Errata

Title & Document Type: 8752A/B Operating & Programming Manual

Manual Part Number: 08752-90054P1

Revision Date: August 1991

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. The HP XXXX referred to in this document is now the Agilent XXXX. For example, model number HP8648A is now model number Agilent 8648A.

About this Manual

We've added this manual to the Agilent website in an effort to help you support your product. This manual provides the best information we could find. It may be incomplete or contain dated information, and the scan quality may not be ideal. If we find a better copy in the future, we will add it to the Agilent website.

Support for Your Product

Agilent no longer sells or supports this product. You will find any other available product information on the Agilent Test & Measurement website:

www.tm.agilent.com

Search for the model number of this product, and the resulting product page will guide you to any available information. Our service centers may be able to perform calibration if no repair parts are needed, but no other support from Agilent is available.

HP 8752A/B NETWORK ANALYZER OPERATING AND PROGRAMMING MANUAL

SERIAL NUMBERS

This manual applies directly to any instruments with the following serial prefix numbers:

HP 8752A – 2901A and above
HP 8752B – 3038A and above

If the serial number of your instrument is not listed on this page, refer to the "Instrument History" section of this manual.

1400 FOUNTAINGROVE PARKWAY, SANTA ROSA, CA 95403 U.S.A.

SERVICE MANUAL PART NO. 08752-90054
P/O Operating and Service Manual Set Part No. 08752-90053

Printed: AUGUST 1991
EDITION 2

HEWLETT PACKARD

CERTIFICATION

Hewlett-Packard Company certifies that this product met its published specifications at the time of shipment from the factory. Hewlett-Packard further certifies that its calibration measurements are traceable to the United States National Institute of Standards and Technology (NIST, formerly NBS), to the extent allowed by the Institute's calibration facility, and to the calibration facilities of other International Standards Organization members.

WARRANTY

This Hewlett-Packard system product is warranted against defects in materials and workmanship for a period corresponding to the individual warranty periods of its component products. Computer and computer peripherals are warranted for a period of 90 days. Instruments are warranted for a period of one year. During the warranty period, Hewlett-Packard Company will, at its option, either repair or replace products which prove to be defective.

Warranty service for products installed by HP and certain other products designated by HP will be performed at Buyer's facility at no charge within HP service travel areas. Outside HP service travel areas, warranty service will be performed at Buyer's facility only upon HP's prior agreement and Buyer shall pay HP's round trip travel expenses. In all other areas, products must be returned to a service facility designated by HP.

For products returned to HP for warranty service, Buyer shall prepay shipping charges to HP and HP shall pay shipping charges to return the product to Buyer. However, Buyer shall pay all shipping charges, duties, and taxes for products returned to HP from another country.

HP warrants that its software and firmware designated by HP for use with an instrument will execute its programming instructions when properly installed on that instrument. HP does not warrant that the operation of the instrument, or software, or firmware will be uninterrupted or error free.

LIMITATION OF WARRANTY

The foregoing warranty shall not apply to defects resulting from improper or inadequate maintenance by Buyer, Buyer-supplied software or interfacing, unauthorized modification or misuse, operation outside of the environmental specifications for the product, or improper site preparation or maintenance.

NO OTHER WARRANTY IS EXPRESSED OR IMPLIED. HP SPECIFICALLY DISCLAIMS THE IMPLIED WARRANTIES OR MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE.

EXCLUSIVE REMEDIES

THE REMEDIES PROVIDED HEREIN ARE BUYER'S SOLE AND EXCLUSIVE REMEDIES. HP SHALL NOT BE LIABLE FOR ANY DIRECT, INDIRECT, SPECIAL, INCIDENTAL, OR CONSEQUENTIAL DAMAGES, WHETHER BASED ON CONTRACT, TORT, OR ANY OTHER LEGAL THEORY.

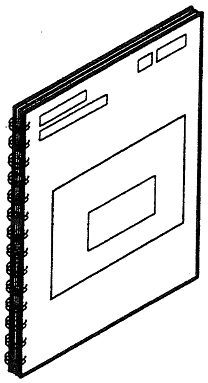
ASSISTANCE

Product maintenance agreements and other customer assistance agreements are available for Hewlett-Packard products.

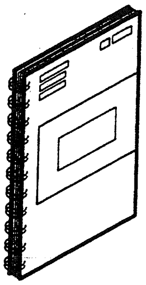
For any assistance, contact your nearest Hewlett-Packard Sales and Service Office. Addresses are provided at the back of this manual.

BP24.2

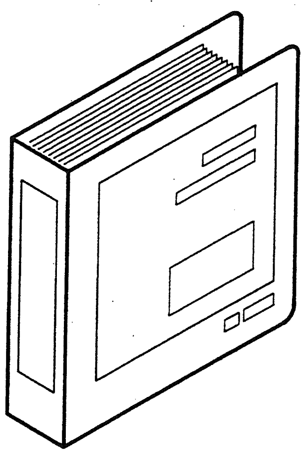
HP 8752A Network Analyzer Documentation Map



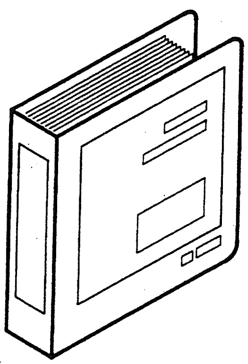
The **User's Guide** walks you through system setup and initial power-on, shows how to make basic measurements, explains commonly-used features, and tells you how to get the most performance from your analyzer.



The **User's Quick Reference** provides a summary of all available user features. It is organized alphabetically by front panel key.



The **Operating Manual** provides general information, specifications, HP-IB Programming information, and in-depth reference information.



The **Service Manual** explains how to verify conformance to published specifications, adjust, troubleshoot, and repair the instrument.

HP 8752A/B Operating Manual
TABLE OF CONTENTS

SAFETY CONSIDERATIONS

GENERAL

This product and related documentation must be reviewed for familiarization with safety markings and instructions before operation. This product has been designed and tested in accordance with international standards.

SAFETY SYMBOLS

Instruction manual symbol: the product will be marked with this symbol when it is necessary for the user to refer to the instruction manual (refer to Table of Contents).

Indicates hazardous voltages.

Indicates earth (ground) terminal.

WARNING

The WARNING sign denotes a hazard. It calls attention to a procedure, practice, or the like, which, if not correctly performed or adhered to, could result in personal injury. Do not proceed beyond a WARNING sign until the indicated conditions are fully understood and met.

CAUTION

The CAUTION sign denotes a hazard. It calls attention to an operating procedure, practice, or the like, which, if not correctly performed or adhered to, could result in damage to or destruction of part or all of the product. Do not proceed beyond a CAUTION sign until the indicated conditions are fully understood and met.

SAFETY EARTH GROUND

This is a Safety Class I product (provided with a protective earthing terminal). An uninterruptible safety earth ground must be provided from the main power source to the product input wiring terminals, power, cord, or supplied power cord set. Whenever it is likely that the protection has been impaired, the product must be made inoperative and secured against any unintended operation.

BEFORE APPLYING POWER

Verify that the product is configured to match the available main power source per the input power configuration instructions provided in this manual.

If this product is to be energized via an autotransformer make sure the common terminal is connected to the neutral (grounded side of the mains supply).

SERVICING

WARNING

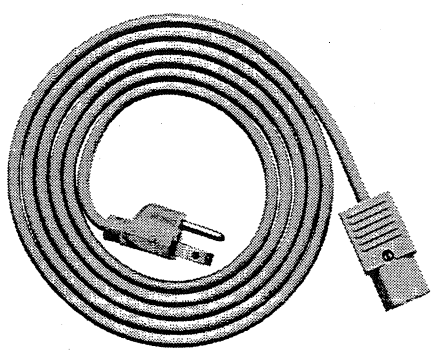
Any servicing, adjustment, maintenance, or repair of this product must be performed only by qualified personnel.

Adjustments described in this manual may be performed with power supplied to the product while protective covers are removed. Energy available at many points may, if contacted, result in personal injury.

Capacitors inside this product may still be charged even when disconnected from their power source.

To avoid a fire hazard, only fuses with the required current rating and of the specified type (normal blow, time delay, etc.) are to be used for replacement.

HP 8752 RF Network Analyzer



Power Cable*

* Power cable/plug supplied depends on country of destination

Figure 1-1. HP 8752 Network Analyzer with Power Cable Supplied

Chapter 1. General Information

CONTENTS

INTRODUCTION

The manual set contains an *Operating Manual* and a *Service Manual*.

Operating Manual

- **User's Guide**
- **Quick Reference**
- **HP-IB Programming**
- **Reference**

The **User's Guide** walks you through system setup and initial power-on, shows how to make basic measurements, and explains commonly-used features. The tutorials inside the *User's Guide* are for inexperienced users.

The **Quick Reference** briefly describes all instrument softkey menus and functions.

The **HP-IB Programming** section explains how to control the analyzer from an external computer.

The **Reference** provides in-depth information on each instrument key and softkey. This section also explains complex subjects such as measurement calibration, time domain, and more.

Service Manual

The *Service Manual* provides the following information:

- How to verify instrument performance.
- How to perform a quick check of instrument functions.
- How to adjust the instrument.
- How to troubleshoot and repair the instrument.
- How to order replacement parts.

HP 8752 DESCRIPTION

CAUTION

A properly grounded AC outlet is mandatory when operating the HP 8752. Operating the instrument with an improperly grounded or floating ground prong may damage the instrument!

CAUTION

Mating a 50Ω(m) connector with a 75Ω(f) connector will DESTROY the 75Ω female.

General

The HP 8752 is a high performance vector network analyzer for laboratory or production measurements of transmission and reflection parameters. It integrates a synthesized RF source and two separate measurement channels, a built-in transmission/reflection test set, and color display. No external equipment is required. An RF cable is supplied.

The HP 8752A provides 50 ohm transmission/reflection measurements and displays magnitude, phase, and group delay responses of active and passive RF networks. The HP 8752B provides the same features, but with a nominal system impedance of 75 ohms.

Two independent display channels and a large color screen display the measured results of one or both channels, in rectangular or polar/Smith chart formats.

Measurement functions are selected with front panel keys and softkey menus. Displayed measurement results can be printed or plotted directly to a compatible peripheral without the use of an external computer. Instrument states can be saved in internal memory for at least three days. In addition, the instrument can control a compatible disk drive for external storage capability.

Plotter/Printer Buffer

The buffer allows a plot or print-out to be made while the instrument continues to make measurements.

Noise Reduction and Measurement Calibration

Trace math, data averaging, trace smoothing, electrical delay, and measurement calibration improve the performance and flexibility of the instrument.

The Instrument and Supplied Cable are a Matched Set

The network analyzer contains error correction factors for the supplied RF test cable. These factors are very specific, and apply to the cable actually shipped with the instrument. The error correction factors should be updated once per year, or whenever the cable is replaced. The procedure for doing this is supplied in the "Adjustments" chapter of the service manual. The internal correction feature is sensitive to the measurement setup, Figure 1-2 shows the optimum test setup.

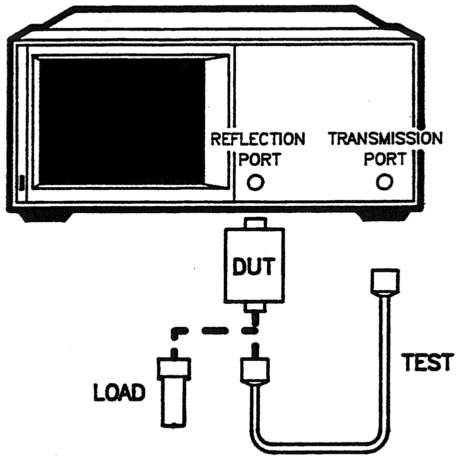


Figure 1-2. Optimum Test Setup for Use with Internal Error Correction

AUTOMATED OPERATION WITHOUT AN EXTERNAL COMPUTER

The HP 8752 can control HP-IB compatible printers, plotters, and disk drives without the use of an external computer.

The test sequence function allows the operator to save all keystrokes in a particular measurement task, and have the HP 8752 perform them automatically at a later time. This feature combines simple operation with advanced features such as limit testing decisions, conditional branching, user-defined prompts, and others. Sequences may be stored to an optional external disk drive. Refer to the "User's Guide" or the "Reference," located in the *Operating Manual*.

AUTOMATED OPERATION WITH AN EXTERNAL COMPUTER

This instrument can be remotely controlled by an external computer. The computer can control all instrument functions with the exception of some internal tests. The computer and instrument communicate over the Hewlett-Packard Interface Bus (HP-IB). Several output modes are available for outputting data. Through a subset of HP-GL (Hewlett-Packard Graphics Language), user graphics can be plotted on the instrument screen. A complete general description of HP-IB is available in *Condensed Description of the Hewlett-Packard Interface Bus* (HP part number 59401-90030), and in *Tutorial Description of the Hewlett-Packard Interface Bus* (HP literature number 5952-0156).

Specific Computer Requirements

This network analyzer can be controlled by several types of computers:

- HP 9000 series 200 or 300 computer.
- HP Vectra Personal Computer using an HP 82300 HP BASIC Language Processor.
- MS-DOS® – compatible personal computer (PC), equipped with the HP 82990A "HP-IB Interface and Command Library." Microsoft® QuickBASIC 4.0 is known to be fully compatible with the HP 82990A.

EQUIPMENT REQUIRED

The HP 8752 is ready to use as shipped. No extra equipment is required.

OPTIONS

Hardware options

Option 003, 3 GHz Operation. This option extends the maximum frequency of the instrument to 3 GHz.

Option 010, Time Domain. Instruments equipped with option 010 can display the time domain response of a network by computing the inverse Fourier transform of the frequency domain response. The response of a test device can be viewed as a function of time or distance.

Option 802, External Disk Drive. This provides an HP 9122 dual 3.5 inch double-sided disk drive. A 1-meter (3.3 foot) HP-IB cable is supplied.

Option 908, Rack Mount Kit for Instruments without Handles. Option 908 is a rack mount kit containing a pair of flanges and the necessary hardware to mount the instrument, with handles detached, in an equipment rack with 482.6 mm (19 inches) horizontal spacing. This kit is separately available; order HP part number 5062-3978.

Option 913, Rack Mount Kit for Instruments with Handles. Option 913 is a rack mount kit containing a pair of flanges and the necessary hardware to mount the instrument with handles attached in an equipment rack with 482.6 mm (19 inches) spacing. This kit is separately available; order HP part number 5062-4072.

Option 910, Extra Manual Set. The standard instrument is supplied with an *Operating and Service Manual* set. Option 910 provides an additional manual set. After initial shipment, order extra manuals by part number. The numbers are listed on the title pages of the manuals and in the "Replaceable Parts" section of the *Service Manual*.

Warranty and Support Options

The HP 8752 includes a one-year on-site service warranty, where available. The "system" consists of an HP 8752 with a test port cable and a calibration kit (if purchased).

Three Year Customer Return Repair Coverage (W30) provides a total of three years of return-to-HP repair service from the time of product delivery.

Three Year On-Site Repair Coverage (W31) provides a total of three years of next day on-site coverage from the time of product delivery.

Three Year Customer Return Calibration Coverage (W32) provides scheduled calibrations in second and third years. Also provides full calibration after a repair performed by HP. Coverage begins at time of product delivery.

Tool Kit. A dedicated tool kit is available for HP 8752 troubleshooting, consisting of extender boards, extender cables, and adapters. The contents of the tool kit are listed in the *Service Manual*.

MEASUREMENT ACCESSORIES

Calibration Kits

Because of the built-in correction, calibration kits are optional. Measurement calibration requires the use of high-quality measurement standards. This makes it possible for the HP 8752 to calculate and compensate for measurement inaccuracies caused by the test setup. The following calibration kits contain precision standard devices with different connector types.

HP 85032B Option 001 50 Ohm Type-N Calibration Kit. The precision standards in this kit are used to calibrate the HP 8752A for measuring devices with type-N connectors. The kit consists of the following standards:

NOTE: The standard HP 85032B can be used instead of the option 001 version if desired (option 001 deletes 7 mm to type-N adapters).

(1) type-N (m) 50 ohm termination
(1) type-N (f) 50 ohm termination
(1) type-N (m) short circuit
(1) type-N (f) short circuit
(1) type-N (m) open circuit with center conductor extender
(1) type-N (f) open circuit

HP 85033C Option 001 3.5 mm Calibration Kit. This kit contains precision standards used to calibrate the HP 8752A for measuring devices with precision 3.5 mm connectors. The kit consists of the following standards:

(1) 3.5 mm (m) 50 ohm termination
(1) 3.5 mm (f) 50 ohm termination
(1) 3.5 mm (f) short circuit
(1) 3.5 mm (m) short circuit
(1) 3.5 mm (f) open circuit with center conductor extender
(1) 3.5 mm (m) open circuit with center conductor extender

In addition to the HP 85033C option 001, the HP 8752A requires:

(2) Type N to 3.5 mm (m) adapters
(2) Type N to 3.5 mm (f) adapters

(The HP 11878A adapter kit supplies these four adapters.)

HP 85036B 75 Ohm Type-N Calibration Kit. The standards in this kit are used to calibrate the HP 8752B. The kit consists of the following standards:

(1) type-N (m) 75 ohm termination
(1) type-N (f) 75 ohm termination
(1) type-N (m) 75 ohm short circuit
(1) type-N (f) 75 ohm short circuit
(1) type-N (m) 75 ohm open circuit
(1) type-N (f) 75 ohm open circuit with center conductor extender

Included in the HP 85036B cal kit are the following adapters:

(1) Type-N (m) to Type-N (m) 75 ohm adapter
(1) Type-N (f) to Type-N (f) 75 ohm adapter
(1) Type-N (m) to Type-N (f) 75 ohm adapter

Verification Kit

Accurate operation of the HP 8752 system can be verified by measuring known devices and comparing the results with recorded data. The HP 85032B (standard or option 001) is required for the HP 8752A. The HP 85036B is required for the HP 8752B. A system verification procedure is provided in the *Service Manual*.

