the state. sta, cal. sta, and data. sta files that reside in the data directory of the analyzer. See Table 5-1 for a brief description of each of these files.

You may have a particular instrument state set up on an analyzer and would like to set up that state on one or more additional analyzers. lb do this you should:

1. Connect to the analyzer using ftp. For example, type f tp **analyzer1** on your computer or workstation. See Chapter 4 for instructions on how to do this.
2. Type cd data at the ftp prompt.
3. Type dir at the ftp prompt to see the listing of files in this directory, as well as a short description of each of them.
4. Type binary at the ftp prompt to specify a binary file transfer.
5. Type get state. sta at the prompt to retrieve the current instrument state file from the analyzer. This copies the file state.sta (which contains the analyzer's current instrument state information) to your computer.
6. Close the connection and exit ftp by typing bye or quit at the prompt.
7. Now you can put the instrument state into a different analyzer. Connect to the desired analyzer using ftp. For example, type ftp **analyzer2** on your computer or workstation.
8. Type cd data at the ftp prompt.
9. Type put state. sta at the ftp prompt. This copies the contents of the state.sta file from your computer to the new analyzer you are connected to. The new analyzer will immediately reinitialize itself with the new instrument state.

The above procedure can be performed with the cal . sta and data. sta files as well.

NOTE

It is possible to have saved an instrument state file from the front panel of the analyzer that contains not only the instrument state settings, but the current calibration and measurement data as well. Putting this one file into the state. sta file will cause the analyzer to recall instrument state, cal state, and measurement data.

CAUTION

When transferring these files between instruments with different model numbers and/or option configurations, it is possible that some instrument state settings will not be compatible. For example, if you try to put an instrument state with a stop frequency of 3 GHz into an HP 8711C, the HP 8711C will limit the frequency to 1.3 GHz (its high end limit). When you transfer this file over ftp, you will not receive any warning or indicator that this has occurred.

Copying Files from UNIX

When copying files from a UNIX environment to the analyzer, files that do not meet the DOS file-naming criteria ($\leq$8 characters in filename, with $\leq$3 characters in extension) will be truncated to comply. For example, if you copy a file from UNIX named "ibasic_program.abcd" it will appear as "ibasic_p.abc" on the analyzer. There will be no feedback or indication from ftp that this has happened.

Copying Programs to and from the Analyzer

This section describes how to use the prog. bas, **prog_run.** bas, and prog,run. scp files that reside in the data directory of the analyzer. See Table 5-1 for a brief description of each of these files.

> NOTE
>
> Your analyzer must have Option 1C2 (IBASIC) to use these program files.

> See Also
>
> Refer to Chapter 6 for information on controlling the analyzer from a computer, and how to interact with an IBASIC program running in the analyzer.

To Copy an IBASIC Program to or from the Analyzer

If you have Option 1C2 (IBASIC) in your analyzer, your analyzer has a built-in controller. You can create IBASIC programs on your computer and copy them to your analyzer. Conversely, you can retrieve a copy of the currently loaded IBASIC program from your analyzer to your computer. From there you might want to copy it into another analyzer, or edit it.

To copy an IBASIC program file named "ib_prog" to the analyzer:

1. Connect to the analyzer using ftp. For example, type f tp my8711 on your computer or workstation. See Chapter 4 for instructions on how to do this.
2. Type cd data at the ftp prompt.
3. Type put ib,prog prog . bas at the prompt to put a copy of your program into the analyzer.
4. Close the connection and exit ftp by typing bye or quit at the prompt.
5. lb run your IBASIC program press [SYSTEM OPTIONS] IBASIC Run on the analyzer.

> NOTE
>
> You can eliminate this last step, and have your program run automatically by using the dynamic data disk file named prog,run. bas. See "To Copy and Run a Program" later in this section.

To copy the currently loaded IBASIC program from your analyzer to your computer:

1. Connect to the analyzer using ftp. For example, type my87 11 on your computer or workstation. See Chapter 4 for instructions on how to do this.
2. Type cd data at the ftp prompt.
3. Type dir at the ftp prompt to see the listing of files in this directory, as well as a short description of each of them.
4. Type get prog . bas at the prompt to retrieve the current IBASIC program file from the analyzer. This copies the program Ele prog.bas to your computer. You may want to give the file a unique name on your local computer by typing something like this: get prog . bas newf ile.
5. Close the connection and exit ftp by typing bye or quit at the prompt.

To Copy and Run a Program with One Command

You can create an IBASIC program or a file with a list of SCPI commands on your computer, and then copy and automatically run it by using the **prog_run.** bas and prog,run. scp files.

lb copy and immediately run an IBASIC program called **ib_prog**:

1. Connect to the analyzer using ftp. For example, type **my871 1** on your computer or workstation. See Chapter 4 for instructions on how to do this.
2. Type cd data at the ftp prompt.
3. Type put ib,prog **prog_run.bas** at the prompt. This copies your program to the analyzer and immediately runs it.
4. You can also copy a file with a list of SCPI commands to the prog,run. scp data file and the commands will be executed immediately. See your analyzer's *Programmer's* Guide for a list of SCPI commands.

   The file you copy to "prog_run.scp" should simply be a list of SCPI commands. For example, a file containing these lines:

   ```
   SENSl:FUNC 'XFR:POW:RAT2,0';DETNBAN;*WAI
   CALC1:MARK:FUNCMAX
   DISP:WINDl:TRAC:Y:AUTO  ONCE
   ```

   sets the analyzer to measure transmission, places a marker on the maximum point, and then autoscales the measurement trace.

NOTE

Avoid the use of queries, as there is no way to read back the analyzer's response.

Copying a Screen Image to a Local File

This section describes how to copy screen images from the analyzer to a file on your computer.

To copy a screen image to your computer:

1. Connect to the analyzer using ftp. For example, type **my8711** on your computer or workstation. See Chapter 4 for instructions on how to do this.
2. Type cd data at the ftp prompt.
3. Type dir at the ftp prompt to see the listing of files in this directory, as well as a short description of each of them.
4. Decide which screen image file you want to retrieve and then use the get command to transfer it to your computer. For example, type get screen **.pcx** image **.pcx** to retrieve the current screen image in PCX format, and place it into a file named image **.pcx** on your computer.

NOTE

The look of the image you retrieve will depend on the selections in the **Define Hardcopy** menu on your analyzer, as well as the file you choose to retrieve from the data directory For example, Figure 5-1 was retrieved with the **"screen.hgl"** file using the analyzer's default hardcopy mode, which includes the measurement graph and the marker table Figure 5-2 was retrieved with the analyzer's hardcopy mode defined as Graph **Only** using the **"screen_m.hgl"** file.

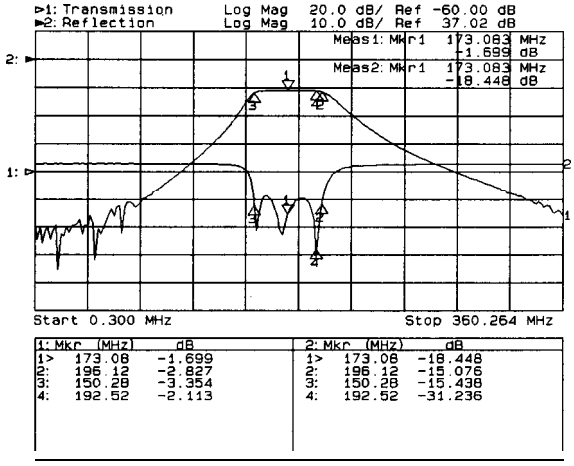


Figure 6-1. **screen.hgl**

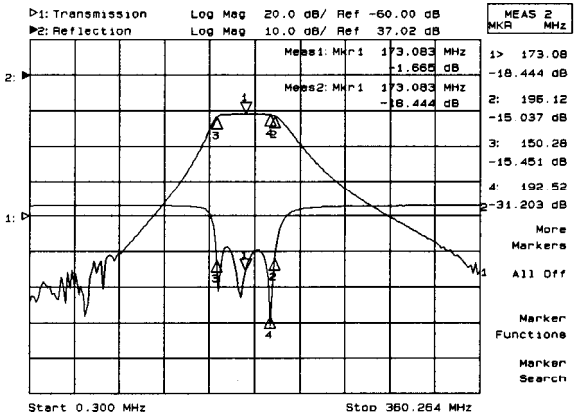


Figure **5-2. screen_m.hgl**

Copying Instrument Parameters in ASCII Text Format

This section describes how to use the parm,all . txt and parm,screen. txt files that reside in the data directory of the analyzer. See Table 5-1 for a brief description of each of these files.

Instrument parameters can be viewed on the analyzer by pressing (SYSTEM OPTIONS) Operating Parameters . Several screens of information are available (the exact number depends upon your model number and option configuration). These screens describe all the current settings and configurations of the analyzer. You can copy all of these screens using "parm_all.txt" or just the current screen with "parm_screen.txt" to an ASCII file on your computer.

To copy instrument operating parameters:

1. Connect to the analyzer using ftp. For example, type my87 11 on your computer or workstation. See Chapter 4 for instructions on how to do this.
2. Type cd data at the ftp prompt.
3. Type get parm,all . txt or get parm,screen. txt at the prompt to copy the desired parameters to your local computer. You can give the file a unique name on your local computer by typing: get parm_all . txt newf ile.
4. Close the connection and exit ftp by typing bye or quit at the prompt.

Copying the parm_screen.txt file to a DOS environment

If you do not rename the "parm_screen.txt" file when copying it to a DOS environment (as in step 3 above), it will be truncated to "parm_scr.txt" in order to comply with DOS file-naming conventions. There will be no feedback or indication from ftp that this has happened.

Retrieving Measurement Data in ASCII Format

This section describes how to use the **trace1.prn**, **trace2.prn**, **trace1**. slp and **trace2.** slp files that reside in the data directory of the analyzer. See Table 5-1 for a brief description of each of these files.