Test Port Return Cable

Type-N RF Cable. This is a replacement type-N cable (the cable supplied with the HP 8752). Order the cable if the connectors on your original cable have been damaged or show wear. HP part numbers: 8120-4781 (50 ohm for HP 8752A), or 8120-2408 (75 ohm for HP 8752B).

Adapter Kits

HP 11852B 50 to 75 Ohm Minimum Loss Pad. This device converts impedance from 50 ohms to 75 ohms or from 75 ohms to 50 ohms. It is used to provide a low SWR impedance match between a 75 ohm device under test and the HP 8752A network analyzer or a 50 ohm measurement accessory. The HP 11852B may also be used to provide a good SWR impedance match between the HP 8752B and a 50 ohm measurement accessory.

The HP 8752A, a 50 ohm system, can measure 75 ohm devices if used with two HP 11852B minimum loss pads and a 75 ohm calibration kit (HP 85036B). The HP 8752B, a 75 ohm system, measures 75 ohm devices directly and uses a 75 ohm calibration kit (HP 85036B).

HP 11853A 50 Ohm Type-N Adapter Kit. This kit contains the connecting hardware required for making measurements on devices with 50 ohm type-N connectors.

HP 11854A 50 Ohm BNC Adapter Kit. This kit contains the connecting hardware required for making measurements on devices with 50 ohm BNC connectors.

HP 11855A 75 Ohm Type-N Adapter Kit. This kit contains the connecting hardware required for making measurements on devices with 75 ohm type-N connectors.

HP 11856A 75 Ohm BNC Adapter Kit. This kit contains the connecting hardware required for making measurements on devices with 75 ohm BNC connectors.

HP 11878A 50 Ohm 3.5 mm Adapter Kit. This kit contains the connecting hardware required for making measurements on devices with 50 ohm 3.5 mm connectors.

SYSTEM ACCESSORIES

System Rack

The HP 85043B system rack is a 124 cm (49 inch) high metal cabinet designed to rack mount the HP 8752 in a system configuration. The rack is equipped with a large built-in work surface, a drawer for calibration kits and other hardware, a bookshelf for system manuals, and a locking rear door for secured access. Lightweight steel instrument support rails support the instruments along their entire depth. Heavy-duty casters make the cabinet easily movable even with the instruments in place. Screw-down lock feet permit leveling and semi-permanent installation: the cabinet is extremely stable when the lock feet are down. Power is supplied to the cabinet through a heavy-duty grounded primary power cable, and to the individual instruments through special power cables included with the cabinet.

Plotters and Printers

The HP 8752 is capable of plotting or printing displayed measurement results directly to a compatible peripheral without the use of an external computer.

The compatible plotters are:

- HP 7440A Option 002 ColorPro Eight-Pen Color Graphics Plotter, plots on ISO A4 or 8 1/2 x 11 inch charts.
- HP 7475A Option 002 Six-Pen Graphics Plotter, plots on ISO A4/A3 or 8 1/2 x 11 inch or 11 x 17 inch charts.
- HP 7550A High-Speed Eight-Pen Graphics Plotter, plots on ISO A4/A3 or 8 1/2 x 11 inch or 11 x 17 inch charts.
- HP 7090 Measurement Plotting System, is a high-performance six-pen programmable digital plotter. It plots on ISO A4/A3 or 8.5 x 11 inch or 11 x 17 inch paper or overhead transparency film.

The compatible printers for both printing and plotting are:

- HP 3630A PaintJet option 002 color printer
- HP 2225A (HP-IB compatible) ThinkJet printer
- HP 2227B QuietJet option 002 printer

Disk Drives

The HP 8752 has the capability of storing instrument states directly to an external disk drive without the use of a computer. Any disk drive that uses CS80 protocol and HP 200/300 series (LIF) format is compatible. Disks may be formatted directly by the HP 8752. An HP 9122 Dual 3.5 inch floppy disk drive is supplied when the HP 8752 option 802 is ordered. Another recommended disk drive is the HP 9153C 20 Megabyte Winchester disk drive.

HP-IB Cables

An HP-IB cable is required for interfacing the HP 8752 with a plotter, printer, external disk drive, or computer. The cables available are HP 10833A (1 m), HP 10833B (2 m), HP 10833C (4 m), and HP 10833D (0.5 m).

POWER AND ENVIRONMENTAL REQUIREMENTS

Rack Power Requirements. If the system has been ordered with an HP 85043B rack, power will be supplied to the rack through its heavy-duty grounded primary power cable, and to the individual instruments in the rack through special power cables included with the rack. The rack should be connected to a circuit capable of supplying 2000 VA without interruption, and without interference from other equipment such as air conditioners or large motors.

Replacing the Line Fuse. To remove the fuse housing, remove the power cord and insert a small screwdriver into the slot at the base of the fuse housing. Pull forward and out. A spare fuse is also supplied in the fuse housing.

Canadian Fuse Requirements. A 3.0 amp fuse is required for use in Canada, regardless of the line voltage.

Environmental Requirements. Some instrument specifications are only warranted if the instrument is operated at 25°C ±5°C. If a potential degradation in performance is acceptable, the instrument may be operated at the temperature range shown below:

- Temperature: Operating = 0°C to +55°C
  Storage = −40°C to +75°C
- Humidity: This product must not be placed in an environment which causes moisture condensation.

  The instrument should be protected from temperature and humidity conditions that might cause internal condensation.
- Altitude: Operating = up to 4500 metres (approximately 15,000 feet)
  Storage = up to 15240 metres (50,000 feet)
- Conducted and radiated interference are in compliance with FTZ 526/527/1979.

The system can be operated in environments outside this range with a possibility of degradation in performance and a higher risk of failure. For temperature limitations on specified performance, refer to the table of specifications.

In addition to the above requirements, the following considerations should be observed:

- The environment should be as dust-free as possible, and any air filters in the instruments or system cabinet should be cleaned regularly.
- Electrostatic discharge (ESD) should be controlled by use of static-safe work procedures. For bench installation, an antistatic bench mat and wrist strap will decrease the possibility of damage from ESD. Part numbers for these items are provided in the replaceable parts section of the *Service Manual*, and in the *User's Guide*.

Storing Rack-Mounted Systems. If the system is rack-mounted, it must be stored with the rack standing upright. If the cabinet is stored in any other position with the instruments installed, the stress may cause mechanical and electrical damage to the instruments. The cabinet can be wheeled about its immediate installation area with the instruments installed, but care must be used since the instruments are sensitive and heavy. Turn the leveling foot on each bottom corner of the HP 85043B cabinet so that the feet do not interfere with movement. For safety's sake, have another person help guide and steady the cabinet.

SERVICING AND SHIPPING THE INSTRUMENT

Servicing the Instrument

This manual is intended for use by the operator of the HP 8752. Operating personnel must not remove the instrument covers. The instrument should be serviced only by qualified personnel who are aware of the hazards involved. Detailed safety precautions are described in the *Service Manual*.

Packing for Shipment. If the instrument or any of the system components is to be returned to Hewlett-Packard for service, attach a tag indicating the service required, return address, instrument model number, and full serial number, then pack as described below. The return address must be a mailing address, not a post office box. Use the blue service tags located at the end of the *Operating Manual*. In any correspondence, refer to the instrument by model number and full serial number.

A rack cabinet should never be shipped with the instruments installed. All instruments should be removed and individually packaged before shipment.

If any instrument is to be reshipped, it is best to use the original factory packaging materials. If these are not available, you can order similar containers and materials from the nearest Hewlett-Packard office. If other packaging materials are used, be sure to wrap the instrument (with service tag) in heavy paper or anti-static plastic, and place the wrapped instrument in a strong shipping container such as a double-wall carton made of 350-pound test material. Pack a three to four inch layer of shock absorbing material around the instrument. Seal the carton securely, and mark it FRAGILE.

INSTRUMENTS DOCUMENTED BY THE MANUAL

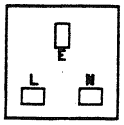
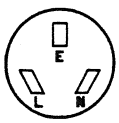
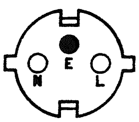
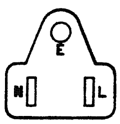
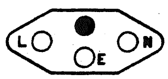
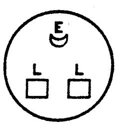
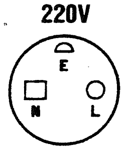
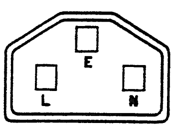
The serial number on the back of the instrument is composed of two parts. The first four digits and letter comprise the *prefix*; the last five digits comprise the *suffix*.

The prefix is the same for all identical instruments; it changes only when a change is made to the instrument. The suffix, however, is assigned sequentially and is different for each instrument. This manual applies directly to instruments with the serial number prefix or prefixes listed on the title page.

History information on past instrument versions (if any) is provided in the "Instrument History" chapter.

Refer any questions to the nearest Hewlett-Packard Sales/Service Office. Always identify the instrument by model number, complete name, and complete serial number in all correspondence. A worldwide listing of HP Sales/Service Offices is provided at the back of this volume.

AC Power Cables Available

Plug Type[1]	Cable HP Part Number[2]	CD[3]	Plug Description[2]	Cable Length (inches)	Cable Color	For Use in Country
250V 	8120-1351 8120-1703	0 6	Straight BS1363A 90°	90 90	Mint Gray Mint Gray	United Kingdom, Cyprus, Nigeria, Zimbabwe, Singapore
250V 	8120-1369 8120-0696	0 4	Straight ZNSS198/ASC112 90°	79 87	Gray Gray	Australia, New Zealand
250V 	8120-1689 8120-1692	7 2	Straight CEE7-VII 90°	79 79	Mint Gray Mint Gray	East and West Europe, Saudi Arabia, Egypt, Republic of So. Africa, India (unpolarized in many nations)
125V 	8120-1348 8120-1398 8120-1754 8120-1378 8120-1521 8120-1676	5 5 7 1 6 2	Straight NEMA5-15P 90° Straight NEMA5-15P Straight NEMA5-15P 90° Straight NEMA5-15P	80 80 36 80 80 36	Black Black Black Jade Gray Jade Gray Jade Gray	United States, Canada, Japan (100V or 200V), Mexico, Philippines, Taiwan
250V 	8120-2104	3	Straight SEV1011.1959 24507,Type 12	79	Gray	Switzerland
250V 	8120-0698	6	Straight NEMA6-15P			United States, Canada
220V 	8120-1957 8120-2956	2 3	Straight DHCK 107 90°	79 79	Gray Gray	Denmark
250V 	8120-1860	6	Straight CEE22-VI (System Cabinet Use)			

1. E = Earth Ground; L = Line; N = Neutral
2. Part number shown for plug is industry identifier for plug only. Number shown for cable is HP Part Number for complete cable including plug.
3. The Check Digit (CD) is a coded digit that represents the specific combination of numbers used in the HP Part Number. It should be supplied with the HP Part Number when ordering any of the power assemblies listed above, to expedite speedy delivery.

Chapter 2. Specifications

INTRODUCTION

This chapter provides information on the stand-alone specifications and characteristics of the analyzer, and on the specified and typical performance of the analyzer system. The analyzer system consists of the analyzer plus the test cable supplied with (and matched to) the analyzer.

ANALYZER SPECIFICATIONS AND CHARACTERISTICS

Table 2-1 lists specifications and typical characteristics of the stand-alone analyzer (without its matched cable). Specifications are the performance standards or limits against which the instrument can be tested. They are field verifiable using performance tests documented in the service manual.

Typical characteristics are not specifications but are non-warranted performance characteristics provided for use in applying the instrument. Typical characteristics are representative of most instruments, though not necessarily factory-tested in each unit. They are not field tested.

Table 2-2 lists supplemental characteristics of the instrument. These are general descriptive information provided for use in applying the instrument.

SYSTEM SPECIFICATIONS AND SYSTEM PERFORMANCE

Table 2-3 lists measurement port specifications for the system (analyzer and matched cable). These specifications are the residual system uncertainties with and without error correction.

The last part of this chapter explains system performance and residual measurement errors. It provides a description of error sources, graphs of typical measurement uncertainty, and information on determining expected performance for a particular system.

Table 2-1. Instrument Specifications (1 of 2)

Specifications describe the instrument's warranted performance over the temperature range of 0° to 55°C (except where noted). Figures denoted as typical are not specifications but are non-warranted performance parameters provided for use in applying the instrument.

SOURCE[1]

Frequency

Range:	standard	300 kHz to 1.3 GHz
	option 003	300 kHz to 3 GHz
Resolution:	1 Hz	
Stability:	typically ±7.5 ppm 0 to 55°C	
	typically ±3 ppm/year	
Accuracy:	10 ppm at 25°C ±5°C	

Output Power

Test port power range:	−20 to +5 dBm
Resolution:	0.1 dB
Flatness[2]:	2 dB Peak to Peak
Level accuracy[2]:	±0.5 dB at 50 MHz
Level linearity[2]:	±0.5 dB from −20 to −15 dBm
	±0.2 dB from −15 to 0 dBm
	±0.5 dB from 0 to +5 dBm

RECEIVER[1]

Input

Frequency range:	Standard	300 kHz to 1.3 GHz
	Option 003	300 kHz to 3 GHz
Dynamic range[3]:	100 dB	

Noise level:		
3 kHz BW	Reflection	−75 dBm (typical)
10 Hz BW	Reflection	−85 dBm (typical)
3 kHz BW	Transmission	−90 dBm
10 Hz BW	Transmission	−100 dBm (typically −110 dBm)

Maximum input level:	0 dBm at Transmission Port
	10 dBm at Reflection Port
Damage Level:	20 dBm or >25 VDC at Transmission Port
	30 dBm or >25 VDC at Reflection Port

Crosstalk: (Reflection Port to Transmission Port)
300 kHz to 1.3 GHz: 100 dB
1.3 to 3 GHz: 90 dB

Magnitude

Dynamic accuracy:	Refer to Figure 2-1a.
Display resolution:	0.01 dB/div.
Marker resolution[3]:	0.001 dB
Trace noise:	(0 dBm, 3 kHz BW; reflection): <0.006 dB rms
	(0 dBm, 3 kHz BW; transmission): <0.006 dB rms
Reference level:	range: ±500 dB
	resolution: 0.001 dB
Stability:	0.02 dB/°C typically

Phase

Dynamic accuracy: Refer to Figure 2-1b.
Range: ±180 degrees
Frequency response (deviation from linear, −10 dBm, 25°±5°C): ±3°
Display resolution: 0.01°/div.
Marker resolution[3]: 0.01°

Trace noise:	(0 dBm, 3 kHz BW; reflection): <0.035° rms
	(0 dBm, 3 kHz BW; transmission): <0.035° rms
Reference level: range:	±500 degrees
	resolution: 0.01 degrees
Stability:	0.1°/°C typically

1. HP 8752B option 003 specifications above 2 GHz are typical due to 75 ohm measurement standards having an upper frequency limit of 2 GHz
2. At 25°C±5°C relative to −5 dBm test port power.
3. This specification applies to transmission measurements in the 300 kHz to 1.3 GHz frequency range at 10 Hz IF bandwidth with response and isolation correction. Dynamic range is limited by maximum receiver input level and system noise floor.
4. Marker resolution is dependent upon the value measured; resolution is limited to five digits.

DYNAMIC ACCURACY

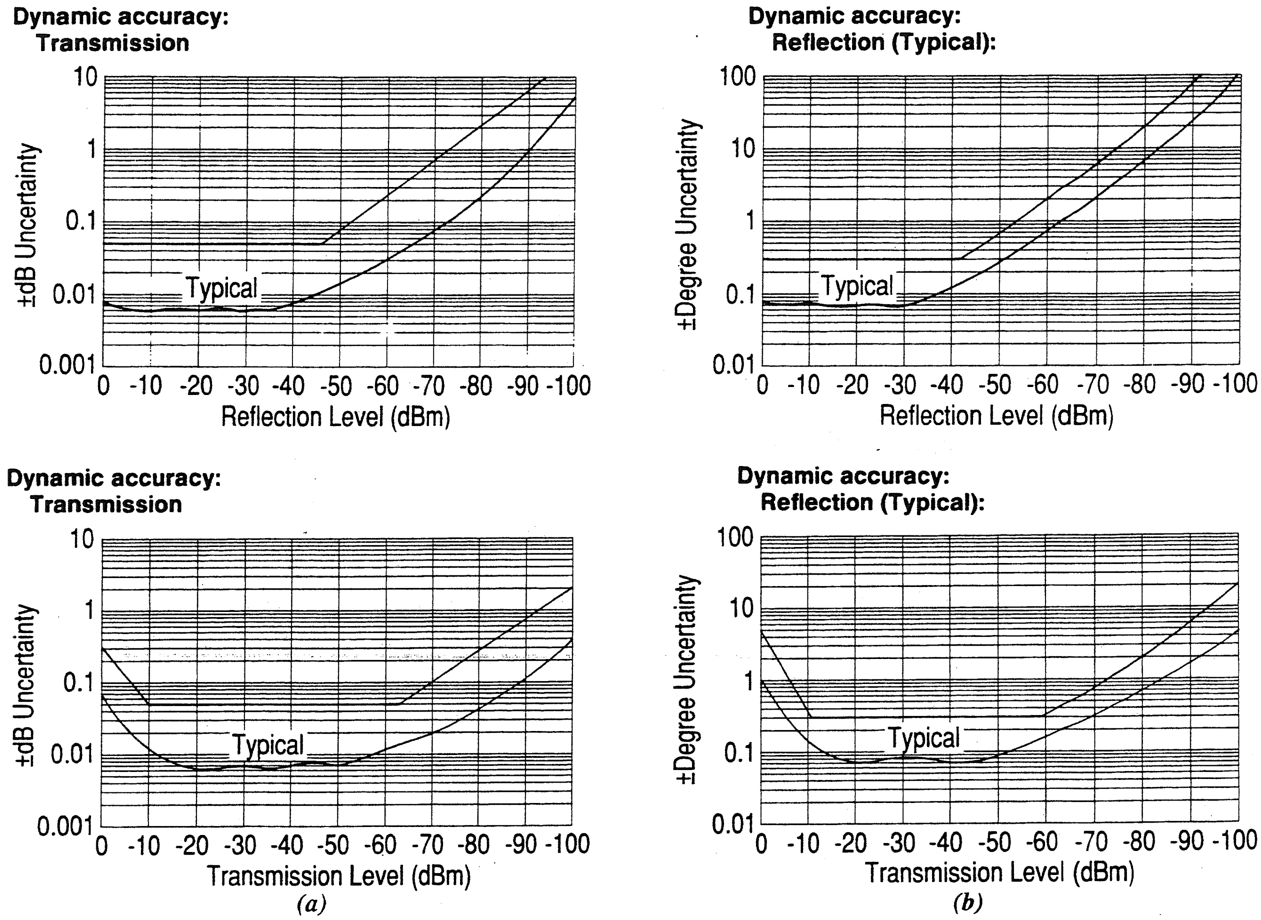


Figure 2-1. Receiver Dynamic Accuracy

Dynamic accuracy for both reflection (typical) and transmission (specified) is determined from the plots in Figure 2-1. The power along the x-axis is expressed in dBm and is equivalent to the power at the transmission port for transmission measurements. For reflection measurements it is the power re-entering the reflection port after reflecting off the DUT.