Measurement data can be saved in ASCII formats that are compatible with many personal computer software packages. The files with the ".prn" extension in the data directory contain measurement data in a two-column format that is directly importable to Lotus@ 1-2-3®, as well as other spreadsheet programs. The files with the ".s1p" extension in the data directory contain measurement data in a format that is importable into CAE programs such as HP EEsof's Microwave Design System (MDS) and Series IV.

lb retrieve measurement data:

1. Connect to the analyzer using ftp. For example, type **my87** 11 on your computer or workstation. See Chapter 4 for instructions on how to do this.
2. Type cd data at the ftp prompt.
3. Type get **trace1.prn** at the prompt to copy the measurement channel 1 data in spreadsheet format. See Table 5-1 for descriptions of the other trace data files. You can give the file a unique name on your local computer by typing: get **trace1.prn** newf ile.
4. Close the connection and exit ftp by typing bye or quit at the prompt.

Importing Graphics or Data into PC Applications

Some of the newer PC word processor and spreadsheet programs allow you to import graphics and data from the Web. The following examples illustrate how to import a screen image from your analyzer into Microsoft@ Word 97, and how to import trace data from your analyzer into Microsoft@ Excel 97.

Import a Screen Snapshot into a Word Processor Program

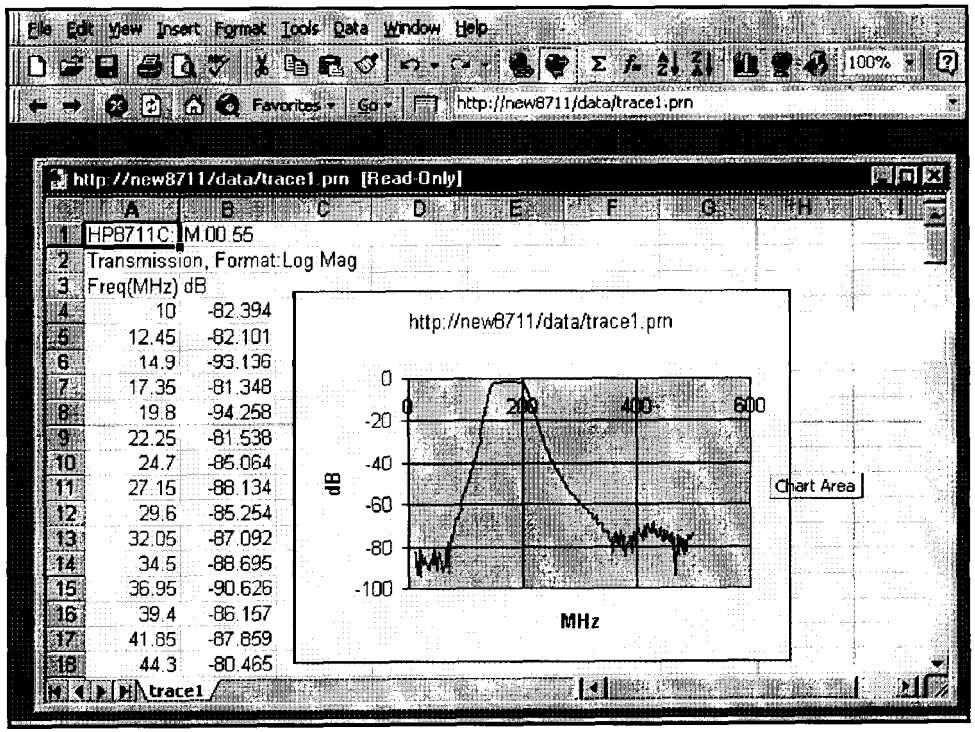
This example steps you through importing a picture of the analyzer's current screen into a word processor. (These steps were developed using Microsoft@ Word 97. Other word processing programs may or may not have the same capability, and will probably have different steps.)

1. Place the cursor at the point in your file where you want to place the imported graphic.
2. Click on Insert, Picture, From File. When the dialog box appears, type `http://my8711/data/screen.pcx` in the File name box.
3. Click on the Insert button in the dialog box.

Import Trace Data into a Spreadsheet Program

This example steps you through importing the analyzer's current trace data into a spreadsheet program. (These steps were developed using Microsoft@ Excel 97. Other spreadsheet programs may or may not have the same capability, and will probably have different steps.)

1. Click on File, Open. When the dialog box appears, type http : **//my8711/data/trace1.s1p** in the File name box.
2. Click on the Open button in the dialog box.
3. A "Text Import Wizard" will guide you through customizing how you want the data to appear in the spreadsheet. Figure 5-3 shows trace data (and a screen snapshot) imported into a spreadsheet program.



php66c

Figure 5-3. Trace Data and Screen Snapshot Imported into a Spreadsheet

6

Controlling the Analyzer via the LAN

Controlling the Analyzer via the LAN

To control your analyzer via LAN, you must use socket programming to send SCPI commands. Your analyzer is listening for SCPI commands at port 5025.

You can also control your analyzer by copying programs to the data directory via ftp (see Chapter 5, and see "IBASIC Communication Across the LAN" in this chapter).

NOTE

The example programs described in this chapter are on the "Example Programs Disk" that was shipped with your analyzer.

The Command Parser Port

You can telnet to the analyzer's command parser port and send SCPI programming interactively, or you can write a program that opens a socket and sends commands to it.

The analyzer listens for SCPI commands on port 23 and port 5025. Port 23 is intended for interactive use, using telnet. When you connect to port 23, a welcome message is displayed. Port 5025 is intended for program use. When you connect to port 5025, no welcome message is displayed.

Entering Commands Directly using Telnet

Before connecting to your analyzer using telnet, you must have connected and configured your analyzer as described in Chapter 1.

Using telnet to send commands to your analyzer works in a similar way to communicating over HP-IB; you establish a connection with the analyzer, and then send and/or receive information via SCPI commands.

NOTE

In the current version of firmwaren there is no way to mirror HP-IB operations such as "device clear" or SRQs. To do so requires XL/LAN, which has not yet been implemented.

As a workaround, you can use the HP E2050A LAN to HP-IB gateway, which supports SICL/LAN and can control your analyzer via HP-IB.

The syntax of the telnet command is:

telnet **hostname**

For example, type telnet **my8711**. A brief message should appear confirming the connection, and telling you the escape sequence for breaking the connection.

```
Trying. . .
Connected to my8711.sr.hp.com.
Escape character is '^]'.
Welcome to the HP 871xC SCPI Parser
```

When you connect to the analyzer, there is no prompt. You can now begin typing analyzer SCPI commands; query results appear on the next line. When you are done, break the telnet connection using the escape character (in this case Ctrl]), and type quit. See the detailed example that follows.

Example

lb connect to the analyzer named "my8711", enter:

telnet **my8711**

The computer responds with:

```
Trying. . .
Connected to my8711.sr.hp.com
Escape character is '^]' .
Welcome to the HP 871xC SCPI Parser
```

The connection was successful. Because the analyzer does not provide a prompt, start entering programming commands. Typical commands might be:

```
SENSl:FUNC 'XFR:POW:RAT 2,0':DET NBAN;*OPC?
CALC1:MARK:FUNCMAX
CALCl:MARK:POIN?
```

The small program above sets the analyzer to measure transmission, places a marker on the maximum point, and then queries the analyzer for the amplitude of the marker.

You need to press Enter after typing in each command. After pressing Enter on the last line in the example above, the analyzer returns the amplitude level of the marker to your computer and displays it on the next line. For example, after typing **CALC1** : MARK : POIN? and pressing Enter, the computer would display:

```
+1.71000000000E+002
```

When you are done, close the telnet connection. Enter the escape character to get the telnet prompt. The escape character (Ctrl and "]" in this example) does not print.

At the telnet prompt, type quit or close.

The telnet connection closes and you see your regular prompt.

```
Connection closed.
$
```

NOTE

If your telnet connection is in a mode called "line-by-line," there is no local echo. This means you will not be able to see the characters you are typing on your computer's display until after you press the Enter key

To remedy this, you need to change your telnet connection to 'character-by-character" mode. This can be accomplished in most systems by escaping out of telnet to the **telnet>** prompt, and then typing mode char. If this does not work, consult your telnet program's documentation for how to change to "character-by-character" mode.

Programming the Analyzer within a C Program

The following example program demonstrates simple socket programming. It is written in C, and compiles in the HP-UX UNIX environment. It should be portable to other UNIX environments with only minor changes.

In UNIX, LAN communication via sockets is very similar to reading or writing a file. The only difference is the openSocket () routine, which uses a few network library routines to create the TCP/IP network connection. Once this connection is created, the standard fread() and fwrite() routines are used for network communication.

In Windows, the routines send() and recv () must be used, since fread () and fwrite () may not work on sockets.

The program reads the analyzer's hostname from the command line, followed by the SCPI command. It then opens a socket to the analyzer using port 5025, and sends the command. If the command appears to be a query, the program queries the analyzer for a response, and prints the response.

This example program can also be used as a utility to talk to your analyzer from the command prompt on your UNIX workstation or Windows95 PC, or from within a script.

```
/*
 ************************************************************************
 *  $Header: lanio.c,v 1.6 97/02/13 13:03:10 roger Exp $
 *  $Revision: 1.6 $
 *  $Date: 97/02/13 13:03:10 $
 *
 *
 *  $Contributor:      LSID, MID $
 *
 *  $Description:      Functions to talk to an HP 8711C/12C/13C/14C/30A
 *                     analyzer via TCP/IP.  Uses command-line arguments.
 *
 *                     A TCP/IP connection to port 5025 is established and
 *                     the resultant file descriptor is used to "talk" to the
 *                     instrument using regular socket I/O mechanisms. $
 *
 *  $Written by:       Glenn Engel.  Modified for WIN32 by Roger Petersen $
 *
 *  871xC Examples:
 *
 *   query the center frequency:
 *        lanio 15.4.43.5 'sens:freq:cent?'
 *
 *   Select Lin Mag format:
 *        lanio my8711.sr.hp.com 'CALC:FORM MLIN'
 *
 *   Take a sweep, wait for end of sweep, move mkr to peak and query x pos:
 *        lanio my8711 ':abort;INIT:CONT OFF;:INIT1;*wai; :calc:mark:max; x?'
 *
 *   Query X and Y values of marker 1 and marker 2 (assumes they are on>:
 *        lanio my8711 'calc:mark1:x?;y?; :calc:mark2:x?;y?'
 *
 *   Check for errors (gets one error>:
 *        lanio my8711 'syst:err?'
 *
 *   Send a list of commands from a file, and number them:
 *        cat scpi,cmds I lanio -n my8711
 *
 ************************************************************************
 *
```

```
 * This program compiles and runs under
 *     - HP-UX 9.05 (UNIX), using HP cc or gcc:
 *           + cc -Aa    -0 -0 lanio lanio.c
 *           + gcc -Wall -0 -0 lanio lanio.c
 *
 *     - Windows 95, using Microsoft Visual C++ 4.0 Standard Edition
 *     - Windows NT 3.51, using Microsoft Visual C++ 4.0
 *           + Be sure to add WSOCK32.LIB to your list of libraries!
 *           + Compile both lanio.c and getopt.c
 *           + Consider re-naming the files to lanio.cpp and getopt.cpp
 *
 * Considerations:
 *     - On UNIX systems, file I/O can be used on network sockets.
 *       This makes programming very convenient, since routines like
 *       getc(), fgets(), fscanf() and fprintf() can be used. These
 *       routines typically use the lower level read() and write() calls.
 *
 *     - In the Windows environment, file operations such as read(), write(),
 *       and close() cannot be assumed to work correctly when applied to
 *       sockets.  Instead, the functions send() and recv() MUST be used.
 */

/* Support both Win32 and HP-UX UNIX environment */
#ifdef _WIN32     /* Visual C++ 4.0 will define this */
# define WINSOCK
#endif

tifndef WINSOCK
# ifndef _HPUX_SOURCE
# define _HPUX_SOURCE
#  endif
#endif

#include <stdio.h>          /* for fprintf and NULL */
#include <string.h>         /* for memcpy and memset */
#include <stdlib.h>         /* for malloc(), atol() */
#include <errno.h>          /* for strerror         */

#ifdef WINSOCK

#include <windows.h>
```

```
# ifndef _WINSOCKAPI_
# include <winsock.h>   // BSD-style socket functions
#  endif

#else /* UNIX with BSD sockets */

# include <sys/socket.h>    /* for connect and socket*/
# include <netinet/in.h>    /* for sockaddr,in       */
# include <netdb.h>         /* for gethostbyname     */

# define SOCKET-ERROR (-1)
# define INVALID-SOCKET (-1)

   typedef int SOCKET;

#endif /* WINSOCK */

#ifdef WINSOCK
  /* Declared in getopt.c.  See example programs disk. */
  extern char *optarg;
  extern int optind;
  extern int getopt(int argc, char * const argv[], const char* optstring);
#else
# include <unistd.h>          /* for getopt(3C) */
#endif

#define COMMAND-ERROR (1)
#define NO,CMD-ERROR (0)

#define SCPI_PORT 5025
#define INPUT,BUF,SIZE (64*1024)


/*************************************************************************
 * Display usage
 *************************************************************************/
static void usage(char *basename)
{
    fprintf(stderr,"Usage: %s [-nqu] <hostname> [<command>]\n", basename);
    fprintf(stderr,"       %s [-nqu] <hostname> < stdin\n", basename);
```

```
    fprintf(stderr,"  -n, number output lines\n");
    fprintf(stderr," -q, quiet; do NOT echo lines\n");
    fprintf(stderr,"  -e, show messages in error queue when done\n");
}


#ifdef WINSOCK
int init_winsock(void)
{
    WORDwVersionRequested;
    WSADATAwsaData;
    int err;
    wVersionRequested = MAKEWORD(1, 1);
    wVersionRequested = MAKEWORD(2, 0);

    err = WSAStartup(wVersionRequested, &wsaData);

    if (err != 0) {
        /* Tell the user that we couldn't find a useable */
        /* winsock.dll.     */
        fprintf(stderr, "Cannot initialize Winsock 1.1.\n");
        return -1;
    }
    return 0;
}

int close_winsock(void)
{
    WSACleanup();
    return 0;
}
#endif /* WINSOCK */


/**************************************************************************
 *
 > $Function: openSocket$
 *
 * $Description:  open a TCP/IP socket connection to the instrument $
 *
 *  $Parameters: $
```

```
 *    (const char *) hostname . . . . Network name of instrument.
 *                                     This can be in dotted decimal notation.
 *    (int) portNumber . . . . . . . . The TCP/IP port to talk to.
 *                                     Use 5025 for the SCPI port.
 *
 * $Return:     (int) . . . . . . . . . A file descriptor similar to open(1).$
 *
 * $Errors:     returns -1 if anything goes wrong $
 *
 ************************************************************************/
SOCKET openSocket(const char *hostname, int portNumber)
{
    struct hostent *hostPtr;
    struct sockaddr-in peeraddr-in;
    SOCKET s;


    memset(&peeraddr_in, 0, sizeof(struct sockaddr,in));

    /*********************************************/
    /* map the desired host name to internal form. */
    /*********************************************/
    hostPtr = gethostbyname(hostname);
    if (hostPtr == NULL)
    {
        fprintf(stderr,"unable to resolve hostname '%s'\n", hostname);
        return INVALID-SOCKET;
    }

    /******************/
    /* create a socket */
    /******************/
    s = socket(AF_INET, SOCK-STREAM, 0);
    i f  (s ==INVALID-SOCKET)
    {
        fprintf(stderr,"unable to create socket to '%s': %s\n",
                hostname, strerror(errno));
        return INVALID-SOCKET;
    }

    memcpy(&peeraddr_in.sin_addr.s_addr, hostPtr->h_addr, hostPtr->h_length);
```

```
    peeraddr_in.sin_family = AF_INET;
    peeraddr_in.sin_port = htons((unsigned short)portNumber);

    if (connect(s, (const struct sockaddr*)&peeraddr_in,
                sizeof(struct sockaddr,in)) == SOCKET-ERROR)
    {
        fprintf(stderr,"unable to create socket to '%s': %s\n",
                hostname, strerror(errno));
        return INVALID-SOCKET;
    }

    return s;
}


/*****************************************************************************
 *
 > $Function: commandInstrument$
 *
 * $Description:  send a SCPI command to the instrument.$
 *
 *  $Parameters: $
 *     (SOCKET) . . . . . file pointer associated with TCP/IP socket.
 *     (const char *command) . . SCPI command string.
 * $Return:  (char *) . . . . . a pointer to the result string.
 *
 * $Errors:   returns 0 if send fails $
 *
 *****************************************************************************/
int commandInstrument(SOCKET sock,
                      const char *command)
{
    int count;

    /* fprintf(stderr, "Sending \"%s\".\n", command); */
    if (strchr(command, '\n') == NULL) {
       fprintf(stderr,
               "Warning: missing newline on command %s.  Might hang.\n",
               command);
    }
```

```
    count = send(sock, command, strlen(command), 0);
    if (count == SOCKET-ERROR) {
        return COMMAND-ERROR;
    }

    return NO_CMD_ERROR;
}

/*****************************************************************************
 *
 > $Function: sendBlock$
 *
 * $Description:  send a SCPI command to the instrument.$
 *
 *  $Parameters: $
 *     (SOCKET) . . . . . file descriptor associated with TCP/IP socket.
 *     (int) . . . . . Block length
 *     (const char *command) . Pointer to block
 * $Return:  (char *) . . . . . a pointer to the result string.
 *
 * $Errors:   returns 0 if send fails $
 *
 *****************************************************************************/
int sendBlock(SOCKET sock, const char *block, int block-length-in-bytes)
{
    int count;

    count = send(sock, block, block-length-in-bytes, 0);
    if (count == SOCKET-ERROR) {
        fprintf(stderr, "ERROR: sendBlock(): send() failed.\n");
        return COMMAND-ERROR;
    }

    return NO-CMD,ERROR;
}


/*****************************************************************************
 * recv_line(): similar to fgets(), but uses recv()
 *****************************************************************************/
char * recv_line(SOCKET sock, char * result, int maxLength)
```

```
{
#ifdef WINSOCK
    int cur-length = 0;
    int count;
    char * ptr = result;
    int err = 1;

    while (cur-length < maxLength) {
        /* Get a byte into ptr */
        count = recv(sock, ptr, 1, 0);

        /* If no chars to read, stop. */
        if (count < 1) {
            break;
        }
        cur-length += count;

        /* If we hit a newline, stop. */
        if (*ptr == '\n') {
            ptr++;
            err = 0;
            break;
        }
        ptr++;

    }

    *ptr = '\0';

    if (err) {
        return NULL;
    } else {
        return result;
    }
#else
    /*********************************************************************
     * Simpler UNIX version, using file I/O. recv() version works too.
     * This demonstrates how to use file I/O on sockets, in UNIX.
     *********************************************************************/
    FILE * instFile;
    instFile = fdopen(sock, "r+");
```

```
    if (instFile == NULL)
    {
        fprintf(stderr, "Unable to create FILE * structure : %s\n",
                strerror(errno));
        exit(2);
    }
    return fgets(result, maxlength, instFile);
#endif
}


/*****************************************************************************
 *
 > $Function: queryResponse() $
 *
 * $Description:  read response from analyzer
 *                (assumes query command was already sent)
 *
 *  $Parameters: $
 *     (SOCKET) . . . . . file descriptor associated with TCP/IP socket.
 *     (char *result) . . . . . where to put the result.
 *     (size_t) maxLength . . . maximum size of result array in bytes.
 *
 * $Return:  (long) . . . . . The number of bytes in result buffer.
 *
 * $Errors:   returns 0 if anything goes wrong. $
 *
 *****************************************************************************/
long queryResponse(SOCKET sock, char *result, size_t maxLength)
{
    long ch;
    char tmp_buf[8];
    long resultBytes = 0;
    int count;