Once the x-axis value is determined, read the corresponding dynamic accuracy from the plot.

Example: Reflection Dynamic Accuracy

(Refer to Figure 2-1b.)

Test port power:	−10 dBm
DUT return loss:	20 dB
Reflected power	= (test port power)
	− (DUT return loss)
	= (−10 dBm) − (20 dB)
	= −30 dBm

Reflection Dynamic Accuracy at −30 dBm: ±0.06 dB

Example: Transmission Dynamic Accuracy

(Refer to Figure 2-1a.)

Test port power:	−10 dBm
DUT insertion loss:	10 dB
Transmitted power	= (test port power)
	− (DUT insertion loss)
	= (−10 dBm) − (10 dB)
	= −20 dBm

Transmission Dynamic Accuracy at −20 dBm: ±0.05 dB

Table 2-2. Supplemental Characteristics (1 of 4)

These are not specifications, but general descriptive information provided for use in applying the instrument.

MEASUREMENT THROUGHPUT SUMMARY

The following shows typical measurement times in milliseconds.

Typical time for completion (msec)

	Number of Points			
	51	**201**	**401**	**1601**
Measurement				
Uncorrected or 1-port cal[1]	120	190	290	890
2-port cal	540	1030	1680	5610
Time domain conversion[2]	125	540	1150	2840
HP-IB data transfer[3]				
Internal	30	50	75	660
ASCII	500	1900	3800	15000
IEEE 754 floating point format:				
32 bit, 32 bit MS-DOS®	40	85	140	500
64 bit	60	125	210	700

REMOTE PROGRAMMING

Interface

HP-IB interface operates according to IEEE 488-1978 and IEC 625 standards and IEEE 728-1982 recommended practices.

Transfer Formats

Binary (internal 48-bit floating point complex format)

CITIFile ASCII

32/64 bit IEEE 754 Floating Point Format

Interface Function Codes

SH1, AH1, T6, TE0, L4, LE0, SR1, RL1, PP0, DC1, DT1, C1, C2, C3, C10, E2

1 Reflection 1-Port calibration with a 3 kHz IF BW. Includes system retrace time but does not include bandswitch time. Time domain gating is assumed off.

2. Option 010 only, gating off.

3 Measured with HP 9000 Series 300 computer

GROUP DELAY CHARACTERISTICS

Group delay is computed by measuring the phase change within a specified frequency step (determined by the frequency span, and the number of points per sweep).

Aperture: Selectable
Maximum aperture: 20% of frequency span
Minimum aperture: (frequency span)/(number of points −1)

Range:
The maximum delay is limited to measuring no more than 180° of phase change within the minimum aperture.

Range = 1/(2 x minimum aperture)

Accuracy:
The following graph shows group delay accuracy at 1.3 GHz with an uncorrected measurement. IF BW is 10 Hz. Insertion loss is assumed to be small and device electrical length is 10 meters.

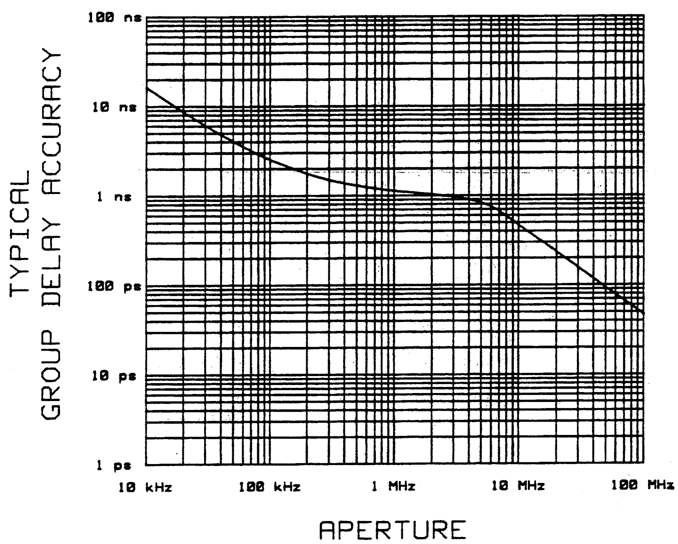


Figure 2-2. Group Delay Accuracy at 1.3 GHz

In general, the following formula can be used to determine the accuracy, in seconds, of a specific group delay measurement.

±(0.003 x Phase accuracy(deg)) / Aperture(Hz)

Depending on the aperture and device length, the phase accuracy used is either incremental phase accuracy or worst case phase accuracy. The above graph shows this transition.

FRONT PANEL CONNECTORS

	HP 8752A	HP 8752B
Connector Type	type-N (female)	type-N (female)
Impedance	50 ohms (nominal)	75 ohms (nominal)
Connector Pin Protrusion	0.204 to 0.207 in	0.204 to 0.207 in

REAR PANEL CONNECTORS

External Reference Frequency Input (EXT REF INPUT)

Frequency	1, 2, 5, and 10 MHz (±200 Hz @ 10 MHz)
Level	−10 dBm to +20 dBm, typical
Impedance	50 ohms

External Auxiliary Input (AUX INPUT)

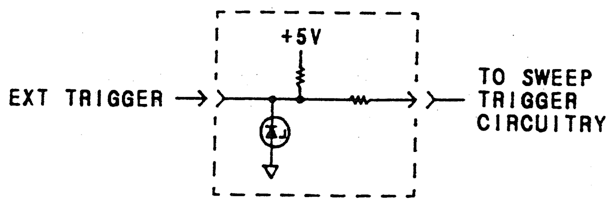
Input Voltage Limits −10V to +10V

External AM Input (EXT AM)

±1 volt into a 5k ohm resistor, 1 kHz maximum, resulting in 8 dB/volt amplitude modulation.

External Trigger (EXT TRIGGER)

Triggers on a negative TTL transition or contact closure to ground.



External Trigger Circuit

LINE POWER

47 to 66 Hz
115V nominal +10% −25% or 230V +10% −15%. 220 VA max.

PROBE POWER

+15V ±2%, 400 mA
−12.6V ±5.5%, 300 mA

ENVIRONMENTAL REQUIREMENTS

Operating Conditions

Temperature	0 to 55°C
Humidity	Operate in a non-condensing environment
Altitude	0 to 4500 meters (15,000 feet)

Non-Operating Storage Conditions

Temperature	−40°C to +70°C
Humidity	Store in a non-condensing environment
Altitude	0 to 15,240 metres (50,000 feet)

WEIGHT

Net	25.4 kg (56 lb)
Shipping	28.4 kg (63 lb)

CABINET DIMENSIONS

178 H x 425 W x 497.8 mm D
(7.0 x 16.75 x 19.0 in)

(These dimensions exclude front and rear panel protrusions. Add 24 mm (1 in) to depth to include the front panel connectors.)

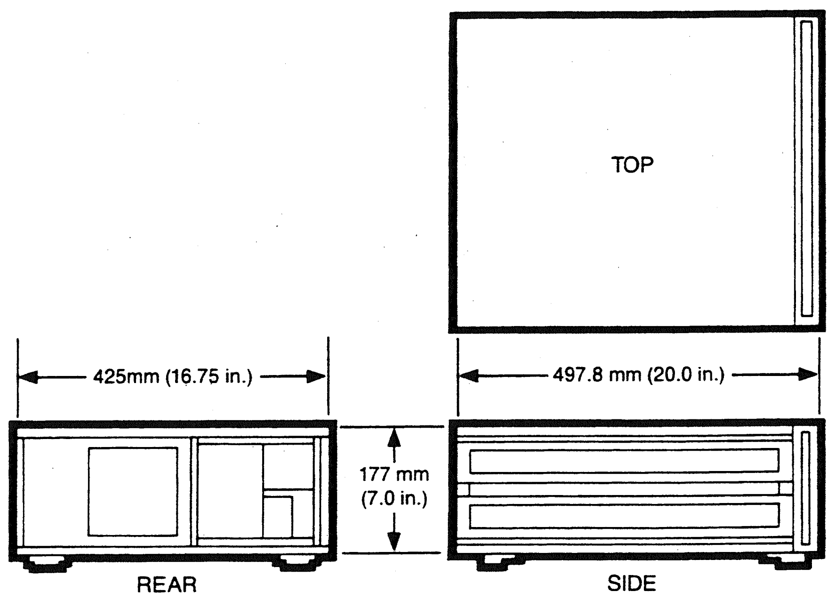


Table 2-3. System Specifications

MEASUREMENT PORTUNCERTAINTIES

The following specifications show the residual system uncertainties with and without error correction. The system is defined as the analyzer plus its matched cable. When two values are given for one specification (for example: 40/35), the number on the left applies up to 1.3 GHz, the number on the right applies from 1.3 to 3 GHz.

NOTE: All HP 8752B option 003 specifications above 2 GHz are typical due to 75 ohm measurement standards having an upper frequency limit of 2 GHz.

Error Term	Uncorrected[1]		3.5 mm[2,3]		Type-N[3]	
	dB	Linear	dB	Linear	dB	Linear
Directvity	−40/−35[4]	0.01/0.0178	−40	0.01	−44	0.0063
Reflection Tracking	±0.2/±0.3	0.0233/ 0.0351	±0.06	0.0069	±0.06	0.0069
Transmission Tracking	±0.2/±0.3	0.0233/ 0.351	±0.05/ ±0.1	0.0058/ 0.0116	±0.05/ ±0.1	0.0058/ 0.0116
Source Match (reflection)	−30/−25	0.0316/ 0.0562	−36	0.0158	−35	0.0178
Source Match (transmission)	−23/−20	0.0708/0.1	−23/−20	0.0708/0.1	−23/−20	0.0708/0.1
Load Match	−23/−20	0.0708/0.1	−23/−20	0.0708/0.1	−23/−20	0.0708/0.1
Crosstalk	−100/−90	0.00001/ 0.00003	−100/−90	0.00001/ 0.00003	−100/−90	0.00001/ 0.00003
Connector Repeatability (typical)	−65	0.0006	−70	0.0003	−65	0.0006

Error Term	dB	Linear
Noise Floor[5], Transmission: (included in dynamic accuracy)	−100 (10 Hz BW)	0.00001
Trace Noise	0.006 dB rms	0.0007
Cable Reflection Magnitude Stability (typical)	−60 dB	0.001

Error Term	Degrees
Cable Transmission Phase Stability (typical):	0.05 x Frequency (in GHz) degrees

1. These uncertainties apply in an environmental temperature of 23° +10/−5°C with an IF bandwidth of 10 Hz
2. HP 8752A only.
3. These uncertainties apply in an environmental temperature of 23° ±3°C, with less than 1° deviation from the temperature at measurement calibration. IF BW is 10 HZ.
4. −30 dB from 300 kHz to 10 MHz.
5. Noise floor is already included in dynamic accuracy performance data.

SYSTEM PERFORMANCE

INTRODUCTION

System performance depends not only on the performance of the analyzer and the cables, but also on operating conditions and on measurement errors inherent in network analysis.

The following pages provide a brief description of the sources of measurement errors, graphs of typical measurement uncertainty for the HP 8752A and 8752B, and information to use in determining the expected performance of a particular system.

Also provided are tips on increasing dynamic range by reducing associated measurement errors.

INCREASING DYNAMIC RANGE

Dynamic range is limited by the maximum receiver input level (the high end of the range), and by either of these two factors on the minimum end of the range:

- System noise floor
- Crosstalk

Noise Floor

System noise floor can be reduced using a narrow IF bandwidth or with averaging, or with a combination of both. These measures can reduce noise floor below the crosstalk error level, making crosstalk error the limiting factor in dynamic range. A response and isolation calibration can then reduce the crosstalk errors. *The noise floor must be less than the crosstalk error or a response and isolation calibration will not reduce crosstalk errors.* The "Calibration" chapter explains how to determine which is greater: noise floor or crosstalk

Crosstalk

Crosstalk is a factor only in measurements requiring wide dynamic range. When crosstalk is greater than the noise floor, a response and isolation calibration can reduce its effects, thereby increasing dynamic range. However, if the noise floor is greater than crosstalk, the response and isolation cal will have no effect on dynamic range. The "Calibration" chapter explains this in detail.

SOURCES OF MEASUREMENT ERRORS

Network analysis measurement errors can be separated into the following types of errors:

- Systematic errors.
- Random errors.
- Drift errors.

Systematic Error Sources

The model for systematic errors is shown below. Refer to the end of the chapter for reflection uncertainty and transmission uncertainty equations derived from the system error model. Systematic errors result from imperfections in the calibration standards, connector standards, connector interface, interconnecting cables, and the instrumentation. All measurements are affected by dynamic accuracy.

Errors in Reflection Measurements. Directivity, source match, and reflection tracking are the errors that affect reflection measurements.

Errors in Transmission Measurements. Crosstalk, source match, load match, transmission tracking, and cable stability are the errors that affect transmission measurements.

Refer to the "Measurement Calibration" chapter for an explanation of each of these individual errors.

Random Error Sources

Non-repeatable errors are trace noise, noise floor (included in dynamic accuracy), and connector repeatability. These errors affect both transmission and reflection measurements.

Drift Errors

The effects of temperature drift are included in the system specifications data.

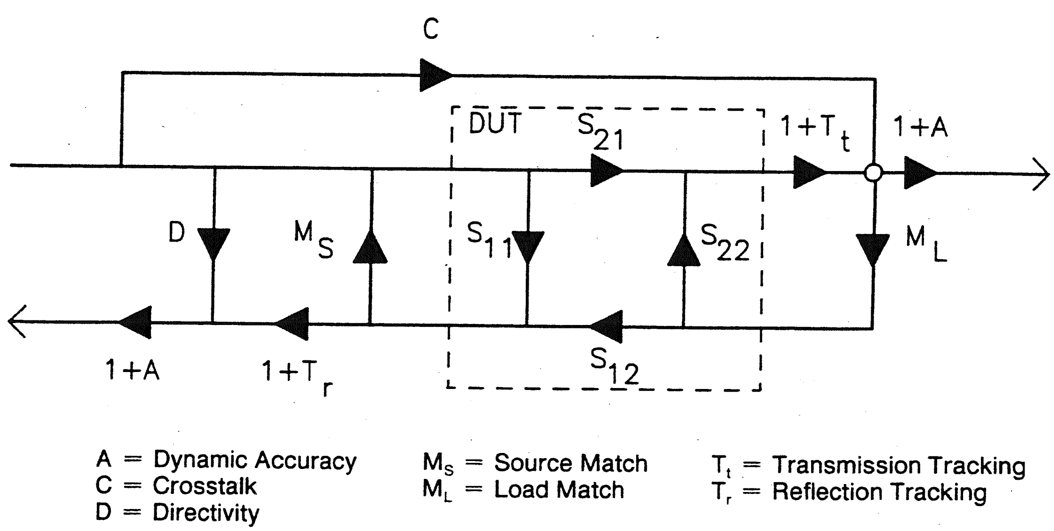


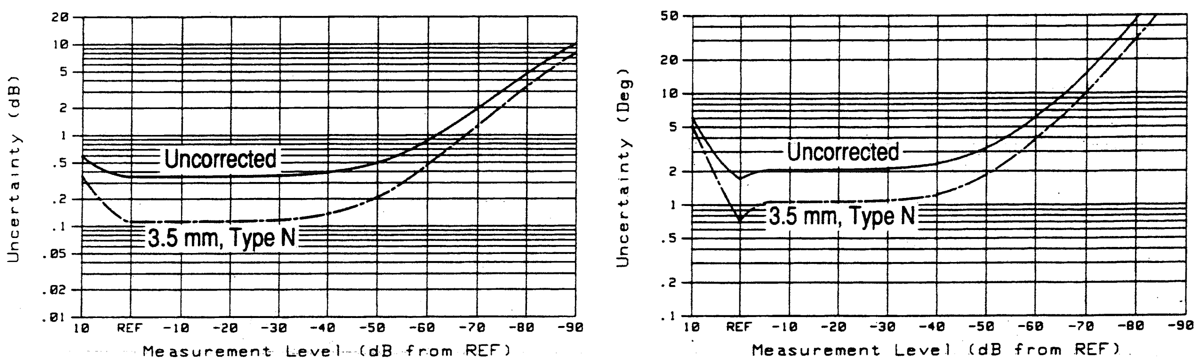
Figure 2-3. System Error Model

TYPICAL MEASUREMENT UNCERTAINTY FOR HP 8752A

The graphs[1] below show the typical measurement uncertainty for the analyzer using type-N or 3.5 mm connectors, with and without error correction. Two graphs are provided for transmission measurements (a magnitude graph and a phase graph), and two for reflection measurements. The graphs on the next page apply to the HP 8752A option 003.

Corrected performance in the transmission measurement graphs shows the improvement obtained from a response and isolation calibration. Corrected performance in the reflection measurement graphs shows the improvement obtained from a reflection 1-port calibration.

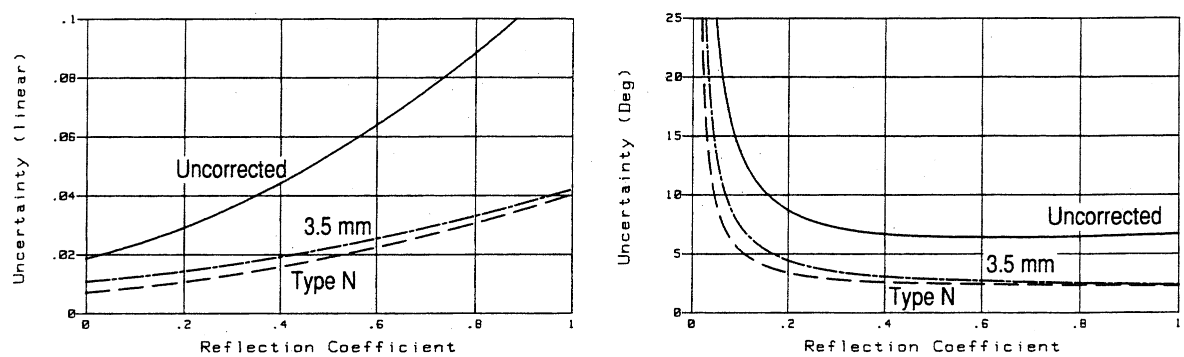
Transmission measurements[2]



Magnitude

Phase

Reflection measurements[3]



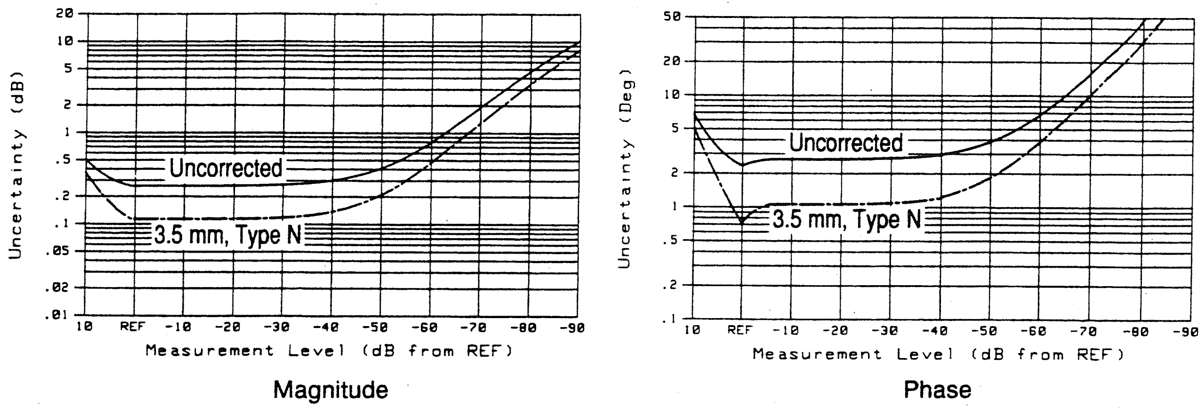
Magnitude

Phase

Figure 2-4. Typical Measurement Uncertainty for Standard HP 8752A (300 kHz to 1.3 GHz)

1. These measurement uncertainty curves utilize an RSS model for the contributions of random errors such as noise, and typical connector repeatabilities; and a worst-case model for the contributions of dynamic accuracy and residual systematic errors.
2. The graphs shown for transmission measurements assume a well-matched device ($S_{11} = S_{22} = 0$).
3. The graphs for reflection measurement uncertainty apply to either a one-port device or a two-port device with more than 6 dB of insertion loss.

Transmission measurements[1]



Reflection measurements[2]

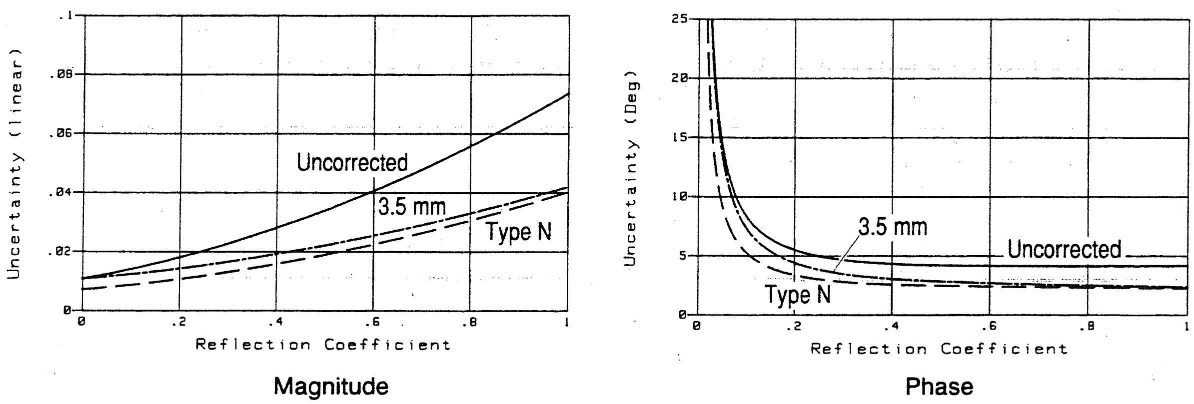


Figure 2-5. Typical Measurement Uncertainty for HP 8752A Option 003 (300 kHz to 3 GHz)

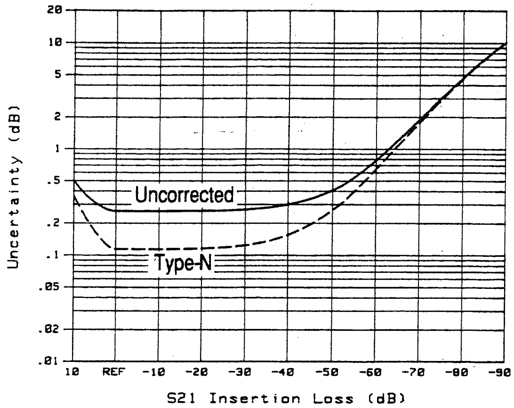
1. The graphs shown for transmission measurements assume a well-matched device ($S_{11} = S_{22} = 0$).
2. The graphs for reflection measurement uncertainty apply to either a one-port device or a two-port device with more than 6 dB of insertion loss.

TYPICAL MEASUREMENT UNCERTAINTY FOR HP 8752B

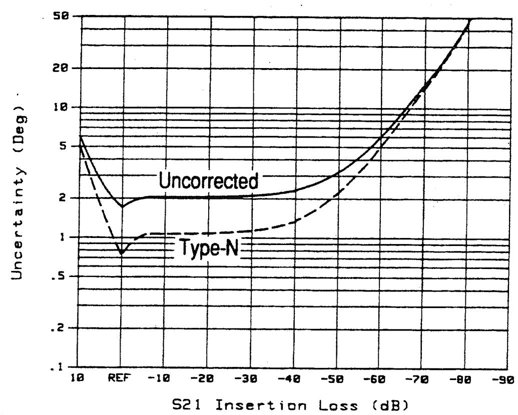
The graphs[1] below show the typical measurement uncertainty for the analyzer using type-N connectors, with and without error correction. Two graphs are provided for transmission measurements (a magnitude graph and a phase graph), and two for reflection measurements. The graphs on the next page apply to the HP 8752B option 003.

Corrected performance in the transmission measurement graphs shows the improvement obtained from a response and isolation calibration. Corrected performance in the reflection measurement graphs shows the improvement obtained from a reflection 1-port calibration.

Transmission measurements[2]

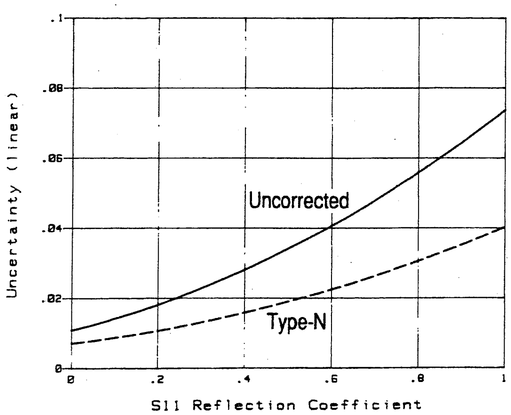


Magnitude

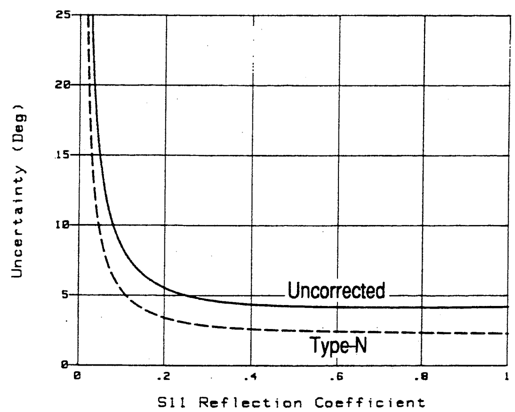


Phase

Reflection measurements[3]



Magnitude

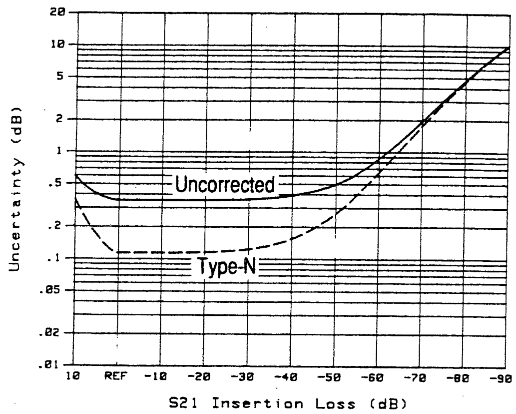


Phase

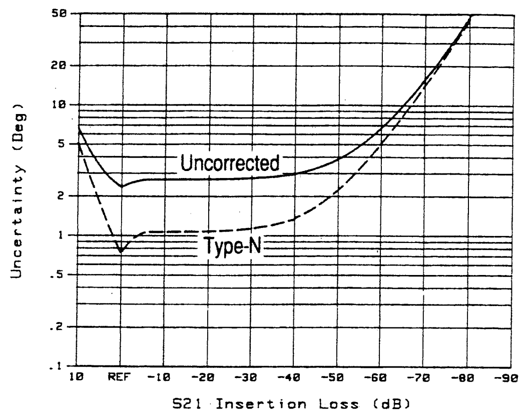
Figure 2-6. Typical Measurement Uncertainty for Standard HP 8752B (300 kHz to 1.3 GHz)

1. These measurement uncertainty curves utilize an RSS model for the contributions of random errors such as noise, and typical connector repeatabilities; and a worst-case model for the contributions of dynamic accuracy and residual systematic errors.
2. The graphs shown for transmission measurements assume a well-matched device ($S_{11} = S_{22} = 0$).
3. The graphs for reflection measurement uncertainty apply to either a one-port device or a two-port device with more than 6 dB of insertion loss

Transmission measurements[1]

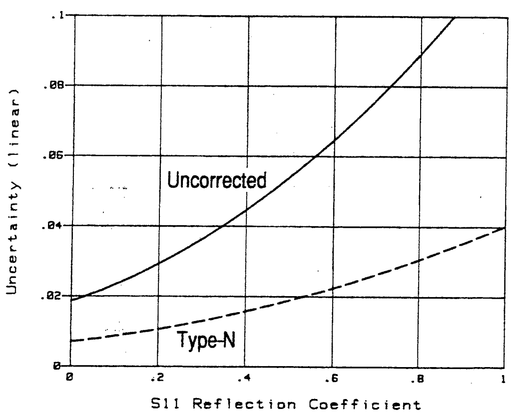


Magnitude

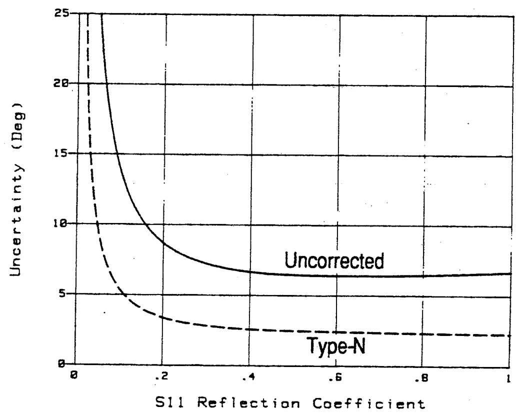


Phase

Reflection measurements[2]



Magnitude



Phase

Figure 2–14. Typical Measurement Uncertainty for HP 8752B Option 003 (300 kHz to 3 GHz)

1. The graphs shown for transmission measurements assume a well-matched device ($S_{11}=S_{22}=0$).
2. The graphs for reflection measurement uncertainty apply to either a one-port device or a two-port device with more than 6 dB of insertion loss.

DETERMINING EXPECTED SYSTEM PERFORMANCE

The uncertainty equations and tables of system specifications provided in this chapter can be used to calculate the expected system performance of the analyzer. The following pages explain how to determine the residual errors of a particular system and combine them to obtain total error-corrected residual uncertainty values, using worksheets (Tables 2-4 and 2-5).

Separate tables are used to determine residual magnitude and phase uncertainties in the following measurement types:

- Table 2-4: Reflection measurements
- Table 2-5: Transmission measurements
- Completed examples of the residual uncertainty tables are provided after Table 2-5.

NOTE: A spreadsheet program can automate the uncertainty worksheets and eliminate mathematical errors.

Determining Crosstalk

The crosstalk error value is required in the transmission uncertainty worksheet. You can use the value in Table 2-3 or measure it as explained below:

Connect impedance-matched loads to the reflection and transmission ports. Set IF bandwidth to 10 Hz by pressing **[AVG]** ***[IF BW]*** **[10] [x1]**. Turn on an averaging factor of 5 by pressing ***[AVERAGING FACTOR]*** **[5] [x1]** ***[AVERAGING ON]***. Averaging reduces the analyzer's noise floor as explained earlier in this section.

Select a transmission measurement, turn on the marker statistics function (see "Using Markers"), and measure the mean value of the trace. Use the mean value plus one standard deviation as the residual crosstalk value of your system.

Table 2-4. Reflection Measurement Uncertainty Worksheet

PART A — Analyzer Performance **Frequency: ________**

In the columns below, enter the values for each term. Values are obtained from Table 2-3 and Figure 2-1.

Error Term	Symbol	dB Value	Linear Value
Directivity	D		________
Reflection tracking	T_r		________
Source match (reflection)	M_s		________
Load match	M_l		________
Dynamic accuracy			
Magnitude[1]	A_m	+ ________	________
Phase	A_p		________
Trace noise	N_h		________
Connector repeatability	R_c		________

PART B — DUT Performance

In the columns below, enter the calculated or measured performance of the DUT.

Performance Parameter	dB Value	Linear Value ($10^{dB\ Value/20}$)
S11	________	________
S21	________	________
S12	________	________

PART C — Total Magnitude Errors (Systematic and Random)

Systematic Errors

In the spaces provided, enter the required linear values from Parts A and B. Then combine these errors to obtain the total sum of systematic errors.

D				= ________	[k]
T_r x S11		________	________	= ________	[l]
M_s x S11 x S11	________	x ________	x ________	= ________	[m]
M_l x S21 x S12	________	x ________	x ________	= ________	[n]
A_m x S11		________	________	= ________	[o]
Total Systematic Errors: k +l +m +n +o				= ________	[S]

Random Errors

Enter the required linear values from Parts A and B. Combine these errors in a root sum of the squares (RSS) fashion to obtain a total sum of random errors.

3 x N_h x S11	3x ________ x ________	= ________	[x]
R_c x (1 + 2 S11 + S11^2)	________ x (1 + 2 x ________ + ________^{2})	= ________	[y]
R_c x S21 x S12	________ x ________ x ________	= ________	[z]

Total Random Errors

$\sqrt{x^2 + y^2 + z^2}$ $\sqrt{\_\_\_\_ + \_\_\_\_ + \_\_\_\_}$ = ________ [R]

TOTAL MAGNITUDE ERRORS:

E_{rm}(linear) = S + R ________ + ________ = ________ [E_{rm}]

E_{rm}(log) = 20 Log (1 + E_{rm}/S11) 20 Log (1 + ________ / ________) = + ________ dB

20 Log (1 − E_{rm}/S11) 20 Log (1 − ________ / ________) = − ________ dB

PART D — Total Phase Errors

E_{rp} = Arcsin[(E_{rm} − (A_m x S11))/S11] + A_p

Arcsin[(________ − (______ x ______)) / ______] + ______ = ± ______ °

1. Enter the value from Figure 2-1 as a positive dB value. Convert to linear using the formula: Linear Value = $10^{dB\ value/20} - 1$.

Table 2-5. Transmission Measurement Uncertainty Worksheet

PART A — Analyzer Performance:

In the columns below, enter the values for each term. Values are obtained from Table 2-3 and Figure 2-1.

Error Term	Symbol	dB Value	Linear Value
Crosstalk[1]	C		______
Transmission tracking	T_t		______
Source match (transmission)	M_s		______
Load match	M_l		______
Dynamic accuracy			
Magnitude[2]	A_m	+ ______	______
Phase	A_p		______
Trace noise	N_h		______
Connector repeatability	R_c		______
Cable Reflection Magnitude			
Stability	S_r		______
Cable Transmission Phase			
Stability (Degrees)	S_t		______

PART B — DUT Performance

In the columns below, enter the calculated or measured performance of the DUT.