    /*********************************************************
     * Read response from analyzer
     *********************************************************/
    count = recv(sock, tmp-buf, 1, 0); /* read 1 char */
    ch = tmp-buf [0];
```

```
if ((count < 1) I I (ch == EOF) || (ch == '\n'))
{
    *result = '\0'; /* null terminate result for ascii */
    return 0;
}

/* use a do-while so we can break out */
do
{
    if (ch == '#')
    {
        /* binary data encountered - figure out what it is */
        long numDigits;
        long numBytes = 0;
        /* char length[10]; */

        count = recv(sock, tmp_buf, 1, 0); /* read 1 char */
        ch = tmp_buf[0];
        if ((count C 1) I | (ch == EOF)) break; /* End of file */

        if (ch < '0' II ch > '9') break; /* unexpected char */
        numDigits = ch - '0';

        i f (numDigits)
        {
            /* read numDigits bytes into result string. */
            count = recv(sock, result, (int)numDigits, 0);
            result[count] = 0; /* null terminate */
            numBytes = atol(result);
        }

        i f (numBytes)
        {
            resultBytes = 0;
            /* Loop until we get all the bytes we requested. */
            /* Each call seems to return up to 1457 bytes (HP-UX 9.05) */
            do {
                int rcount;
                rcount = recv(sock, result, (int)numBytes, 0);
                resultBytes += rcount;
```

```
                result      += rcount;  /* Advance pointer */
            } while ( resultBytes < numBytes );

            /*************************************************************
             * For LAN dumps, there is always an extra trailing newline
             * Since there is no EOI line.  For ASCII dumps this is
             * great but for binary dumps, it is not needed.
             *************************************************************/
            if (resultBytes == numBytes)
            {
                char junk;
                count = recv(sock, &junk, 1, 0);
            }
        }
        else
        {
            /* indefinite block . . . dump til we can an extra line feed */
            do
            {
                if (recv_line(sock, result, maxLength) == NULL) break;
                if (strlen(result)==1 && *result == '\n') break;
                resultBytes += strlen(result);
                result += strlen(result);
            } while (I);
        }
    }
    else
    {
        /* ASCII response (not a binary block) */
        *result = (char)ch;
        if (recv_line(sock, result+1, maxLength-1) == NULL) return 0;

        /* REMOVE trailing newline, if present. And terminate string. */
        resultBytes = strlen(result);
        if (result[resultBytes-1] == '\n') resultBytes -= 1;
        result[resultBytes] = '\0';
    }
} while (0);

return resultBytes;
}
```

```
/*************************************************************************
 *
 > $Function: queryInstrument() $
 *
 * $Description:  send a SCPI command to the instrument, return a response.$
 *
 *  $Parameters: $
 *      (SOCKET) . . . . . file descriptor associated with TCP/IP socket.
 *      (const char *command) . . SCPI command string.
 *      (char *result) . . . . . where to put the result.
 *      (size-t) maxLength . . . . maximum size of result array in bytes.
 *
 * $Return:  (long) . . . . . The number of bytes in result buffer.
 *
 * $Errors:   returns 0 if anything goes wrong. $
 *
 *************************************************************************/
long queryInstrument(SOCKET sock,
                     const char *command, char *result, size_t maxLength)
{
    int command-err;

    /*********************************************************
     * Send command to analyzer
     *********************************************************/
    command-err = commandInstrument(sock,  command);
    if (command-err) return COMMAND-ERROR;

    return queryResponse(sock, result, maxLength);
}



/*************************************************************************
 *
 > $Function: showErrors$
 *
 * $Description: query the SCPI error queue, until empty.  Print results. $
```

```
 *
 * $Return:  (void)
 *
 ***************************************************************************/
void showErrors(SOCKET sock)
{
    const char * command = "SYST:ERR?\n";
    char result_str[256];

    do {
        queryInstrument(sock, command, result,str, sizeof(result_str)-1);

        /*****************************************************************
         * Typical result-str:
         *     -221,"Settings conflict; Frequency span reduced."
         *     +0 "No error"
         * Don't bother decoding.
         *****************************************************************/
        if (strncmp(result_str, "+0,", 3) == 0) {
            /* Matched +0,"No error" */
            break;
        }
        puts(result_str);
    } while (1);

}

/***************************************************************************
 *
 > $Function: isQuery$
 *
 * $Description: Test current SCPI command to see if it a query. $
 *
 * $Return:  (unsigned char) . . . non-zero if command is a query.  0 if not.
 *
 ***************************************************************************/
unsigned char isQuery( char* cmd )
{
    unsigned char q = 0 ;
    char *query ;
```

```
    /*********************************************************/
    /* if the command has a '?' in it, use queryInstrument. */
    /* otherwise, simply send the command.                   */
    /* Actually, we must a little more specific so that      */
    /* marker value queries are treated as commands.         */
    /* Example:  SENS:FREQ:CENT (CALC1:MARK1:X?)             */
    /*********************************************************/
    if ( (query = strchr(cmd,'?')) != NULL)
    {
        /* Make sure we don't have a marker value query, or
         * any command with a '?' followed by a ')' character.
         * This kind of command is not a query from our point of view.
         * The analyzer does the query internally, and uses the result.
         */
        query++ ;        /* bump past '?' */
        while (*query)
        {
            if (*query == ' ') /* attempt to ignore white spc */
                query++ ;
            else break ;
        }

        if ( *query != ')' )
        {
            q = l ;
        }
    }
    return q ;
}


/*****************************************************************************
 *
 > $Function: main$
 *
 * $Description: Read command line arguments, and talk to analyzer.
                 Send query results to stdout. $
 *
 * $Return:  (int) . . . non-zero if an error occurs
 *
 *****************************************************************************/
```

```
int main(int argc, char *argv[])
{

    SOCKETinstSock;
    char *charBuf = (char *) malloc(INPUT_BUF_SIZE);
    char *basename;
    int chr;
    char command[1024];
    char *destination;
    unsigned char quiet = 0;
    unsigned char show-errs = 0;
    int number = 0;

    basename = strrchr(argv[0], '/');
    if (basename != NULL)
        basename++ ;
    else
        basename = argv [0];

    while ( ( chr = getopt(argc,argv,"qune")) != EOF )
        switch (chr)
        {
            case 'q': quiet = 1; break;
            case 'n': number = 1; break ;
            case 'e': show-errs = I; break ;
            case 'u':
            case '?': usage(basename); exit(l) ;
        }

    /* now look for hostname and optional <command> */
    if (optind < argc)
    {
        destination = argv[optind++] ;
        strcpy(command, "");
        if (optind < argc)
        {
            while (optind < argc) {
                /* <hostname> <command> provided; only one command string */
                strcat(command, argv[optind++]);
                if (optind < argc) {
                    strcat(command, " ");
```

```
            } else {
                strcat(command, "\n");
            }
        }
    }
    else
    {
        /* Only <hostname> provided; input on <stdin> */
        strcpy(command, "");

        if (optind > argc)
        {
            usage(basename);
            exit(1);
        }
    }
}
else
{
    /* no hostname! */
    usage(basename);
    exit(1);
}

/*********************************************/
/* open a socket connection to the instrument */
/*********************************************/
#ifdef WINSOCK
    if (init_winsock() != 0) {
        exit(l);
    }
#endif /* WINSOCK */

    instSock = openSocket(destination,  SCPI,PORT);
    if (instSock == INVALID-SOCKET) {
        fprintf(stderr, "Unable to open socket.\n");
        return I;
    }
    /* fprintf(stderr, "Socket opened.\n"); */

    if (strlen(command) > 0)
```

```
{
    /*********************************************************/
    /* if the command has a '?' in it, use queryInstrument. */
    /* otherwise, simply send the command.                   */
    /*********************************************************/
    if ( isQuery(command) )
    {
        long bufBytes;
        bufBytes = queryInstrument(instSock, command,
                                   charBuf, INPUT,BUF,SIZE);
        if (!quiet)
        {
            fwrite(charBuf, bufBytes, 1, stdout);
            fwrite("\n", 1, 1, stdout);
            fflush(stdout);
        }
    }
    else
    {
        commandInstrument(instSock, command);
    }
}
else
{
    /* read a line from <stdin> */
    while ( gets(charBuf) != NULL )
    {
        if ( !strlen(charBuf) )
            continue;

        if ( *charBuf == '#' || *charBuf == '!' )
            continue;

        strcat(charBuf, "\n");

        if (!quiet)
        {
            if (number)
            {
                char num[10];
                sprintf(num,"%d: ",number);
```

```
                fwrite(num, strlen(num), 1, stdout);
            }
            fwrite(charBuf, strlen(charBuf), 1, stdout) ;
            fflush(stdout);
        }

        if ( isQuery(charBuf) )
        {
            long bufBytes;

            /* Put the query response into the same buffer as the
             * command string appended after the null terminator.
             */
            bufBytes = queryInstrument(instSock, charBuf,
                                       charBuf + strlen(charBuf) + 1,
                                       INPUT-BUF,SIZE -strlen(charBuf));
            if (!quiet)
            {
                fwrite("  ", 2, 1, stdout) ;
                fwrite(charBuf + strlen(charBuf)+1, bufBytes, 1, stdout);
                fwrite("\n", I, 1, stdout) ;
                fflush(stdout);
            }
        }
        else
        {
            commandInstrument(instSock, charBuf);
        }
        if (number) number++;
    }
}

if (show-errs) {
    showErrors(instSock);
}

tifdef WINSOCK
    closesocket(instSock);
    close_winsock();
#else
    close(instSock);
```

```
#endif /* WINSOCK */

    return 0;
}

/* End of lanio.c */
```