Performance Parameter	dB Value	Linear Value ($10^{dB\ Value/20}$)
S11	______	______
S21	______	______
S12	______	______
S22	______	______

PART C — Total Magnitude Errors (Systematic and Random)

Systematic Errors

In the spaces provided, enter the required linear values from Parts A and B. Then combine these errors to obtain the total sum of systematic errors.

C = ______ [k]

T_t x S21 ______ x ______ = ______ [l]

M_s x S11 x S21 ______ x ______ x ______ = ______ [m]

(S_r + M_l) x S21 x S22 (______ + ______) x ______ x ______ = ______ [n]

A_m x S21 ______ x ______ = ______ [o]

Total Systematic Errors: k +l +m +n +o = ______ [S]

Random Errors

Enter the required linear values from Parts A and B. Combine these errors in a root sum of the squares (RSS) fashion to obtain a total sum of random errors.

3 x N_h x S21 3x ______ x ______ = ______ [x]

R_c x S21 x (1 + S11) ______ x ______ x (1 + ______) = ______ [y]

R_c x S21 x (1 + S22) ______ x ______ x (1 + ______) = ______ [z]

Total Random Errors

$\sqrt{x^2 + y^2 + z^2}$ $\sqrt{\_\_\_\_ + \_\_\_\_ + \_\_\_\_}$ = ______ [R]

TOTAL MAGNITUDE ERRORS:

E_{tm}(linear) = S + R ______ + ______ = ______ E_{tm}

E_{tm}(log) = 20 Log (1 + E_{tm}/S21) 20 Log (1 + ______ / ______) = + ______ dB

20 Log (1 − E_{tm}/S21) 20 Log (1 − ______ / ______) = − ______ dB

PART D — Total Phase Errors

E_{tp} = Arcsin[(E_{tm} − (A_m x S21))/S21] S_t + A_p

Arcsin[(______ − (______ x ______)) / ______] + ______ + ______ = ± ______ °

1. Use the value listed in Table 2-3 or measure (using the instructions provided in "Determining Crosstalk," earlier in this chapter).
1. Enter the value from Figure 2-1 as a positive dB value. Convert to linear using the formula: Linear Value = $10^{dB\ value/20} - 1$.

EXAMPLE

Table 2–4. Reflection Measurement Uncertainty Worksheet

PART A — Analyzer Performance **Frequency:** 1.0 GHz

In the columns below, enter the values for each term. Values are obtained from Table 2-3 and Figure 2-1.

Error Term	Symbol	dB Value	Linear Value
Directivity	D		.01
Reflection tracking	T_r		.0233
Source match (reflection)	M_s		.0316
Load match	M_l		.0718
Dynamic accuracy			
Magnitude[1]	A_m	+ .05	.0058
Phase	A_p		.3
Trace noise	N_h		.0007
Connector repeatability	R_c		.0006

PART B — DUT Performance

In the columns below, enter the calculated or measured performance of the DUT.

Performance Parameter	dB Value	Linear Value ($10^{\text{dB Value}/20}$)
S11	−22	.08
S21	−20	.1
S12	−20	.1

PART C — Total Magnitude Errors (Systematic and Random)

Systematic Errors

In the spaces provided, enter the required linear values from Parts A and B. Then combine these errors to obtain the total sum of systematic errors.

D				= .01	[k]
T_r x S11		.0233	x .08	= .001864	[l]
M_s x S11 x S11	.0316	x .08	x .08	= .000202	[m]
M_l x S21 x S12	.0708	x .1	x .1	= .000708	[n]
A_m x S11		.0058	x .08	= .000464	[o]
Total Systematic Errors: k +l +m +n +o				= .013238	[S]

Random Errors

Enter the required linear values from Parts A and B. Combine these errors in a root sum of the squares (RSS) fashion to obtain a total sum of random errors.

3 x N_h x S11   3x .007 x .08 = .00168 [x]

R_c x (1 + 2 S11 + S11^2)   .0006 x (1 + 2 x .08 + .08^2) = .0007 [y]

R_c x S21 x S12   .0006 x .1 x .1 = .000006 [z]

Total Random Errors

$\sqrt{x^2 + y^2 + z^2}$ $\sqrt{2.822\times10^{-6} + 4.9\times10^{-7} + 3.6\times10^{-11}}$ = .00182 [R]

TOTAL MAGNITUDE ERRORS:

E_{rm}(linear) = S + R   .01328 + .00182 = .01506 [E_{rm}]

E_{rm}(log) = 20 Log (1 + E_{rm}/S11)   20 Log (1 + .01506 / .08) = + 1.50 dB

20 Log (1 − E_{rm}/S11)   20 Log (1 − .01506 / .08) = − 1.81 dB

PART D — Total Phase Errors

E_{rp} = Arcsin[(E_{rm} − (A_m x S11))/S11] + A_p

Arcsin[(.01506 − (.0058 x .08)) / .08] + .3 = ± 10.8 °

1. Enter the value from Figure 2-1 as a positive dB value. Convert to linear using the formula: Linear Value = $10^{\text{dB value}/20} - 1$.

EXAMPLE

Table 2-5. Transmission Measurement Uncertainty Worksheet

PART A — Analyzer Performance: Frequency 1.0 GHz

In the columns below, enter the values for each term. Values are obtained from Table 2-3 and Figure 2-1.

Error Term	Symbol	dB Value	Linear Value
Crosstalk[1]	C		.00001
Transmission tracking	T_t		.0233
Source match (transmission)	M_s		.0708
Load match	M_l		.0708
Dynamic accuracy			
Magnitude[2]	A_m	+ .05	.0058
Phase	A_p		.3
Trace noise	N_h		.0007
Connector repeatability	R_c		.0006
Cable Reflection Magnitude			
Stability	S_r		.001
Cable Transmission Phase			
Stability (Degrees)	S_t		.05

PART B — DUT Performance

In the columns below, enter the calculated or measured performance of the DUT.

Performance Parameter	dB Value	Linear Value ($10^{\text{dB Value}/20}$)
S11	−22	.08
S21	−20	.1
S12	−20	.1
S22	−23	.071

PART C — Total Magnitude Errors (Systematic and Random)

Systematic Errors

In the spaces provided, enter the required linear values from Parts A and B. Then combine these errors to obtain the total sum of systematic errors.

C = .00001 [k]

T_t x S21 .0233 x .1 = .00233 [l]

M_s x S11 x S21 .0708 x .08 x .1 = .000566 [m]

(S_r + M_l) x S21 x S22 (.001 + .0708) x .1 x .071 = .00051 [n]

A_m x S21 .0058 x .1 = .00058 [o]

Total Systematic Errors: k + l + m + n + o = .004 [S]

Random Errors

Enter the required linear values from Parts A and B. Combine these errors in a root sum of the squares (RSS) fashion to obtain a total sum of random errors.

3 x N_h x S21 3x .0007 x .1 = .00021 [x]

R_c x S21 x (1 + S11) .0006 x .1 x (1 + .08) = .000065 [y]

R_c x S21 x (1 + S22) .0006 x .1 x (1 + .071) = .000064 [z]

Total Random Errors

$\sqrt{x^2 + y^2 + z^2}$ $\sqrt{4.41 \times 10^{-8} + 4.2 \times 10^{-9} + 4.129 \times 10^{-9}}$ = .000229 [R]

TOTAL MAGNITUDE ERRORS:

E_{tm}(linear) = S + R .004 + .00029 = .004229 E_{tm}

E_{tm}(log) = 20 Log (1 + E_{tm}/S21) 20 Log (1 + .004229 / .1) = + .36 dB

20 Log (1 − E_{tm}/S21) 20 Log (1 − .004229 / .1) = − .38 dB

PART D — Total Phase Errors

E_{tp} = Arcsin[(E_{tm} − (A_m x S21))/S21] S_t + A_p

Arcsin[(.004229 − (.0058 x .1)) / .1] + .05 + .3 = ± 2.44°

1. Use the value listed in Table 2-3 or measure (using the instructions provided in "Determining Crosstalk," earlier in this chapter).

1. Enter the value from Figure 2-1 as a positive dB value. Convert to linear using the formula: Linear Value = $10^{\text{dB value}/20} - 1$.

REFLECTION UNCERTAINTY EQUATIONS

This page shows how E_{rm} is derived from analysis of the system error model shown in Figure 2-3.

Total Reflection Magnitude Uncertainty (E_{rm})

An analysis of the error model (Figure 2-3) yields an equation for the reflection magnitude uncertainty. The equation contains all of the first order terms and the significant second order terms. The three terms under the radical are random and are combined on a root sum of the squares (RSS) basis. The terms in the systematic error group are combined on a worst case basis. In all cases, the error terms and the S-parameters are treated as linear absolute magnitudes.

$E_{rm}(\log) = 20\log(1 \pm E_{rm}/S11)$

where

$E_{rm} = S_r + \sqrt{X_r^2 + Y_r^2 + Z_r^2}$

S_r = systematic error = $D + T_r \times S11 + M_s \times S11^2 + M_l \times S21 \times S12 + A_m \times S11$

X_r = random trace noise = $3 \times N_h \times S11$

Y_r = random port 1 repeatability = $R_c \times (1 + 2\,S11 + S11^2)$

Z_r = random port 2 repeatability = $R_c \times S21 \times S12$

Total Reflection Phase Uncertainty (E_{rp})

Reflection phase uncertainty is determined from a comparison of the magnitude uncertainty with the test signal magnitude. The worst case phase angle is computed. The result is combined with phase dynamic accuracy.

$E_{rp} = \arcsin((E_{rm} - A_m \times S11)/S11) + A_p$

TRANSMISSION UNCERTAINTY EQUATIONS

This page shows how E_{tm} is derived from analysis of the system error model shown in Figure 2-3.

Total Transmission Magnitude Uncertainty (E_{tm})

An analysis of the error model in Figure 2-3 yields an equation for the transmission magnitude uncertainty. The equation contains all of the first order terms and some of the significant second order terms. The three terms under the radical are random and are combined on an RSS basis. The terms in the systematic error group are combined on a worst case basis. In all cases, the error terms are treated as linear absolute magnitudes.

E_{tm} (log) = 20 log (1 ± E_{tm} / S21)

where

$$E_{tm} = S_t + \sqrt{Xt^2 + Yt^2 + Zt^2}$$

S_t = systematic error = $C + T_t \times S21 + M_s \times S11 \times S21 + (M_l + S_r) \times S21 \times S22 + A_m \times S21$

X_t = random high-level noise = $3 \times N_h \times S21$

Y_t = random port 1 repeatability = $R_c \times S21 + R_c \times S11 \times S21$

Z_t = random port 2 repeatability = $R_c \times S21 + R_c \times S22 \times S21$

Total Transmission Phase Uncertainty (E_{tp})

Transmission phase uncertainty is calculated from a comparison of the magnitude uncertainty with the test signal magnitude. The worst case phase angle is computed. This result is combined with the error terms related to phase dynamic accuracy, cable phase stability, and thermal drift of the total system.

E_{tp} = arcsin $((E_{tm} - A_m \times S21) / S21) + S_t + A_p$

Chapter 3. HP-IB Remote Programming

CHAPTER CONTENTS

INTRODUCTION

The analyzer is factory-equipped with a remote programming digital interface using the Hewlett-Packard Interface Bus (HP-IB). (HP-IB is Hewlett-Packard's hardware, software, documentation, and support for IEEE 488.1 and IEC-625, worldwide standards for interfacing instruments.) This allows the analyzer to be controlled by an external computer that sends commands or instructions to and receives data from the analyzer using the HP-IB. In this way, a remote operator has the same control of the instrument available to a local operator from the front panel, except for control of the line power switch.

In addition, the analyzer itself can use HP-IB to directly control compatible peripherals, without the use of an external controller. It can output measurement results directly to a compatible printer or plotter, or store instrument states to a compatible disk drive.

This chapter provides an overview of HP-IB operation. Chapter 10 provides information on different controller modes, and on setting up the analyzer as a controller of peripherals. It also explains how to use the analyzer as a controller to print, plot, and store to an external disk. HP-IB equivalent mnemonics for front panel functions are provided in parentheses throughout this manual.

More complete information on programming the analyzer remotely over HP-IB is provided in the following documents:

- *HP-IB Programming Guide for the HP 8752A and HP 8753C Using the HP 9000 Series 200/300 Desktop Computer (BASIC)*. This is a tutorial introduction to remote operation of the analyzer using an HP 9000 series 200 or 300 computer. It includes examples of remote measurements using BASIC programming. These examples are also stored on the example programs disk provided with the analyzer. The *HP-IB Programming Guide* assumes familiarity with front panel operation of the instrument.

- *HP-IB Quick Reference for the HP 8700-Series Analyzers*. This is a complete reference summary for remote operation of the analyzer with a controller. It includes both functional and alphabetical lists of all analyzer HP-IB commands. This guide is intended for use by those familiar with HP-IB programming and the basic functions of the analyzer.

A complete general description of the HP-IB is available in *Tutorial Description of the Hewlett-Packard Interface Bus*, HP publication 5952-0156. For more information on the IEEE-488.1 standard refer to *IEEE Standard Digital Interface for Programmable Instrumentation*, published by the Institute of Electrical and Electronics Engineers, Inc., 345 East 47th Street, New York, New York 10017.

HOW HP-IB WORKS

The HP-IB uses a party-line bus structure in which up to 15 devices can be connected on one contiguous bus. The interface consists of 16 signal lines and 8 ground lines in a shielded cable. With this cabling system, many different types of devices including instruments, computers, plotters, printers, and disk drives can be connected in parallel.

Every HP-IB device must be capable of performing one or more of the following interface functions:

Talker

A talker is a device capable of sending device-dependent data when addressed to talk. There can be only one talker at any given time. Examples of this type of device are voltmeters, counters, and tape readers. The analyzer is a talker when it sends trace data or marker information over the bus.

Listener

A listener is a device capable of receiving device-dependent data when addressed to listen. There can be any number of listeners at any given time. Examples of this type of device are printers, power supplies, and signal generators. The analyzer is a listener when it is controlled over the bus by a computer.

Controller

A controller is a device capable of managing the operation of the bus and addressing talkers and listeners. There can be only one active controller at any time. Examples of controllers include desktop computers and minicomputers. In a multiple-controller system, active control can be passed between controllers, but there can only be one *system controller*, which acts as the master, and can regain active control at any time. The analyzer is an active controller when it plots, prints, or stores to an external disk drive in the pass control mode. The analyzer is a system controller when it is in the system controller mode. These modes are discussed in more detail in Chapter 10 under *HP-IB Menu*.

HP-IB BUS STRUCTURE

Data Bus

The data bus consists of eight bidirectional lines that are used to transfer data from one device to another. Programming commands and data are typically encoded on these lines in ASCII, although binary encoding is often used to speed up the transfer of large arrays. Both ASCII and binary data formats are available to the analyzer. In addition, every byte transferred over HP-IB undergoes a *handshake* to ensure valid data.

Handshake Lines

A three-line handshake scheme coordinates the transfer of data between talkers and listeners. This technique forces data transfers to occur at the speed of the slowest device, and ensures data integrity in multiple listener transfers. With most computing controllers and instruments, the handshake is performed automatically, which makes it transparent to the programmer.

Control Lines

The data bus also has five control lines that the controller uses both to send bus commands and to address devices.

IFC. Interface Clear. Only the system controller uses this line. When this line is true (low), all devices (addressed or not) unaddress and go to an idle state.

ATN. Attention. The active controller uses this line to define whether the information on the data bus is a *command* or is *data*. When this line is true (low), the bus is in the command mode and the data lines carry bus commands. When this line is false (high), the bus is in the data mode and the data lines carry device-dependent instructions or data.

SRQ. Service Request. This line is set true (low) when a device requests service: the active controller services the requesting device. The analyzer can be enabled to pull the SRQ line for a variety of reasons.

REN. Remote Enable. Only the system controller uses this line. When this line is set true (low), the bus is in the remote mode, and devices are addressed either to listen or to talk. When the bus is in remote and a device is addressed, it receives instructions from HP-IB rather than from its front panel (the **[LOCAL]** key returns the device to front panel operation). When this line is set false (high), the bus and all devices return to local operation.

EOI. End or Identify. This line is used by a talker to indicate the last data byte in a multiple byte transmission, or by an active controller to initiate a parallel poll sequence. The analyzer recognizes the EOI line as a terminator, and it pulls the EOI line with the last byte of a message output (data, markers, plots, prints, error messages). The analyzer does not respond to parallel poll.

Figure 3-1 illustrates the structure of the HP-IB bus lines.

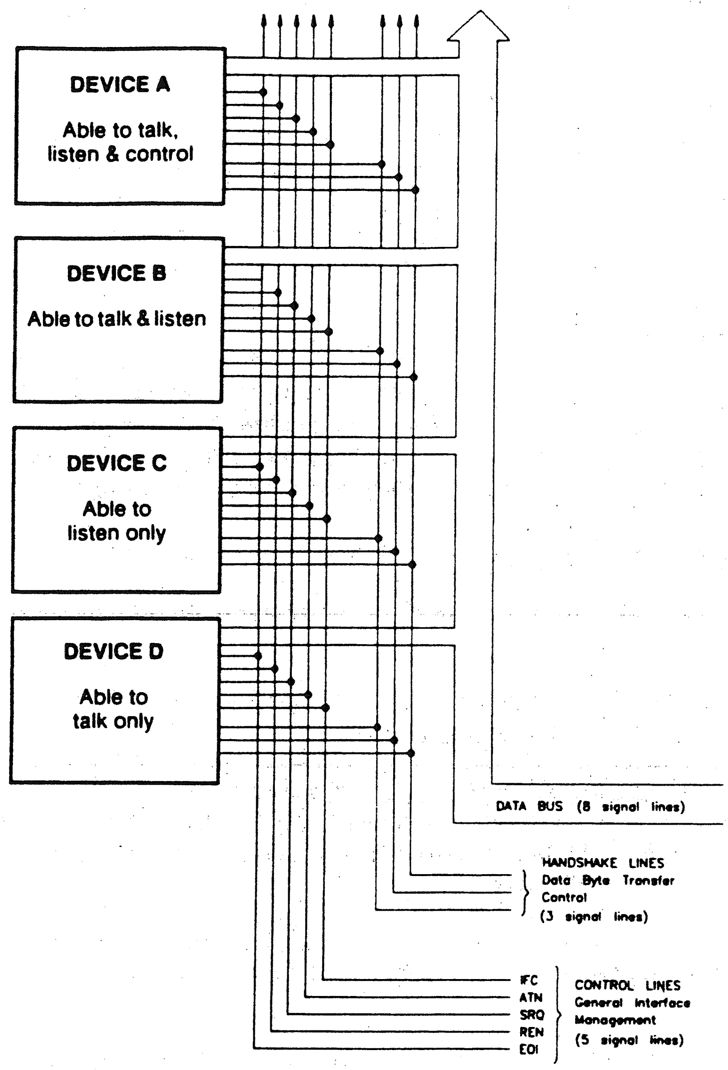


Figure 3-1. HP-IB Structure

HP-IB REQUIREMENTS

Number of Interconnected Devices:	15 maximum.
Interconnection Path/ Maximum Cable Length:	20 meters maximum or 2 meters per device, whichever is less.
Message Transfer Scheme:	Byte serial/ bit parallel asynchronous data transfer using a 3-line handshake system.
Data Rate:	Maximum of 1 megabyte per second over limited distances with tri-state drivers. Actual data rate depends on the transfer rate of the slowest device involved.
Address Capability:	Primary addresses: 31 talk, 31 listen. A maximum of 1 talker and 14 listeners at one time.
Multiple Controller Capability:	In systems with more than one controller (like the analyzer system), only one can be active at a time. The active controller can pass control to another controller, but only the system controller can assume unconditional control. Only one system controller is allowed. The system controller is hard-wired to assume bus control after a power failure.

ANALYZER HP-IB CAPABILITIES

As defined by the IEEE 488.1 standard, the analyzer has the following capabilities:

SH1	Full source handshake.
AH1	Full acceptor handshake.
T6	Basic talker, answers serial poll, unaddresses if MLA is issued. No talk-only mode.
L4	Basic listener, unaddresses if MTA is issued. No listen-only mode.
SR1	Complete service request (SRQ) capabilities.
RL1	Complete remote/local capability including local lockout.
PP0	Does not respond to parallel poll.
DC1	Complete device clear.
DT1	Responds to a group execute trigger in the hold trigger mode.
C1,C2,C3	System controller capabilities in system controller mode.
C10	Pass control capabilities in pass control mode.
E2	Tri-state drivers.

BUS MODE

The analyzer uses a single-bus architecture. The single bus allows both the analyzer and the host controller to have complete access to the peripherals in the system.

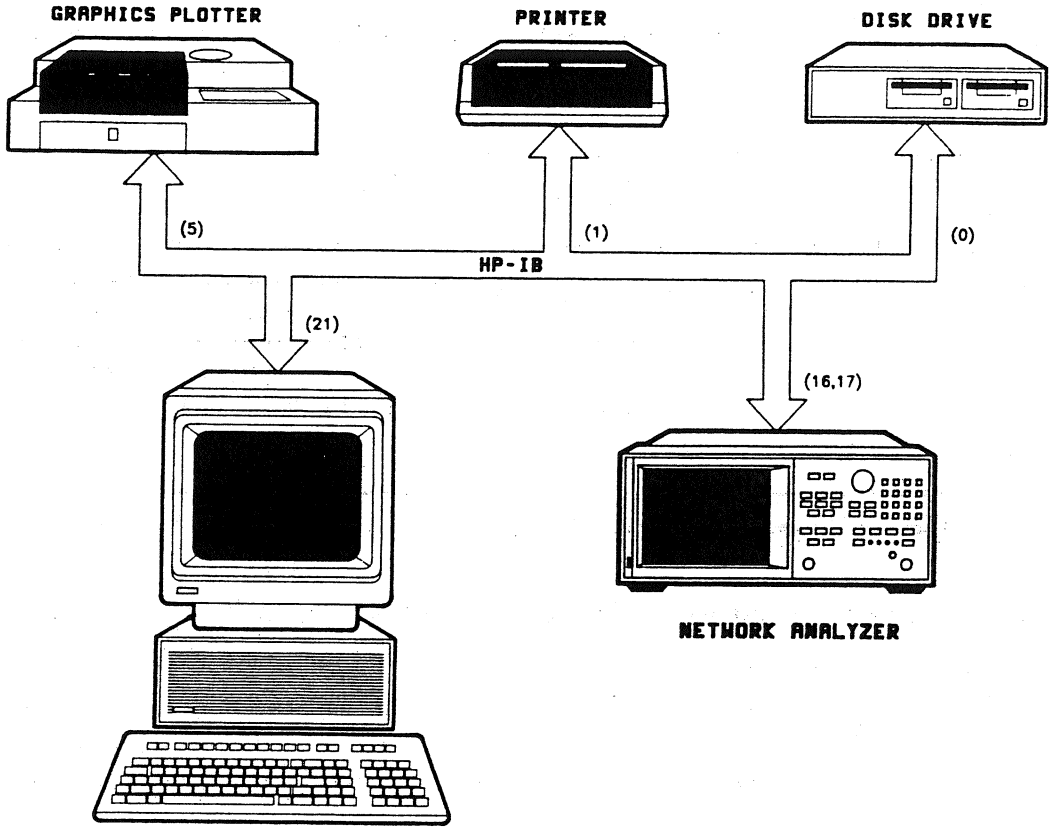


Figure 3-2. Analyzer Single Bus Concept

Three different controller modes are possible, system controller, talker/listener, and pass control.

System Controller. This mode allows the analyzer to control peripherals directly in a stand-alone environment (without an external controller). This mode can only be selected manually from the analyzer front panel. Use this mode for operation when no computer is connected to the analyzer. Do not use this mode for programming.

Talker/Listener. This is the traditional programming mode, in which the computer is involved in all peripheral access operations. Peripheral access (plotting and printing only) is also possible by addressing the analyzer to talk, addressing the peripheral to listen, and placing the HP-IB in the data mode.

Pass Control. This mode allows you to control the analyzer over HP-IB as with the talker/listener mode, and also allows the analyzer to take or pass control in order to plot, print, and access a disk. During the peripheral operation, the host computer is free to perform other internal tasks such as data or display manipulation (the bus is tied up by the analyzer during this time). After a task is completed, the host controller accepts control again when the analyzer returns it.

In general, use the talker/listener mode for programming the analyzer unless you desire direct peripheral access. Preset does not affect the selected bus mode, but the bus mode returns talker/listener if power is cycled.

Chapter 10 explains the three different bus modes in detail, and provides information on setting the correct bus mode. Programming information for talker/listener mode and pass control mode is provided in the *HP-IB Programming Guide.*

SETTING ADDRESSES

In communications through HP-IB, each instrument on the bus is identified by an HP-IB address. This address code must be different for each instrument on the bus. Refer to *Address Menu* in Chapter 10 for information on default addresses, and on setting and changing addresses. These addresses are not affected when you press **[PRESET]** or cycle the power (although the **[PRESET]** key must be pressed to implement a change to the analyzer address).

VALID CHARACTERS

The analyzer accepts ASCII letters, numbers, decimal points, +/−, semicolons, quotation marks (''), carriage returns (CR), and linefeeds (LF). Both upper and lower case are acceptable. Leading zeros, spaces, carriage returns, and unnecessary terminators are ignored, except those within a command or appendage. Carriage returns are ignored. An invalid character causes a syntax error. Syntax errors are described in more detail under in the *HP-IB Programming Guide*.

CODE NAMING CONVENTION

The analyzer HP-IB commands are derived from their front panel key titles (where possible), according to the naming convention below.

Convention	Key Title	For HP-IB Code Use	Example
One Word	Power Start	First Four Letters	POWE STAR
Two Words	Electrical Delay Search Right	First Three Letters of First Word First Letter of Second Word	ELED SEAR
Two Words in a Group	Marker →Center Gate →Span	First Four Letters of Both	MARKCENT GATESPAN
Three Words	Cal Kit N 50Ω Pen Num Data	First Three Letters of First Word First Letter of Second Word First Four Letters of Third Word	CALKN50 PENNDATA

Some codes require appendages (on, off, 1, 2, etc.). Codes that have no front panel equivalent are HP-IB only commands, and use a similar convention based on the common name of the function. Where possible, analyzer codes are compatible with HP 8510A/B codes.

Front panel equivalent codes and HP-IB only codes are summarized in the *HP-IB Quick Reference*.

UNITS AND TERMINATORS

The analyzer outputs data in basic units and assumes these basic units when it receives an input, unless the input is otherwise qualified. The basic units and allowable expressions follow; either upper or lower case is acceptable.

Basic Units	Allowable Expressions
Seconds	S
Milliseconds	MS
Microseconds	US
Nanoseconds	NS
Picoseconds	PS
Femtoseconds	FS
Hertz	HZ
Kilohertz	KHZ
Megahertz	MHZ
Gigahertz	GHZ
dB or dBm	DB
Volts	V

Terminators are used to indicate the end of a command to allow the analyzer to recover to the next command in the event of a syntax error. The semicolon is the recommended command terminator. The line feed (LF) character and the HP-IB EOI line can also be used as terminators. The analyzer ignores the carriage return (CR) character.

HP-IB DEBUG MODE

An HP-IB diagnostic feature (debug mode) is available in the HP-IB menu. Activating the debug mode causes the analyzer to scroll incoming HP-IB commands across the display. Nonprintable characters are represented with a π. Any time the analyzer receives a syntax error, the commands halt, and a pointer $\wedge$ indicates the misunderstood character. The *HP-IB Programming Guide* explains how to clear a syntax error.

DISPLAY GRAPHICS

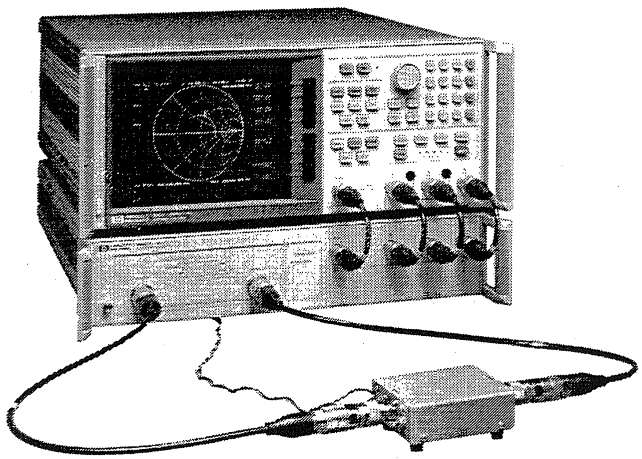
The analyzer display can be used as a graphics display for displaying connection diagrams or custom instructions to an operator. The display accepts a subset of Hewlett-Packard Graphics Language (HP-GL) commands issued by an external computer. Some user graphics can be created using the test sequencing feature.

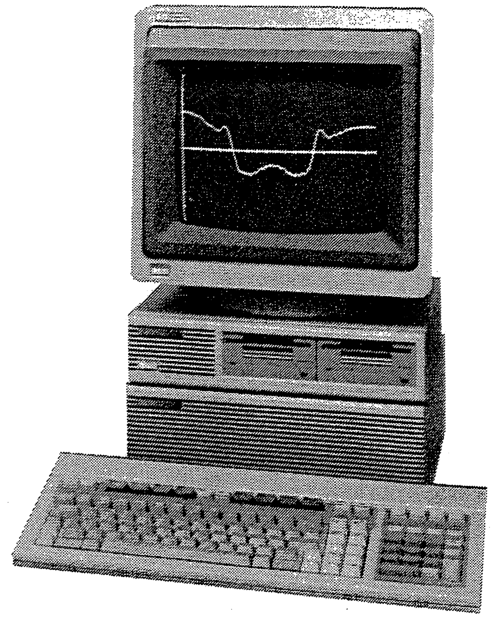
NOTE: The analyzer display occupies an additional address on the HP-IB. Determine the display bus address by adding 1 to the analyzer address (if the analyzer address is an even number), or subtracting 1 (if it is an odd number). Thus the factory default display address for graphics is 17.

HP-IB Programming Guide

HEWLETT PACKARD

For the HP 8752A and HP 8753C Network Analyzers with the HP 9000 series 200/300 desktop computer (BASIC)





Introduction

This programming guide is an introduction to remote operation of the HP 8752A and 8753C Network Analyzers using an HP 9000 series 200 or 300 computer. It is a tutorial introduction, using BASIC programming examples. The examples are on the Example Programs disk (part number 08753-10014), included with the operating manual. This document is closely associated with the `HP-IB Quick Reference` for the HP 8700-series network analyzers. The *HP-IB Quick Reference* provides complete programming information in a concise format. Included in the *HP-IB Quick Reference* are both functional and alphabetical lists of HP-IB commands. The *Quick Reference* also lists HP-IB commands, along with its softkey menu explanations.

The Hewlett-Packard computers specifically addressed are the HP 9000 series 200 and 300 computers, operating with BASIC 2.0 with AP2_1, or BASIC 3.0 or higher. This includes the 216 (9816), 217 (9817), 220 (9920), 226 (9826), 236 (9836), 310 and 320 computers.

The reader should become familiar with the operation of the network analyzer before controlling it over HP-IB. This document is not intended to teach BASIC programming or to discuss HP-IB theory except at an introductory level: read "For more information," next, for documents better suited to these tasks.

For more information

For more information concerning the operation of the network analyzer, refer to the following:

User's Guide
Quick Reference
Operating Manual

For more information concerning BASIC, see the manual set for the BASIC revision being used. For example:

BASIC 5.0 Programming Techniques	98613-90012
BASIC 5.0 Language Reference	98613-90052

For more information concerning HP-IB, see:

BASIC 5.0 Interfacing Techniques	98613-90022
Tutorial Description of the Hewlett-Packard Interface Bus	5952-0156
Condensed Description of the Hewlett-Packard Interface Bus	59401-90030

Table of contents

Required equipment

To run the examples of this *Programming Guide*, the following equipment is required:

1. HP 8752A or 8753C Network Analyzer.
2. HP 9000 series 200 or 300 computer with enough memory to hold BASIC, needed binaries, and at least 64 kBytes of program space. In addition, 512 kBytes are needed for BASIC 3.0 or higher operating systems, with the binaries suggested in step 2 in the section `Powering up the system`. A disk drive (e.g. HP 9122) is required to load BASIC if no internal disk drive is available.
3. HP BASIC 2.0 with AP2–1, or BASIC 3.0 or higher operating system.
4. HP 10833A/B/C/D HP–IB cables to interconnect the computer, the network analyzer, and any peripherals.

Microsoft® is a U.S. registered trademark of Microsoft Corporation.
MS–DOS® is a U.S. registered trademark of Microsoft Corporation.

Optional equipment

1. HP 85032B 50 ohm type–N calibration kit.
2. HP 11852D test port return cables.
3. A test device such as a filter to use in the example measurement programs.
4. HP 7440A ColorPro plotter, an HP 2225A Thinkjet printer, or an HP 9122 or HP 9153 CS80 disk drive. See the General Information section of the manual for a more complete list of compatible peripherals.

Using other computers

Although the examples in this guide apply only to the equipment listed above, other computers can control the network analyzer.

- HP VECTRA Personal Computer using an HP 82300 BASIC Language Processor.
- MS–DOS® compatible computer (PC) with HP 82990A "HP–IB Interface and Command Library". Microsoft® Quick BASIC 4.0 is fully compatible with the HP 82990A.

Powering up the system

1. **Set up the network analyzer as shown in Figure 1.** Connect the network analyzer to the computer with an HP–IB cable. The network analyzer has only one HP–IB interface, but it occupies two addresses: one for the instrument, one for the display. The display address is the instrument address with the least significant bit complemented. The default addresses are 16 for the instrument, 17 for the display. Devices on the HP–IB cannot occupy the same address as the network analyzer.
2. **Turn on the computer and load the BASIC operating system.** For BASIC 2.0, load AP2–1 if available. If BASIC 3.0 or higher is used, load the following BASIC binary extensions: HPIB, GRAPH, IO, KBD, and ERR. Depending on the disk drive, a binary such as CS80 may be also required.
3. **Turn the network analyzer on.** To verify the network analyzer's address, press **[LOCAL]** ***[SET ADDRESSES]*** and ***[ADDRESS: 875x]***. If the address has been changed from the default value (16), return it to 16 while performing the examples in this document by pressing **[1] [6] [x1]** and then pressetting the instrument. Make sure the instrument is in either ***[USE PASS CONTROL]*** or ***[TALKER/LISTENER]*** mode, as indicated under the **[LOCAL]** key. These are the only modes in which the network analyzer will accept commands over HP–IB.
4. **On the computer, type the following:** `OUTPUT 716;"PRES;"`**[EXECUTE]** or **[RETURN]**) This will preset the network analyzer. If Preset does not occur, there is a problem. First check all HP–IB addresses and connections: most HP–IB problems are caused by an incorrect address and bad or loose HP–IB cables.

NOTE: Only the 9826 and 9836 computers have an **[EXECUTE]** key. The HP 216 has an **[EXEC]** key with the same function. All other computers use the **[RETURN]** key as both execute and enter. The notation **[EXECUTE]** is used in this document.