IBASIC Communication Across the LAN

You may need a way for an IBASIC program running on the analyzer to signal a remote computer that it has completed some operation.

IBASIC currently cannot communicate directly across LAN using the ASSIGN and OUTPUT or ENTER commands. However, IBASIC can use the following SCPI command to send a message to a remote computer via LAN:

DIAGnostic:COMMunicate:LAN:SEND <IP_ADDR>,<PORT_NUM>,<STRING>, <TIMEOUT>

This command opens a socket to the remote computer, and sends the specified string. The **<IP_ADDR>** argument specifies the IP address of the remote computer. The **<PORT_NUM>** argument specifies the port number to use. The <STRING> is the message to be sent. The **<TIMEOUT>** argument specifies the timeout interval (0 to 75 seconds).

For example:

```
DIAGnostic:COMMunicate:LAN:SEND '15.4.40.49',8001,'Ready!',30
```

If the remote computer is not listening for a LAN connection at the specified port, this command will block, and wait for the remote computer to accept the connection. In this example, it will time out after 30 seconds. If no timeout interval is specified, it will time out after the default timeout of about 75 seconds (the standard TCP/IP timeout period).

The following IBASIC example program demonstrates LAN communication using IBASIC.

```
100  !
110  ! This program demonstrates how IBASIC can communicate
120  !  with a remote computer via LAN. This is done using a
130  !  SCPI command that sends a LAN message to the computer:
140  !     DIAG:COMM:LAN:SEND'15.4.40.49',8003,'Ready!'
150  ! 8003 is an arbitrary unused port number.
160  !
170  DIM Cmd$[256]
180  DIM Msg$[128]
190  DIM Snum$[16]
```

```
200  !
210  ! Initialize the instrument
220  !
230  ASSIGN @Na TO 800
240  OUTPUT @Na;"SYST:PRES;*WAI"
250  OUTPUT @Na;"SENS1:STAT ON;*WAI"
260  OUTPUT @Na;"POW1:MODEFIXed"    ! Freq sweep
270  OUTPUT @Na;"DISP:ANN:FREQ1:MODECSPAN"
280  OUTPUT @Na;"SENS1:FREQ:CENT 177e6;SPAN 200e6;*WAI"
290  ! Put sweep in hold
300  OUTPUT @Na;"ABOR;:INIT1:CONTOFF;*WAI"
310  ! Sync up with analyzer
320  OUTPUT @Na;"*OPC?"
330  ENTER @Na;Opc
340  !
350  !
360  ! Get serial number
370  OUTPUT @Na;"DIAG:SNUM?"
380  ENTER @Na;Snum$
390  Snum$=Snum$[2,11]     ! remove quotes
400  !
410  ! Begin infinite loop:
420  !   - Take sweep
430  !   - Compute bandwidth
440  !   - Send signal to computer
450  !
460 Loop: !
470  DISP "Taking sweep..."
480  Count=Count+1
490  ! Take sweep, and wait for it to finish.
500  OUTPUT @Na;"INIT1;*OPC?"
510  ENTER @Na;Opc
520  ! Autoscale trace to give feedback.
530  OUTPUT @Na;"DISP:WIND1:TRAC:Y:AUTO ONCE"
540  ! Perform a search for the -3 dB bandwidth of the filter
550  ! This function uses several markers to find 4 key values.
560  OUTPUT @Na;"CALC1:MARK:BWID-3;FUNC:RES?"
570  ! Read the four values:  the bandwidth, center
580  ! frequency, Q and the insertion loss.
590  ENTER @Na;Bwid,Center_f,Q,Loss
600  !
```

```
610  ! Signal computer that we are done,
620  ! so that it can come grab the meas results
630  !
640  ! Create a string that looks like this:
650  !    Ready!,"US36100007",6.159E+7,1.7248E+8,-1.6088<LF>
660  ! Could send any string.  Could also save meas results to
670  ! a file, and send filename, and computer could FTP the file.
680  !
690  Msg$="'Ready!,"&Snum$&","
700  Msg$=Msg$&val$(Bwid)&","&val$(Center_f)&","&val$(Loss)
710  Msg$=Msg$&chr$(10)&"'"
720  !
730  ! Send the message to the computer, via LAN
740  !
750  Cmd$="DIAG:COMM:LAN:SEND '15.4.40.49',8003,"&Msg$
760  OUTPUT@Na;Cmd$
770  DISP "Done with loop ";Count;" Continuing..."
780  !
790  ! Pause, and wait for computer to grab data.
800  ! Computer will send 'PROG:STAT CONT' when ready
810  !
820  PAUSE
830  GOTO Loop
840  END
```

Controlling Multiple Analyzers using a Perl Script

The following Perl script demonstrates how you can control a network of analyzers from your workstation.

The script downloads an IBASIC program to a group of analyzers. The BASIC program makes a measurement, and then signals the computer that it needs service. (See the previous section, "IBASIC Communication Across the LAN" to see how the IBASIC program accomplishes this.)

The computer receives this signal, then queries the analyzer for measurement data, and then tells the IBASIC program to continue.

```
# ! /usr/bin/perl
#
# Perl script to listen on a port, and print received messages.
#
# This script is based on the "server" example in the
# book "Programming perl" by O'Reilley & Associates, Inc.
#

# require 'sys/socket.ph';        # Not needed on HP-UX
require 'ctime.pl';               # Allow use of ctime() to get date


# Get the port number from the command line (first arg).
# If no argument, default to a high-numbered port.
# Users can use ports above 1024 or so.
($port) = @ARGV;
$port = 8003 unless $port;


$AF_INET = 2;             # from /usr/include/sys/socket.h
$SOCK_STREAM = I;         # from /usr/include/sys/socket.h
$PF_INET = $AF_INET;      # from /usr/include/sys/socket.h
```

```
# Is this line noise?  No, it's the pack format:
#   s= unsigned short, n = short in network order
#  a4 = 4 ascii characters, null padded,
#  x8 = 8 null bytes (?)
$sockaddr = 'S n a4 x8';

chop($this_hostname = `hostname`);

($name, $alias, $proto) = getprotobyname('tcp');

#
# Create arguments for bind() and connect() calls, below.
#   0000 = Wildcard address
#
$thisport = pack($sockaddr, $AF_INET, $port, "\0\0\0\0");


select(NEW_SOCK); $| = I  ;   select(stdout);

#
# Open a network connection via a socket
#
socket(SOCK, $PF_INET, $SOCK_STREAM, $proto) ||
    die "cannot create socket: $!\n";
bind(SOCK, $thisport)                        ||
    die "cannot bind socket: $!\n";
listen(SOCK, SOMAXCONN)                      ||
    die "cannot listen socket: $!\n";


printf "Listening on port %d.\n", $port;
#
#
for ($con = I;; $con++) {
   #
   # Wait for incoming connections
   #
   $client_addr = accept(NEW_SOCK,SOCK) ||
       die "cannot accept socket: $!\n";

   # We have a connection!
```

```
# printf("Accepted connection #$con!\n");


#
# Here we used to call fork() to fork a child process.
# However, this causes problems if the parent doesn't wait()
# for the child -- zombie child processes are left behind!
# To fix this, it might be better to exec() the child process
# code so that it doesn't wait for the parent.
# This way, we can handle multipe overlapping messages.
# Even if we don't fork(), listen() will allow multiple
# pending connections.
#
# if (($child = fork()) == 0) {

#
# Get info about incoming connection, and print it
#
($af, $port, $ipaddr) = unpack($sockaddr, $client_addr);
@ipaddr = unpack('C4', $ipaddr);
$IP_addr = sprintf("%d.%d.%d.%d",
                   @ipaddr[0], @ipaddr[1], @ipaddr[2], @ipaddr[3]);
$Date = &ctime(time());  # "Mon Oct 21 21:52:22 PDT 1996\n"
printf "%d: Got message from %s at %s", $con, $IP_addr, $Date;

#
# Read incoming message, and save it to a file
# Append it to a file named  data.IP_addr giving
# each analyzer its own data file.
#
$file = "./data.$IP_addr";
# print "Routing input to $file.\n";
open(FILE_OUT, ">> $file") || die "Cannot open $file: $!";
print FILE-OUT $Date;
while (<NEW_SOCK>) {
    print FILE-OUT "$_";
}
close(NEW_SOCK);
close(FILE_OUT);

# Tell the instrument's IBASIC program to continue
```

```
    system("lanio $IP_addr 'PROGram:STATe CONT'");
}

exit 0;
```

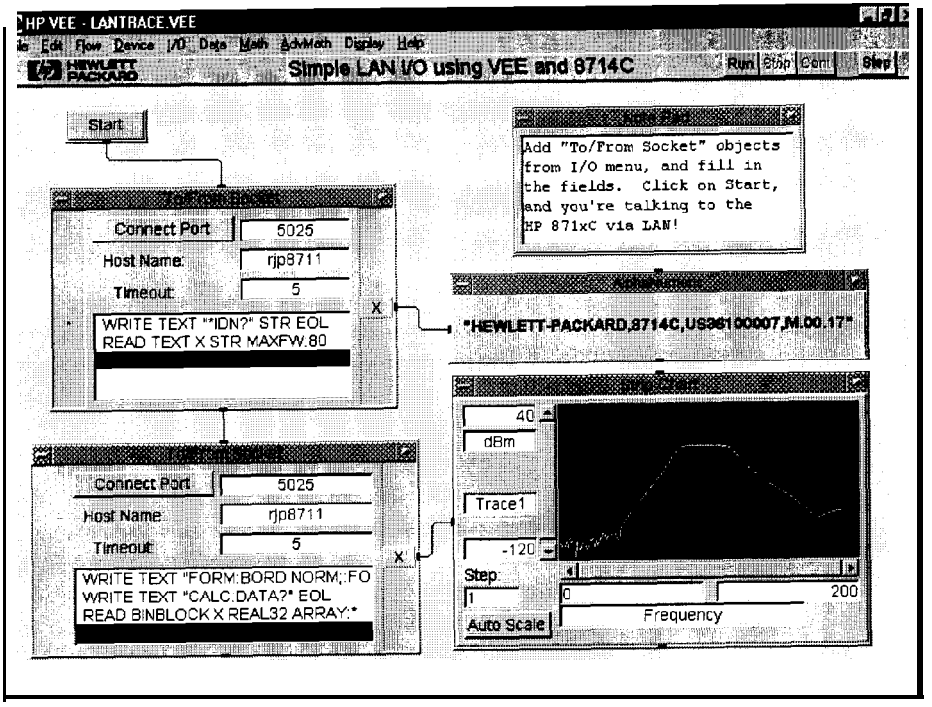
lb control your analyzer via LAN using HP VEE, click on the VEE menu titled "I/O." Then select "lb/From Socket" and position the I/O object box on the screen. Fill in the following Eelds:

Connect Port:	5025
Host Name:	my8711
Timeout :	15

For faster troubleshooting, you may want to set the timeout to a smaller number. If the hostname you enter doesn't work, try using the IP address of your analyzer (example: 15.4.43.5). Using the IP address rather than the hostname may also be faster. See Figure 6-1 for an example of an HP VEE screen.

NOTE

Since the current firmware revision does not support SICL/LAN, there is no way to perform SRQs or device clears via LAN. Future revisions may support this protocol.



php64c

Figure **6-1.** Sample HP VEE Screen

Controlling the Analyzer with a Java™ Applet

The following example program demonstrates simple socket programming with Java. It is written in Java programming language, and will compile with Java compilers versions 1.0 and above.

This program is on the example programs disk that was shipped with your analyzer. View the README file on the example programs disk before using this program.

```
import java.awt.*;
import java.io.*;
import java.net.*;
import java.applet.*;

// This is a SCPI Demo to demonstrate how one can communicate with the
// HP87xx network analyzer with a JAVA capable browser. This is the
// Main class for the SCPI Demo. This applet will need Socks.class to
// support the I/O commands and a ScpiDemo.html for a browser to load
// the applet.
// To use this applet, either compile this applet with a Java compiler
// or use the existing compiled classes. copy ScpiDemo.class,
// Socks.class and ScpiDemo.html to a floppy. Insert the floppy into
// your instrument. Load up a browser on your computer and do the
// following:
//      1. Load this URL in your browser:
//         ftp://<Your instrument's IP address or name>/int/ScpiDemo.html
//      2. There should be two text windows show up in the browser:
//         The top one is the SCPI response text area for any response
//         coming back from the instrument. The bottom one is for you
//         to enter a SCPI command. Type in a SCPI command and hit enter.
//         If the command expects a response, it will show up in the top
//         window.
public class ScpiDemo extends java.applet.Applet implements Runnable {
    Thread      responseThread;
    Socks       sck;
    URL         appletBase;
    TextField   scpiCommand = new TextField();
    TextArea    scpiResponse = new TextArea(10, 60);
```

```
Panel        southPanel = new Panel();
Panel        p;

// Initialize the applets
public void init() {

    SetupSockets();
    SetupPanels();

    // Set up font type for both panels
    Font font = new Font("TimesRoman", Font.BOLD,14);
    scpiResponse.setFont(font);
    scpiCommand.setFont(font);
    scpiResponse.appendText("SCPI Demo Program:  Response messages\n");
    scpiResponse.appendText("————————————————————----------\n");
}

// This routine is called whenever the applet is actived
public void start() {
    // Open the sockets if not already opened
    sck.OpenSockets();
    // Start a response thread
    StartResponseThread(true);
}

// This routine is called whenever the applet is out of scope
// i.e. minize browser
public void stop() {
    // Close all local sockets
    sck.CloseSockets();
    // Kill the response thread
    StartResponseThread(false);
}

// Action for sending out scpi commands
// This routine is called whenever a command is received from the
// SCPI command panel.
public boolean action(Event evt, Object what) {
    // If this is the correct target
    if (evt.target== scpiCommand) {
        // Get the scpi command
```

```
            String str = scpiCommand.getText();
            // Send it out to the Scpi socket
            sck.ScpiWriteLine(str);
            // Query for any error
            sck.ScpiWriteLine("syst:err?");
            return true;
        }
        return false;
    }

    // Start/Stop a Response thread to display the response strings
    private void StartResponseThread(boolean start) {
        if (start) {
            // Start a response thread
            responseThread = new Thread(this);
            responseThread.start();
        }
        else {
            // Kill the response thread
            responseThread = null;
        }
    }

    // Response thread running
    public void run() {
        String str = ""; // Initialize str to null

        // Clear the error queue before starting the thread
        // in case if there's any error messages from the previous actions
        while ( str.indexOf("No error") == -1 ) {
            sck.ScpiWriteLine("syst:err?");
            str = sck.ScpiReadLine();
        }

        // Start receiving response or error messages
        while(true) {
            str = sck.ScpiReadLine();
            // If response messages is "No error", do no display it
            if ( str.indexOf("No error") == -1 ) {
                // Display the error message in the Response panel
                scpiResponse.appendText(str+"\n");
```

```
                // query for any error messages
                sck.ScpiWriteLine("syst:err?");
            }
        }
    }

    // Set up and open the SCPI sockets
    private void SetupSockets() {
        // Get server url
        appletBase = (URL)getCodeBase();
        // Open the sockets
        sck = new Socks(appletBase);
    }

    // Set up the SCPI command and response panels
    private void SetupPanels() {
        // Set up SCPI command panel
        southPanel.setLayout(new GridLayout(1, 1));
        p = new Panel();
        p.setLayout(new BorderLayout());
        p.add("West", new Label("SCPI command:"));
        p.add("Center", scpiCommand);
        southPanel.add(p);

        // Set up the Response panel
        setLayout(new BorderLayout(2,2));
        add("Center", scpiResponse);
        add("South", southPanel);
    }

}


// Socks class is responsible for open/close/read/write operations
// from the predefined socket ports.  For this example program,
// the only port used is 5025 for the SCPI port.
class Socks extends java.applet.Applet {
    // Socket Info
    // To add a new socket, add a constant here,
    // change MAX_NUM_OF_SOCKETS
    // then, edit the constructor for the new socket.
```

```
public final int SCPI=0;
private final int MAX_NUM_OF_SOCKETS=1;

// Port number
// 5025 is the dedicated port number for HP8711's SCPI port
private final int SCPI,PORT = 5025;

// Socket info
private URL appletBase;
private Socket[] sock = new Socket[MAX_NUM_OF_SOCKETS];
private DataInputStream[] sockIn=new DataInputStream[MAX_NUM_OF_SOCKETS];
private PrintStream[] sockOut = new PrintStream[MAX_NUM_OF_SOCKETS];
private int[] port = new int[MAX_NUM_OF_SOCKETS];
private boolean[] sockOpen = new boolean[MAX_NUM_OF_SOCKETS];

// Constructor
Socks(URL appletB)
{
    appletBase = appletB;

    // Set up for port array.
    port[SCPI] = SCPI,PORT;

    // Initialize the sock array
    for ( int i = 0; i < MAX_NUM_OF_SOCKETS; i++ ) {
        sock[i] = null;
        sockIn[i] = null;
        sockOut[i] = null;
        sockOpen[i] = false;
    }
}

//***** Sockects open/close routines
// Open the socket(s) if not already opened
public void OpenSockets()
{
    try {
        // Open each socket if possible
        for ( int i = 0; i < MAX,NUM-OF-SOCKETS; i++ ){
            if ( !sockOpen[i] ) {
                sock[i] = new Socket(appletBase.getHost(),port[i]);
```

```
                sockIn[i]=new DataInputStream(sock[i].getInputStream());
                sockOut[i]=new PrintStream(sock[i].getOutputStream());
                if ( (sock[i] != null) && (sockIn[i] != null) &&
                     (sockOut[i] != null) ) {
                    sockOpen[i] = true;
                }
            }
        }
    }
    catch (IOException e) {
        System.out.println("Sock, Open Error "+e.getMessage());
    }
}

// Close the socket(s) if opened
public void CloseSocket(int s)
{
    try {
        if ( sockOpen[s] == true ) {
            // write blank line to exit servers elegantly
            sockOut[s].println();
            sockOut[s].flush();
            sockIn[s].close();
            sockOut[s].close();
            sock[s].close();
            sockOpen[s] = false;
        }
    }
    catch (IOException e) {
        System.out.println("Sock, Close Error "+e.getMessage());
    }
}

// Close all sockets
public void CloseSockets()
{
    for ( int i=0; i < MAX_NUM_OF_SOCKETS; i++ ) {
        CloseSocket(i);
    }
}
```

```
// Return the status of the socket, open or close.
public boolean SockOpen(int s)
{
    return sockOpen[s];
}


//************* Socket I/O routines.

//*** I/O routines for SCPI socket

// Write an ASCII string with carriage return to SCPI socket
public void ScpiWriteLine(String command)
{
    if ( SockOpen(SCPI) ) {
        sockOut[SCPI].println(command);
        sockOut[SCPI].flush();
    }
}

// Read an ASCII string, terminated with carriage return
// from SCPI socket
public String ScpiReadLine()
{
    try {
        if ( SockOpen(SCPI) ) {
            return sockIn[SCPI].readLine();
        }
    }
    catch (IOException e) {
        System.out.println("Scpi Read Line Error "+e.getMessage());
    }
    return null;
}

// Read a byte from SCPI socket
public byte ScpiReadByte()
{
    try {
        if ( SockOpen(SCPI) ) {
            return sockIn[SCPI].readByte();
```

```
            }
        }
        catch (IOException e) {
            System.out.println("Scpi Read Byte Error "+e.getMessage());
        }
        return 0;
    }

}
```

Controlling the Analyzer using HP BASIC

Controlling your analyzer via LAN using HP BASIC requires the analyzer to support a protocol called SICL/LAN. The current version of firmware does not support this protocol. This protocol may be implemented in future flrmware revisions.

As a workaround, you can use the HP E2050A LAN to HP-IB gateway, which supports SICL/LAN and can control your analyzer via HP-IB.

7

General Troubleshooting

General Troubleshooting

This chapter provides troubleshooting information for the LAN interface. It is arranged in four sections:

- Troubleshooting the initial connection
- Subnets and Gateways
- Solutions to common problems
- Getting service support

Troubleshooting the Initial Connection

Getting the analyzer to work with your network often requires detailed knowledge of your local network software. This section attempts to help you with some common problems, but because of the wide variety of network software available, it cannot cover all problems you may encounter.