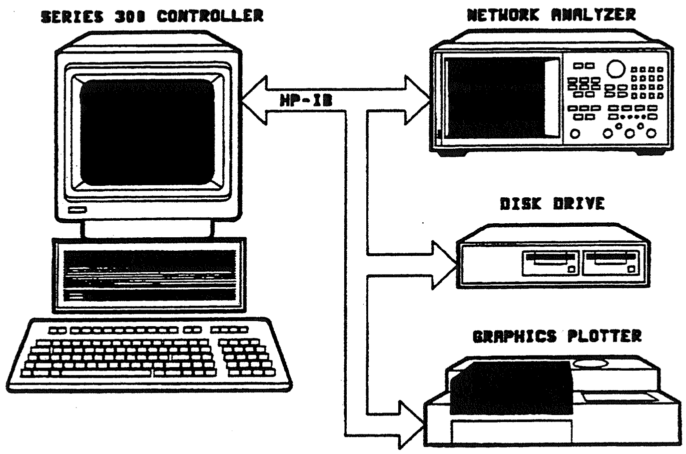


Figure 1. HP–IB connections in a typical setup.

Basic Instrument Control

A computer controls the network analyzer by sending it commands over HP–IB. The commands are specific to the network analyzer. Each command is executed automatically, taking precedence over manual control of the network analyzer. A command applies only to the active channel except where functions are coupled between channels, just as with front panel operation. Most commands are equivalent to front panel functions. For example, type:

OUTPUT 716;"STAR 10 MHZ;" and press **[EXECUTE]**.

The network analyzer now has a start frequency of 10 MHz. The construction of the command is:

OUTPUT 716;"STAR 10 MHZ;"

The BASIC data output statement. The data is directed to interface 7 (HP–IB), and on out to the device at address 16 (the network analyzer).

The network analyzer mnemonic for setting the start frequency. The mnemonic, less the quotation marks, is sent literally by the OUTPUT statement, followed by a carriage return, line feed.

The STAR 10 MHZ; command performs the same function as pressing **[START]** and keying in 10 **[M/u]**. STAR is the root mnemonic for the start key, 10 is the data, and MHZ are the units. The network analyzer's root mnemonics are derived from the equivalent key label where possible, otherwise from the common name for the function. The *HP–IB Quick Reference* lists all the root mnemonics, and all the different units accepted.

The semicolon following MHZ terminates the command inside the network analyzer. It removes start frequency from the active entry area, and prepares the network analyzer for the next command. If there is a syntax error in a command, the network analyzer will ignore the command and look for the next terminator. When it finds the next terminator, it starts processing incoming commands normally. Characters between the syntax error and the next terminator are lost. A line feed also acts as terminator. The BASIC OUTPUT statement transmits a carriage return, line feed following the data. This can be suppressed by putting a semicolon at the end of the statement.

The OUTPUT 716; statement will transmit all items listed, as long as they are separated by commas or semicolons. It will transmit literal information enclosed in quotes, numeric variables, string variables, and arrays. A carriage return, line feed is transmitted after each item. This can be stopped by separating items with semicolons instead of commas.

The front panel remote (R) and listen (L) HP–IB status indicators are on. The network analyzer automatically goes into remote mode when sent a command with the OUTPUT statement. In remote mode, the network analyzer ignores all front panel keys except the local key. Pressing the **[LOCAL]** key returns the network analyzer to manual operation, unless the universal HP–IB command LOCAL LOCKOUT 7 has been issued. The only way to get out of local lockout is to issue the LOCAL 7 command, or cycle power on the network analyzer.

Setting a parameter is one form of command the network analyzer will accept. It will also accept simple commands that require no operand. For example, execute:

OUTPUT 716;"AUTO;"

In response, the network analyzer autoscales the active channel. Autoscale only applies to the active channel, unlike start frequency, which applies to both channels as long as the channels are stimulus coupled.

The network analyzer will also accept commands that turn various functions on and off. Execute:

OUTPUT 716;"DUACON;"

This causes the network analyzer to display both channels. To go back to single channel display mode, execute:

OUTPUT 716;"DUACOFF;"

The construction of the command starts with the root mnemonic DUAC (dual channel display,) and ON or OFF appended to the root to form the entire command.

The network analyzer does not distinguish between upper and lower case letters. For example, execute:

OUTPUT 716;"auto;" The network analyzer also has a debug mode to aid in trouble–shooting systems. When debug mode is on, thenetwork analyzer scrolls incoming HP–IB commands across the display. To turn the mode on manually, press **[LOCAL]** *[HP–IB DIAG ON]*. To turn it on over HP–IB, execute:

OUTPUT 716;"DEBUON;"

Command interrogate

Suppose the operator has changed the power level from the front panel. The computer can find the new level by using the network analyzer's command interrogate function. If a question mark is appended to the root of a command, the network analyzer will output the value of that function. For instance, `POWE 5 DB;` sets the output power to 5 dB, and `POWE?;` outputs the current RF output power at the test port. For example, type SCRATCH and press **[EXECUTE]** to clear old programs. Type EDIT and press **[EXECUTE]** to get into the edit mode. Then type in:

```
10      OUTPUT 716;"POWE?;"
20      ENTER 716;Reply
30      DISP Reply
40      END
```

Run the program. The computer will display the source power level in dBm. The preset level is 0 dBm. Change the power level by pressing **[LOCAL] [MENU]** ***[POWER]*** and then entering **[1] [x1]**. Run the program again.

When the network analyzer receives `POWE?`, it prepares to transmit the current RF source power level. The BASIC statement `ENTER 716` allows the network analyzer to transmit information to the computer by addressing it to talk. This turns the network analyzer front panel talk light (T) on. The computer places the data transmitted by the network analyzer into the variables listed in the enter statement. In this case, the network analyzer transmits the output power, which gets placed in the variable `Reply`.

The `ENTER` statement takes the stream of binary data output by the network analyzer and reformats it back into numbers and ASCII strings. With the formatting in its default state, the enter statement will format the data into real variables, integers, or ASCII strings, depending on the variable being filled. The variable list must match the data the network analyzer has to transmit: if there are too few variables, data is lost, and if there are too many variables for the data available, a BASIC error is generated.

The formatting done by the enter statement can be changed. As discussed in *Data transfer from analyzer to computer*, the formatting can be turned off to allow binary transfers of data. Also, the `ENTER USING` statement can be used to selectively control the formatting.

On/off commands can be also be interrogated. The reply is a one if the function is on, a zero if it is off. Similarly, if a command controls a function that is underlined on the network analyzer display when active, interrogating that command yields a one if the command is underlined, a zero if it is not. For example, there are nine options on the format menu: only one is underlined at a time. The underlined option will return a one when interrogated.

For instance, rewrite line 10 as:

```
10      OUTPUT 716;"DUAC?;"
```

Run the program once, note the result, then press **[LOCAL] [DISPLAY]** ***[DUAL CHAN]*** to toggle the display mode, and run the program again.

Another example is to rewrite line 10 as:

```
10      OUTPUT 716;"PHAS?;"
```

In this case, the program will display a one if phase is currently being displayed. Since the command only applies to the active channel, the response to the `PHAS?` inquiry depends on which channel is active.

Held commands

When the network analyzer is executing a command that cannot be interrupted, it will hold off processing new HP-IB commands. It will fill the 16 character input buffer, and then halt HP-IB until the held command has completed execution. This action will be clear to a programmer unless HP-IB timeouts have been set with the `ON TIMEOUT` statement.

While a held command is executing, the network analyzer will still service the HP-IB interface commands, such as `SPOLL(716)`, `CLEAR 716`, and `ABORT 7`. Executing `CLEAR 716` or `CLEAR 7` will abort a command hold off, leaving the held command to complete execution as if it had been begun from the front panel. These commands also clear the input buffer, destroying any commands received after the held command. If the network analyzer has halted the bus because its input buffer was full, `ABORT 7` will release the bus.

Operation complete

Occasionally, there is a need to find out when certain operations have completed inside the network analyzer. For instance, a program should not have the operator connect the next calibration standard while the network analyzer is still measuring the current one.

To provide such information, the network analyzer has an Operation Complete reporting mechanism that will indicate when certain key commands have completed operation. The mechanism is activated by sending either `OPC` or `OPC?` immediately before an OPC'able command. When the command completes execution, bit 0 of the event status register will be set. If OPC was interrogated with `OPC?`, the network analyzer will output a 1 when the command completes execution.

Chapter 3. HP-IB Remote Programming

CHAPTER CONTENTS

INTRODUCTION

The analyzer is factory-equipped with a remote programming digital interface using the Hewlett-Packard Interface Bus (HP-IB). (HP-IB is Hewlett-Packard's hardware, software, documentation, and support for IEEE 488.1 and IEC-625, worldwide standards for interfacing instruments.) This allows the analyzer to be controlled by an external computer that sends commands or instructions to and receives data from the analyzer using the HP-IB. In this way, a remote operator has the same control of the instrument available to a local operator from the front panel, except for control of the line power switch.

In addition, the analyzer itself can use HP-IB to directly control compatible peripherals, without the use of an external controller. It can output measurement results directly to a compatible printer or plotter, or store instrument states to a compatible disk drive.

This chapter provides an overview of HP-IB operation. Chapter 10 provides information on different controller modes, and on setting up the analyzer as a controller of peripherals. It also explains how to use the analyzer as a controller to print, plot, and store to an external disk. HP-IB equivalent mnemonics for front panel functions are provided in parentheses throughout this manual.

More complete information on programming the analyzer remotely over HP-IB is provided in the following documents:

- *HP-IB Programming Guide for the HP 8752A and HP 8753C Using the HP 9000 Series 200/300 Desktop Computer (BASIC)*. This is a tutorial introduction to remote operation of the analyzer using an HP 9000 series 200 or 300 computer. It includes examples of remote measurements using BASIC programming. These examples are also stored on the example programs disk provided with the analyzer. The *HP-IB Programming Guide* assumes familiarity with front panel operation of the instrument.

- *HP-IB Quick Reference for the HP 8700-Series Analyzers*. This is a complete reference summary for remote operation of the analyzer with a controller. It includes both functional and alphabetical lists of all analyzer HP-IB commands. This guide is intended for use by those familiar with HP-IB programming and the basic functions of the analyzer.

A complete general description of the HP-IB is available in *Tutorial Description of the Hewlett-Packard Interface Bus*, HP publication 5952-0156. For more information on the IEEE-488.1 standard refer to *IEEE Standard Digital Interface for Programmable Instrumentation*, published by the Institute of Electrical and Electronics Engineers, Inc., 345 East 47th Street, New York, New York 10017.

HOW HP-IB WORKS

The HP-IB uses a party-line bus structure in which up to 15 devices can be connected on one contiguous bus. The interface consists of 16 signal lines and 8 ground lines in a shielded cable. With this cabling system, many different types of devices including instruments, computers, plotters, printers, and disk drives can be connected in parallel.

Every HP-IB device must be capable of performing one or more of the following interface functions:

Talker

A talker is a device capable of sending device-dependent data when addressed to talk. There can be only one talker at any given time. Examples of this type of device are voltmeters, counters, and tape readers. The analyzer is a talker when it sends trace data or marker information over the bus.

Listener

A listener is a device capable of receiving device-dependent data when addressed to listen. There can be any number of listeners at any given time. Examples of this type of device are printers, power supplies, and signal generators. The analyzer is a listener when it is controlled over the bus by a computer.

Controller

A controller is a device capable of managing the operation of the bus and addressing talkers and listeners. There can be only one active controller at any time. Examples of controllers include desktop computers and minicomputers. In a multiple-controller system, active control can be passed between controllers, but there can only be one *system controller*, which acts as the master, and can regain active control at any time. The analyzer is an active controller when it plots, prints, or stores to an external disk drive in the pass control mode. The analyzer is a system controller when it is in the system controller mode. These modes are discussed in more detail in Chapter 10 under *HP-IB Menu*.

HP-IB BUS STRUCTURE

Data Bus

The data bus consists of eight bidirectional lines that are used to transfer data from one device to another. Programming commands and data are typically encoded on these lines in ASCII, although binary encoding is often used to speed up the transfer of large arrays. Both ASCII and binary data formats are available to the analyzer. In addition, every byte transferred over HP-IB undergoes a *handshake* to ensure valid data.

Handshake Lines

A three-line handshake scheme coordinates the transfer of data between talkers and listeners. This technique forces data transfers to occur at the speed of the slowest device, and ensures data integrity in multiple listener transfers. With most computing controllers and instruments, the handshake is performed automatically, which makes it transparent to the programmer.

Control Lines

The data bus also has five control lines that the controller uses both to send bus commands and to address devices.

IFC. Interface Clear. Only the system controller uses this line. When this line is true (low), all devices (addressed or not) unaddress and go to an idle state.

ATN. Attention. The active controller uses this line to define whether the information on the data bus is a *command* or is *data*. When this line is true (low), the bus is in the command mode and the data lines carry bus commands. When this line is false (high), the bus is in the data mode and the data lines carry device-dependent instructions or data.

SRQ. Service Request. This line is set true (low) when a device requests service: the active controller services the requesting device. The analyzer can be enabled to pull the SRQ line for a variety of reasons.

REN. Remote Enable. Only the system controller uses this line. When this line is set true (low), the bus is in the remote mode, and devices are addressed either to listen or to talk. When the bus is in remote and a device is addressed, it receives instructions from HP-IB rather than from its front panel (the **[LOCAL]** key returns the device to front panel operation). When this line is set false (high), the bus and all devices return to local operation.

EOI. End or Identify. This line is used by a talker to indicate the last data byte in a multiple byte transmission, or by an active controller to initiate a parallel poll sequence. The analyzer recognizes the EOI line as a terminator, and it pulls the EOI line with the last byte of a message output (data, markers, plots, prints, error messages). The analyzer does not respond to parallel poll.

Figure 3-1 illustrates the structure of the HP-IB bus lines.

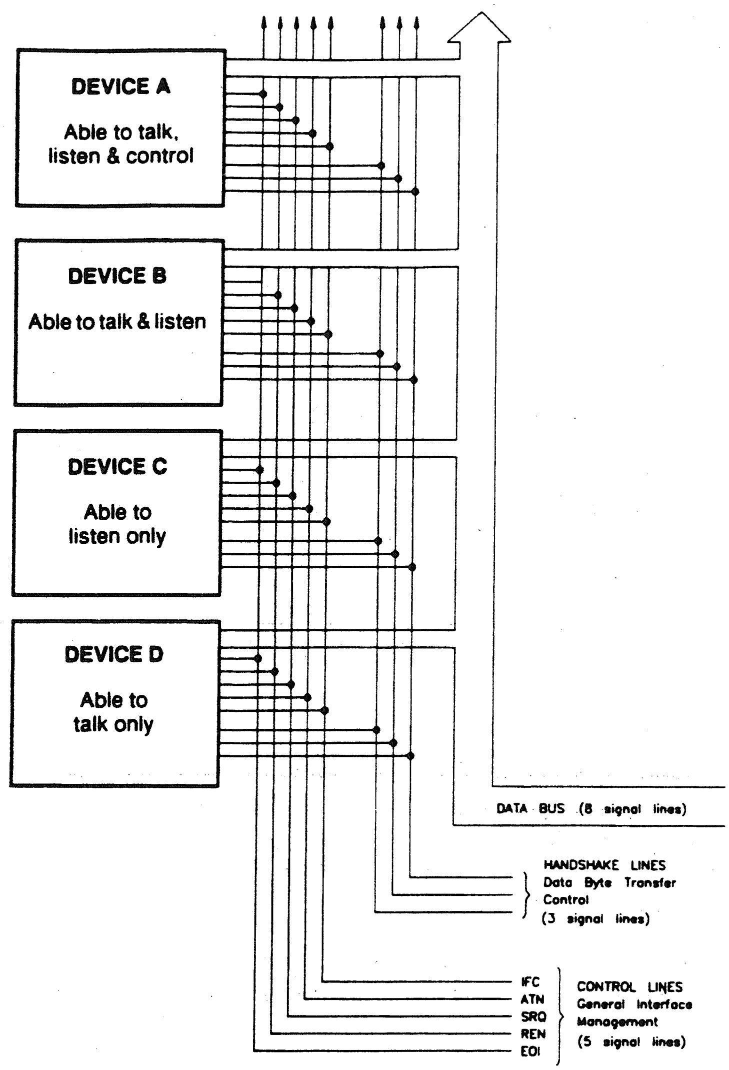


Figure 3-1. HP-IB Structure

HP-IB REQUIREMENTS

Number of Interconnected Devices:	15 maximum.
Interconnection Path/ Maximum Cable Length:	20 meters maximum or 2 meters per device, whichever is less.
Message Transfer Scheme:	Byte serial/ bit parallel asynchronous data transfer using a 3-line handshake system.
Data Rate:	Maximum of 1 megabyte per second over limited distances with tri-state drivers. Actual data rate depends on the transfer rate of the slowest device involved.
Address Capability:	Primary addresses: 31 talk, 31 listen. A maximum of 1 talker and 14 listeners at one time.
Multiple Controller Capability:	In systems with more than one controller (like the analyzer system), only one can be active at a time. The active controller can pass control to another controller, but only the system controller can assume unconditional control. Only one system controller is allowed. The system controller is hard-wired to assume bus control after a power failure.

ANALYZER HP-IB CAPABILITIES

As defined by the IEEE 488.1 standard, the analyzer has the following capabilities:

SH1	Full source handshake.
AH1	Full acceptor handshake.
T6	Basic talker, answers serial poll, unaddresses if MLA is issued. No talk-only mode.
L4	Basic listener, unaddresses if MTA is issued. No listen-only mode.
SR1	Complete service request (SRQ) capabilities.
RL1	Complete remote/local capability including local lockout.
PP0	Does not respond to parallel poll.
DC1	Complete device clear.
DT1	Responds to a group execute trigger in the hold trigger mode.
C1,C2,C3	System controller capabilities in system controller mode.
C10	Pass control capabilities in pass control mode.
E2	Tri-state drivers.

BUS MODE

The analyzer uses a single-bus architecture. The single bus allows both the analyzer and the host controller to have complete access to the peripherals in the system.