Assess the Problem

The LAN interface does not need or include any proprietary driver software. The analyzer LAN interface was designed to operate with common network utilities and drivers.

Either a hardware problem or a software problem can prevent the analyzer's remote file server from communicating over the LAN.

Timeout Errors

Error messages such as "Device Timeout, " "File Timeout," "Operation Timeout," or other similar messages from workstations or PCs indicate timeout problems with the computer. lb increase your timeout period, refer to your computer documentation for instructions.

Packets Routinely lost

If packets are routinely lost, proceed to the troubleshooting section in this chapter relating to your network.

Problems Transferring or Copying Files

If you have problems copying files out of or into the analyzer, you might be experiencing timeout problems. See the previous section on "Timeout Errors. "

Communications not Established

If you have just installed and configured the LAN interface and you have never been able to access the analyzer via ftp or telnet, go directly to "Ping the Analyzer from your Computer or Workstation" in this chapter.

If you have previously been able to access the analyzer via ftp or telnet and now cannot do so, check the following:

- Has any hardware been added or moved on your network? This includes adding or removing any workstations or peripherals, or changing any cabling.
- Have software applications been added to the network?
- Have any configuration files been modified?
- Have any of the following files been deleted or overwritten?

  UNIX:

  /etc/hosts

  /etc/inetd.conf

  /etc/services

  PCs:

  dependent network Eles

If you know or suspect that something has changed on your network, consult with your network administrator.

Ping the Analyzer from your Computer or Workstation

Verify the communications link between the computer and the analyzer remote file server using the ping utility.

From a UNIX workstation, type:

```
ping hostname 64 10
```

where 64 is the packet size, and 10 is the number of packets transmitted.

From a DOS or Windows environment, type:

```
ping hostname 10
```

where 10 is the number of echo requests.

Normal Response for UNIX

A normal response to the ping will be a total of 9, 10, or possibly 11 packets received with a minimal average of round-trip time. The minimal average will be different from network to network. LAN traffic will cause the round-trip time to vary widely.

Because the number of packets received depends on your network traffic and integrity, the normal number might be different for your network.

Normal Response for DOS or Windows

A normal response to the ping will be a total of 9, 10, or possibly 11 packets received if 10 echo requests were specified.

Because the number of packets received depends on your network traffic and integrity, the normal number might be different for your network.

Error Messages

- If error messages appear, check the command syntax before continuing with the troubleshooting. If the syntax is correct, resolve the error messages using your network documentation, or by consulting your network administrator.
- If an unknown host error message appears, check the node names database to see that the hostname and IP address for your analyzer are correctly entered.

No Response

No packets received indicates no response from a ping.

- If there is no response, try typing in the IP address with the ping command, instead of using the hostname. Check that the typed address matches the IP address assigned in the **LAN Port** Setup menu, then check the other addresses in the menu.
- Check that the hostname and IP address are correctly entered in the node names database.
- If you are using a UNIX environment, ping each node along the route between your workstation and the analyzer, starting with the your workstation. Ping each gateway, then attempt a ping of the remote file server.
- If the analyzer still does not respond to ping, you should suspect a hardware problem with the analyzer. To check the analyzer performance, refer to "Verify the Analyzer Performance" in this chapter.

Intermittent Response

If you received 1 to 8 packets back, there is probably a problem with the network. Because the number of packets received depends on your network traffic and integrity, the number might be different for your network.

Use a LAN analyzer or LAN management software to monitor activity and determine where bottlenecks or other problems are occurring. The analyzer will still function, but communications over the LAN will be slower.

On a single-client/single-server network, the most likely cause of intermittent response to an echo request is a hardware problem with the LAN module installed in the PC, the cable, or the analyzer. To check the analyzer, refer to "Verify the Analyzer Performance" later in this chapter.

Ping your Computer or Other Device from your Analyzer

The last section helped you verify connectivity from your computer to your analyzer. This section helps you verify the connectivity path in the opposite direction-from your analyzer to your computer.

To Use the Built-In Ping Utility

'lb check for connectivity to your computer or any other device on your network from your analyzer:

1. Press (SYSTEM OPTIONS) **LAN** **LAN Port Setup** **Diagnostic Utilities**.
2. Press **IP Address** to Ping and enter the IP address of the computer or device you are trying verify connectivity to.
3. Press **Perform Ping** .

Normal Response

A normal response after pressing **Perform Ping** is shown below. The analyzer successfully attempts four cycles of communications with the indicated network device, and displays the response time for each cycle.

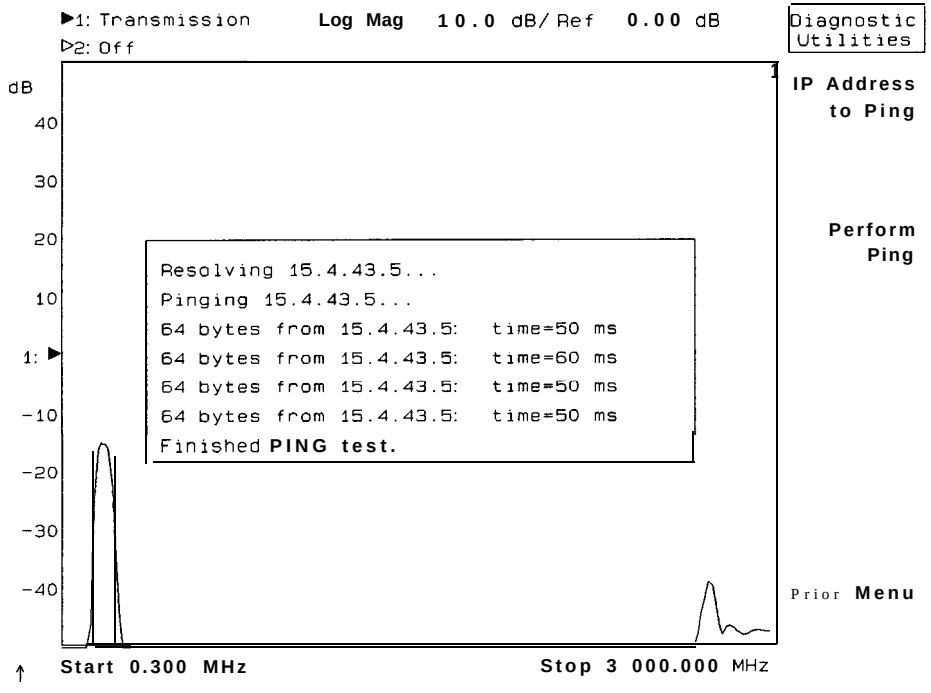


Figure 7-1. Example of a Successful Ping

Timeout Response

If communication is not established with the selected device within one second for each cycle, the display will look like this:

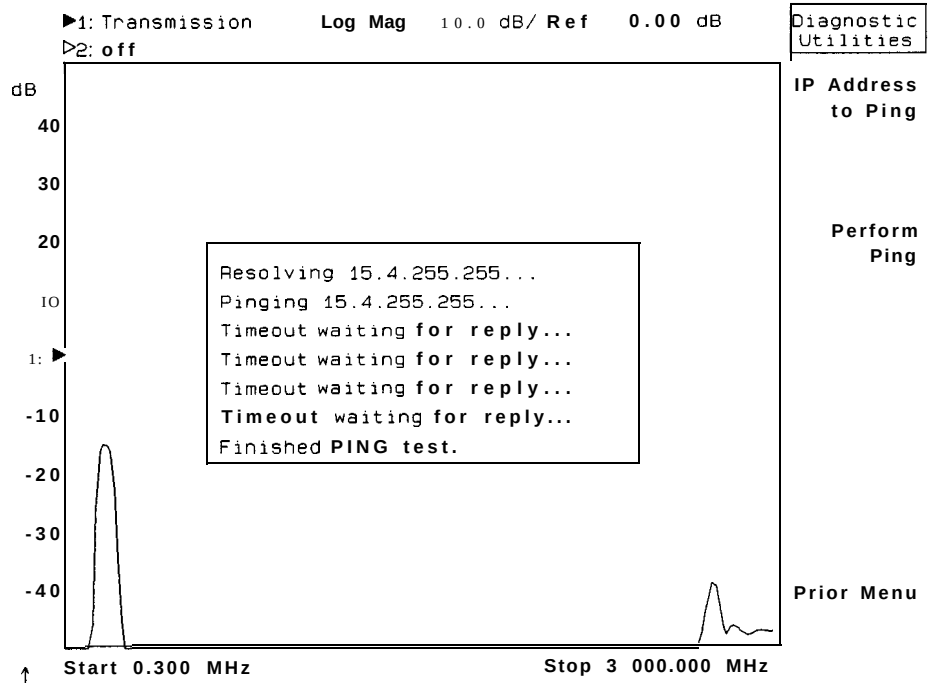


Figure 7-2. Example of a Failed Ping

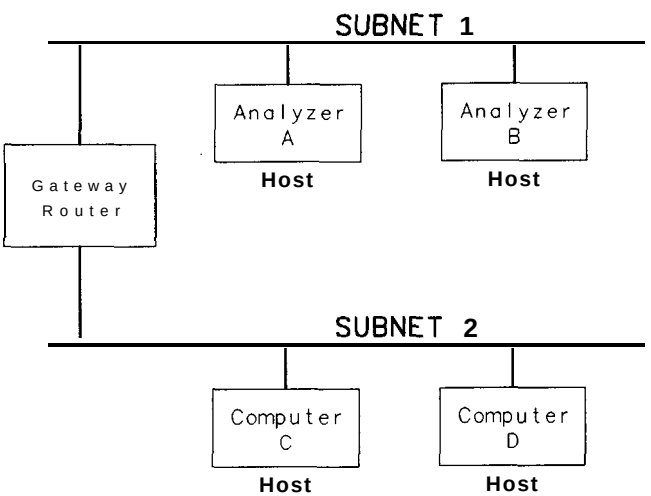
Subnets and Gateways

When you configure your analyzer as described in Chapter 1, you should enter a value for Gateway **IP Address** and **Subnet Mask** if they apply to your LAN. Your network administrator can tell you if you need to enter these values, and will supply you with the values to enter. This section provides some basic information on subnetting and gateways.

In large systems, the LAN is often split into subnets. Each subnet is isolated from other subnets using a router. Each subnet uses a unique and contiguous range of IP addresses for its hosts. The router acts as the "gateway" between the subnets. The router decides whether or not LAN traffic is allowed from one subnet to another.

Figure 7-3 shows a portion of a LAN system that includes a router (gateway) and two subnets, each including two hosts.

Each host is a unique device (such as a computer or an analyzer) with a unique IP address. The router also has a unique IP address.



php63c

Figure 7-3. Example of a LAN with Two **Subnets**

In order for a host on Subnet 1 to communicate with a host on Subnet 2, there are two configuration parameters that must be set up correctly:

Gateway IP Address-the address of the router

Subnet Mask-a number that allows the host to determine if communication is allowed

These parameters are used by the analyzer (host) and the router to define the IP address ranges used by each subnet, and by the router.

Refer again to Figure 7-3 for the following discussion:

You would like Analyzer A to communicate with Computer C. Note that they exist on different subnets, separated by a router.

Before Analyzer A tries to access Computer C, the analyzer looks at its subnet mask setting, and uses this mask to determine if Computer C is on the same LAN subnet.

If the analyzer determines from the subnet mask setting that Computer C is on the same subnet, then the analyzer establishes direct communication with Computer C. (It sends LAN packets directly to Computer C's IP address.)

If the analyzer determines that Computer C is on a different subnet from the analyzer (as in Figure 7-3), then the analyzer must send LAN packets to the router's IP address. The router then forwards the packets to Computer C.

Troubleshooting Subnet Problems

If your analyzer and computer are on separate LAN segments (subnets), separated by a gateway router, and you are experiencing difficulties in communicating:

- Be sure the analyzer's **Gateway** IP **Address** and **Subnet Mask** under the **LAN Port Setup** menu have been configured properly.

  See Chapter 1.

  Your network administrator should be able to tell you whether or not you need to enter these parameters, and should provide you with the correct numbers if you do.

- If you have configured the gateway address, your analyzer's IP address, and the subnet mask properly, but are still having problems:

  1. Connect your computer and analyzer directly to each other (with no gateway routers between them). This can be done by connecting the computer and analyzer to the same subnet, or by the use of a "cross-over" cable. See "Point-to-Point Connections" in Chapter 1.
  2. Configure *both* your computer and your analyzer so that they are both using a subnet mask value of 0.0.0.0, thus disabling gateway routing.
  3. Now try the ping test in both directions as described in "Troubleshooting the Initial Connection" earlier in this chapter. If it works, and it didn't before, you've determined that you have a problem with subnetting. Contact your network administrator for assistance.

Solutions to Common Problems

This section describes common problems you may encounter when using the analyzer on a LAN. It assumes you have been able to connect to the analyzer in the past. If this is not so, refer to the previous sections first.

If you cannot connect to the analyzer

If you suspect a bad LAN connection between your computer and analyzer, you can verify the network connection by using the ping command described earlier in this chapter or another similar echo request utility.

If a bad connection is revealed:

- Make sure the analyzer is turned on.
- Make sure the analyzer's **LAN State** is set to ON. (Press [SYSTEM OPTIONS] **LAN** .)
- Check the physical connection to the LAN.
- Make sure the Internet (IP) Address of the analyzer is set up correctly in the LAN Port Setup menu. (Press [SYSTEM OPTIONS] **LAN** **LAN Port Setup** .)
- If the analyzer and the computer are on different networks or subnets, make sure the gateway address and subnet mask values are set correctly. See "Troubleshooting Subnet Problems" earlier in this chapter.

If you cannot access the file system via ftp

If you get a "connection refused" message:

- If the power to the analyzer was just turned on, make sure that you wait about 25 seconds before attempting the connection.

If you get a "connection timed out" message:

- Verify the LAN connection between your computer and the analyzer. Refer to "If you cannot connect to the analyzer" earlier in this section.

If you cannot telnet to the command parser port

If you get a "connection refused" message:

- Try including the telnet port number (23) in the command.

If you get a "connection timed out" or "no response from host" message:

- Verify the LAN connection between your computer and the analyzer. Refer to "If you cannot connect to the analyzer" earlier in this section.

If you get a "connection refused" or "no response from host" message:

- If the analyzer was just turned on, make sure that you wait about 25 seconds, before attempting the connection.

If you get an "operation timed-out" message

- Check the LAN connection between the computer and the analyzer. Refer to "If you cannot connect to the analyzer" in this section.
- Increase the file time-out value on your PC or workstation.

If all else fails

- Contact your network administrator.
- If you still cannot solve the problem, contact an HP Service Center for repair information.

Getting Service Support

This section provides information about support services.

HP on-site service

With HP on-site service, HP pays for parts, labor, and travel to have an HP service representative visit your site for repairing equipment under warranty. You can purchase on-site service for the analyzer. Support contracts are also available for either 4 hour response or next day response. The support contracts are available for economical support beyond the 90-day warranty period. Contact your nearest Hewlett-Packard service or sales office for information regarding support contracts. See Table 7-1.

Return to HP service

The benchtop analyzers default to return to HP service. With return to HP service, you return the equipment to your nearest Hewlett-Packard service center for repair. During the warranty period, HP pays for parts and labor needed for repair. After the warranty period, you are billed for the parts and labor.

Table 7-1. Hewlett-Packard Sales and Service Offices

UNITED STATES

Instrument Support Center
Hewlett-Packard Company
(800) 403-0801

EUROPEAN FIELD OPERATIONS

Headquarters
Hewlett-Packard S.A.
150, Route du Nant-d'Avril
1217 Meyrin 2/Geneva
Switzerland
(41 221 780.8111

France
Hewlett-Packard France
1 Avenue Du Canada
Zone D'Activite De Courtaboeuf
F-91947 Les Ulis Cedex
France
(33 1) 69 82 60 60

Germany
Hewlett-Packard GmbH
Hewlett-Packard Strasse
61352 Bad Homburg v.d.H
Germany
(49 61721 16-0

Great Britain
Hewlett-Packard Ltd.
Eskdale Road, Winnersh Triangle
Wokingham, Berkshire RG415DZ
England
(44 7341 696622

INTERCON FIELD OPERATIONS

Headquarters
Hewlett-Packard Company
3495 Deer Creek Road
Palo Alto, Californie, USA
943041316
(415) 857-5027

Australia
Hewlett-Packard Australia Ltd.
3141 Joseph Street
Blackburn, Victoria 3130
161 3) 895-2895

Canada
Hewlett-Packard (Canada) Ltd.
17500 South Service Road
Trans- Canada Highway
Kirkland, Quebec H9J 2X8
Canada
15141 697-4232

China
China Hewlett-Packard Company
38 Bei Sen Huan XI Road
Shuang Yu Shu
Hai Dian District
Beijing, China
186 1) 256-6888

Japan
Hewlett-Packard Japan, Ltd.
9-1 Tekekura-Cho, Hechioji
Tokyo 192, Japan
181 4261 60-2111

Singapore
Hewlett-Packard Singapore (Pte.) Ltd.
150 Beach Road
#29-00 Geteway West
Singepore 0718
(65) 291-9088

Taiwan
Hewlett-Packard Taiwan
8th Floor, H-P Building
337 Fu Hsing North Road
Taipei, Taiwan
(886 2) 712-0404

Glossary

Glossary

10Base-T
A physical network connection that uses twisted pair cables with RJ-45 connectors.

Ethernet
A network that adheres to the IEEE 802.3 Local Area Network standard.

Ethernet address
A hexadecimal number which is used to identify a machine on a network. Each analyzer is assigned a unique Ethernet address at the factory and it is stored in the analyzer's ROM.

Ethertwist
See 10Base-T.

FTP
(File Transfer Protocol) A service that allows you to remotely transfer files among different operating systems.

ftp
(File Transfer Program) A file transfer program that uses file transfer protocol.

gateway
A generic term usually referring to a router.

host
A computer or device on a network.

host name
A unique name that is used to identify each host machine on a network. The host name is created by the user or the system administrator. The hostname is directly linked to a specific IP address, and can usually be used in place of the harder-to-remember IP address.

http
HyperText Transfer Protocol-used to carry World Wide Web (WWW) traffic.

internet
The connection of two or more distinct networks. Often a gateway or router is used to make the connection.

Internet
The largest internet (see above) in the world, connecting millions of networks. The Internet uses the TCP/IP protocol.

IP address
(Internet Protocol Address) A unique number that is assigned to each device which is to be connected to a TCP/IP network. Before using your analyzer on a network, your network administrator will need to assign an IP address.

An IP address consists of a 32-bit value presented in decimal dot notation: 4 octets (bytes) separated by a dot.

For example, the binary address 10000000 00000111 00001111 00000001 has the decimal dot notation of 128.7.15.1

network administrator
Similar to system administrator.

ping
A utility that allows you to determine the status of the connections between devices and a network. The ping utility is usually included with software packages that provide networking services. Your analyzer has a ping utility included in its firmware.

protocol
A set of conventions that specify how information will be formatted and transmitted on a network, and how machines on a network will communicate.

router
A device that moves traffic from one network to another.

server
A device that is configured to provide a service to other devices on network, such as shared access to a file system or printer. The analyzer acts like a server on your network.

socket
An endpoint for communication over a network.

system administrator
A person who manages systems and machines on a network. The system administrator is responsible for installing software and hardware on the network and assigning addresses and names to machines.

TCP/IP
(Transmission Control Protocol/Internet Protocol) A set of standards for communications between computers and between networks.

telnet
A protocol that allows users to create a session to run programs on or transfer information to and from a remote computer.

ThinLAN
A physical network connection that uses coax cables with BNC connectors.

time out
A period of system inactivity during which the system awaits user or network response. If there is no response by the end of the period, the system takes an action.

Index

Index