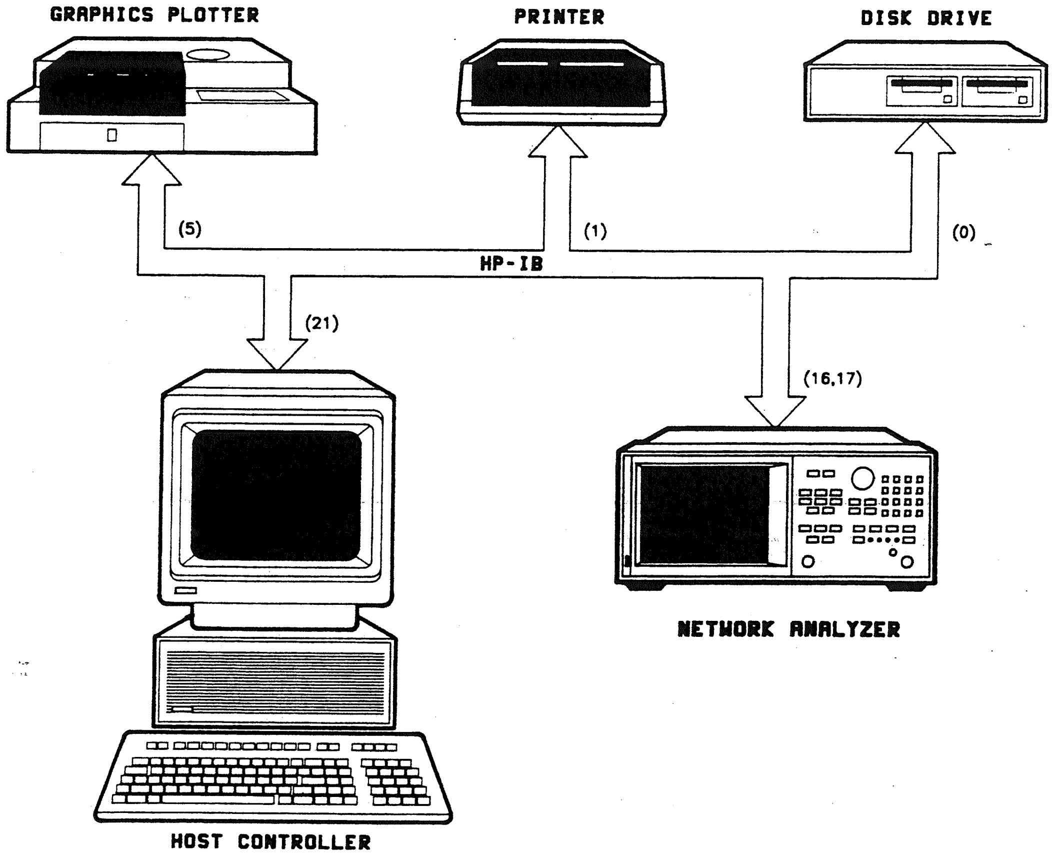


Figure 3-2. Analyzer Single Bus Concept

Three different controller modes are possible, system controller, talker/listener, and pass control.

System Controller. This mode allows the analyzer to control peripherals directly in a stand-alone environment (without an external controller). This mode can only be selected manually from the analyzer front panel. Use this mode for operation when no computer is connected to the analyzer. Do not use this mode for programming.

Talker/Listener. This is the traditional programming mode, in which the computer is involved in all peripheral access operations. Peripheral access (plotting and printing only) is also possible by addressing the analyzer to talk, addressing the peripheral to listen, and placing the HP-IB in the data mode.

Pass Control. This mode allows you to control the analyzer over HP-IB as with the talker/listener mode, and also allows the analyzer to take or pass control in order to plot, print, and access a disk. During the peripheral operation, the host computer is free to perform other internal tasks such as data or display manipulation (the bus is tied up by the analyzer during this time). After a task is completed, the host controller accepts control again when the analyzer returns it.

In general, use the talker/listener mode for programming the analyzer unless you desire direct peripheral access. Preset does not affect the selected bus mode, but the bus mode returns talker/listener if power is cycled.

Chapter 10 explains the three different bus modes in detail, and provides information on setting the correct bus mode. Programming information for talker/listener mode and pass control mode is provided in the *HP-IB Programming Guide.*

SETTING ADDRESSES

In communications through HP-IB, each instrument on the bus is identified by an HP-IB address. This address code must be different for each instrument on the bus. Refer to *Address Menu* in Chapter 10 for information on default addresses, and on setting and changing addresses. These addresses are not affected when you press **[PRESET]** or cycle the power (although the **[PRESET]** key must be pressed to implement a change to the analyzer address).

VALID CHARACTERS

The analyzer accepts ASCII letters, numbers, decimal points, +/−, semicolons, quotation marks ("), carriage returns (CR), and linefeeds (LF). Both upper and lower case are acceptable. Leading zeros, spaces, carriage returns, and unnecessary terminators are ignored, except those within a command or appendage. Carriage returns are ignored. An invalid character causes a syntax error. Syntax errors are described in more detail under in the *HP-IB Programming Guide*.

CODE NAMING CONVENTION

The analyzer HP-IB commands are derived from their front panel key titles (where possible), according to the naming convention below.

Convention	Key Title	For HP-IB Code Use	Example
One Word	Power Start	First Four Letters	POWE STAR
Two Words	Electrical Delay Search Right	First Three Letters of First Word First Letter of Second Word	ELED SEAR
Two Words in a Group	Marker →Center Gate →Span	First Four Letters of Both	MARKCENT GATESPAN
Three Words	Cal Kit N 50Ω Pen Num Data	First Three Letters of First Word First Letter of Second Word First Four Letters of Third Word	CALKN50 PENNDATA

Some codes require appendages (on, off, 1, 2, etc.). Codes that have no front panel equivalent are HP-IB only commands, and use a similar convention based on the common name of the function. Where possible, analyzer codes are compatible with HP 8510A/B codes.

Front panel equivalent codes and HP-IB only codes are summarized in the *HP-IB Quick Reference*.

UNITS AND TERMINATORS

The analyzer outputs data in basic units and assumes these basic units when it receives an input, unless the input is otherwise qualified. The basic units and allowable expressions follow; either upper or lower case is acceptable.

Basic Units	Allowable Expressions
Seconds	S
Milliseconds	MS
Microseconds	US
Nanoseconds	NS
Picoseconds	PS
Femtoseconds	FS
Hertz	HZ
Kilohertz	KHZ
Megahertz	MHZ
Gigahertz	GHZ
dB or dBm	DB
Volts	V

Terminators are used to indicate the end of a command to allow the analyzer to recover to the next command in the event of a syntax error. The semicolon is the recommended command terminator. The line feed (LF) character and the HP-IB EOI line can also be used as terminators. The analyzer ignores the carriage return (CR) character.

HP-IB DEBUG MODE

An HP-IB diagnostic feature (debug mode) is available in the HP-IB menu. Activating the debug mode causes the analyzer to scroll incoming HP-IB commands across the display. Nonprintable characters are represented with a π. Any time the analyzer receives a syntax error, the commands halt, and a pointer $\wedge$ indicates the misunderstood character. The *HP-IB Programming Guide* explains how to clear a syntax error.

DISPLAY GRAPHICS

The analyzer display can be used as a graphics display for displaying connection diagrams or custom instructions to an operator. The display accepts a subset of Hewlett-Packard Graphics Language (HP-GL) commands issued by an external computer. Some user graphics can be created using the test sequencing feature.

NOTE: The analyzer display occupies an additional address on the HP-IB. Determine the display bus address by adding 1 to the analyzer address (if the analyzer address is an even number), or subtracting 1 (if it is an odd number). Thus the factory default display address for graphics is 17.

As an example: type **SCRATCH**, press **[EXECUTE]**, type **EDIT**, press **[EXECUTE]**, and type in the following program:

`10 OUTPUT 716;"SWET 3 S;OPC?;SING;"`	Set the sweep time to 3 seconds, and OPC a single sweep.
`20 DISP "SWEEPING"`	
`30 ENTER 716;Reply`	The program will halt until the network analyzer completes the sweep and issues a one.
`40 DISP "DONE"`	
`50 END`	

Running this program causes the computer to display the sweeping message for about 3 seconds, as the instrument executes the sweep. The computer will display `DONE` just as the instrument goes into hold. When the `DONE` message appears, the program could then continue on, being assured that there is a valid data trace in the instrument. Without single sweep, we would have had to wait at least two sweep times to ensure good data.

Preparing for HP-IB control

At the beginning of a program, the network analyzer has to be taken from an unknown state and brought under computer control. One way to do this is with an abort/clear sequence. `ABORT 7` is used to halt bus activity and return control to the computer. `CLEAR 716` will then prepare the network analyzer to receive commands by clearing syntax errors, the input command buffer, and any messages waiting to be output.

The abort/clear sequence makes the network analyzer ready to receive HP-IB commands. The next step is to put the network analyzer into a known state. The most convenient way to do this is to send `PRES`, which returns the instrument to the preset state. If preset cannot be used and the status reporting mechanism is going to be used, `CLES` can be sent to clear all of the status reporting registers and their enables.

Type **SCRATCH**, press **[EXECUTE]**, type **EDIT**, press **[EXECUTE]**, and type in the following program:

`10 ABORT 7`	This halts all bus action and gives active control to the computer.
`20 CLEAR 716`	This clears all HP-IB errors, resets the HP-IB interface, clears syntax errors. It does not affect the status reporting system.
`30 OUTPUT 716;"PRES;"`	Preset the instrument. This clears the status reporting system, as well as resetting all the front panel settings, except the HP-IB mode and HP-IB addresses.
`40 END`	

This program brings the network analyzer to a known state, ready to respond to HP-IB control.

The network analyzer will not respond to HP-IB commands unless the remote line is asserted. When the remote line is asserted and the network analyzer is addressed to listen, it automatically goes into remote mode. Remote mode means that all the front panel keys are disabled except **[LOCAL]** and the line power switch. `ABORT 7` asserts the remote line, which remains asserted until a `LOCAL 7` statement is executed. Another way to assert the remote line is to execute:

```
REMOTE 716
```

This statement asserts remote and addresses the network analyzer to listen so it goes into remote mode. Press any front panel key except local. None will respond until you press **[LOCAL]**.

The local key can also be disabled with the sequence:

```
REMOTE 716
LOCAL LOCKOUT 7
```

Now no front panel keys will respond. The HP 8753C can be returned to local mode temporarily with:

```
LOCAL 716
```

But as soon as the HP 8753C is next addressed to listen, it goes back into local lockout. The only way to clear local lockout, aside from cycling power, is to execute:

```
LOCAL 7
```

Which un-asserts the remote line on the interface. This puts the instrument into local mode and clears local lockout. Be sure to put the instrument back into remote mode.

Measurement Programming

The previous section of this document outlined how to get commands into the network analyzer. The next step is to organize the commands into a measurement sequence. A typical measurement sequence consists of the following steps:

1. Set up the instrument.
2. Calibrate.
3. Connect the device.
4. Take data.
5. Post process data.
6. Transfer data.

Set up the instrument

Define the measurement by setting all of the basic measurement parameters. These include all the stimulus parameters: sweep type, span, sweep time, number of points, and RF power level. They also include the parameter to be measured, and both IF averaging and IF bandwidth. These parameters define the way data is gathered and processed within the instrument, and to change one requires that a new sweep be taken.

There are other parameters that can be set within the instrument that do not affect data gathering directly, such as smoothing, trace scaling or trace math. These functions are classed as post processing functions: they can be changed with the instrument in hold mode, and the data will correctly reflect the current state.

The save/recall registers and the learn string are two rapid ways of setting up an entire instrument state. The learn string is a summary of the instrument state compacted into a string that can be read into the computer and retransmitted to the network analyzer. See *Example 6A, Using the learn string*, for a discussion of how to do this.

Calibrate

Measurement calibration is normally performed once the instrument state has been defined. Measurement calibration is not required to make a measurement, but it does improve the accuracy of the data.

There are several ways to calibrate the instrument. The simplest is to stop the program and have the operator perform the calibration from the front panel. Alternatively, the computer can be used to guide the operator through the calibration, as discussed in Example 2A and 2B, *1-port calibration and Full 2-port calibration* (HP 8753C only. Full 2-port calibration is not available in the HP 8752A). The last option is to transfer calibration data from a previous calibration back into the instrument, as discussed in Example 6C, *Reading calibration data.*

Connect device under test

Have the operator connect and adjust the device. The computer can be used to speed the adjustment process by setting up such functions as limit testing, bandwidth searches, and trace statistics. All adjustments take place at this stage so that there is no danger of taking data from the device while it is being adjusted.

Take data

With the device connected and adjusted, measure its frequency response, and hold the data within the instrument so that there is a valid trace to analyze.

The single sweep command `SING` is designed to ensure a valid sweep. All stimulus changes are completed before the sweep is started, and the HP–IB hold state is not released until the formatted trace is displayed. When the sweep is complete, the instrument is put into hold, freezing the data inside the instrument. Because single sweep is OPC'able, it is easy to determine when the sweep has been completed.

The number of groups command `NUMGn` is designed to work the same as single sweep, except that it triggers n sweeps. This is useful, for example, in making a measurement with an averaging factor n. (n can be 1 to 999). Both single sweep and number of groups restart averaging.

Post process

With valid data to operate on, the post–processing functions can be used. Referring ahead to Figure 2, any function that affects the data after the error correction stage can be used. The most useful functions are trace statistics, marker searches, electrical delay offset, time domain, and gating. If a 2–port calibration is active, then any of the four S–parameters can be viewed without taking a new sweep.

Transfer data

Lastly, read the results out of the instrument. All the data output commands are designed to ensure that the data transmitted reflects the current state of the instrument:

- `OUTPDATA`, `OUTPRAWn`, and `OUTPFORM` will not transmit data until all formatting functions have completed.
- `OUTPLIML`, `OUTPLIMM`, and `OUTPLIMF` will not transmit data until limit test has occurred, if on.
- `OUTPMARK` will activate a marker if one is not already selected, and it will make sure that any current marker searches have completed before transmitting data.
- `OUTPMSTA` makes sure that statistics have been calculated for the current trace before transmitting data. If statistics is not on, it will turn statistics on to update the current values, and then turn it off.
- `OUTPMWID` makes sure that a bandwidth search has been executed for the current trace before transmitting data. If bandwidth search is not on, it will turn the search on to update the current values, and then turn it off.

Data transfer is discussed further in Examples 3A through 3C, *Data transfer using ASCII transfer format, etc.*

Basic Programming Examples

Example 1: Setting up a basic measurement

In general, the procedure for setting up measurements on the network analyzer via HP–IB follows the same sequence as if the setup was performed manually. There is no required order, as long as the desired frequency range, number of points and power level are set prior to performing the calibration.

This example illustrates how a basic measurement can be set up on the network analyzer. The program will first select the desired parameter, the measurement format, and then the frequency range. Performing calibrations is described later.

By interrogating the analyzer to determine the actual values of the start and stop frequencies, the computer can keep track of the actual frequencies.

This example program is stored on the Example Programs disk as **IPG1.**

10	`ABORT 7`	
20	`CLEAR 716`	Prepare for HP–IB control.
30	`OUTPUT 716;"PRES;"`	Preset the network analyzer.
40	`OUTPUT 716;"CHAN1; S11; LOGM;"`	Make channel 1 the active channel, and measure the reflection parameter, S_{11} for the HP 8753C, or REFL for the HP 8752A, displaying its magnitude in dB.
50	`OUTPUT 716;"CHAN2; S11; PHAS;"`	Make channel 2 the active channel, and measure the phase of S_{11} on it.
60	`OUTPUT 716;"DUACON;"`	Tell the analyzer to display both channels simultaneously.
70	`INPUT "ENTER START FREQUENCY (MHz):",F_start`	Input a start frequency.
80	`INPUT "ENTER STOP FREQUENCY (MHz):",F_stop`	Input a stop frequency.
90	`OUTPUT 716;"STAR"; F_start;"MHZ;"`	Set the start frequency to `F_start`.
100	`OUTPUT 716;"STOP";F_stop; "MHZ;"`	Set the stop frequency to `F_stop`.
110	`DISP F_start, F_Stop`	Show the current start and stop frequencies.
120	`END`	

Running the program

The program sets up a measurement of reflection log magnitude on channel 1, reflection phase on channel 2, and turns on the dual channel display mode. When prompted for start and stop frequencies, enter any value in MHz from 0.3 (300 kHz) to 3 GHz (1.3 GHz for the HP 8752A). These will be entered into the network analyzer, and the frequencies will be displayed.

Performing a measurement calibration

This section will demonstrate how to coordinate a measurement calibration over HP–IB. The HP–IB program follows the keystrokes required to calibrate from the front panel: there is a command for every step.

The general key sequence is to select the calibration, measure the calibration standards, and then declare the calibration done. The actual sequence depends on the calibration kit and changes slightly for 2-port calibrations*, which are divided into three calibration sub-sequences.

Calibration kits

The calibration kit tells the network analyzer what standards to expect at each step of the calibration. The set of standards associated with a given calibration is termed a class. For example, measuring the short during a 1-port calibration is one calibration step. All of the shorts that can be used for this calibration step make up the class, which is called class $S_{11}B$. For the 7 mm* and the 3.5 mm cal kits, class $S_{11}B$ has only one standard in it. For type-N cal kits, class $S_{11}B$ has two standards in it: male and female shorts.

When doing a 1-port calibration in 7* or 3.5 mm, selecting ***[SHORT]*** automatically measures the short because there is only one standard in the class. When doing the same calibration in type-N, selecting ***[SHORTS]*** brings up a second menu, allowing the user to select which standard in the class is to be measured. The sex listed refers to the test port: if the test port is female, then the user selects the female short option.

Doing a 1-port calibration over HP-IB is very similar. In 7* or 3.5 mm, sending `CLASS11B` will automatically measure the short. In type-N, sending `CLASS11B` brings up the menu with the male and female short options. To select a standard, use `STANA` or `STANB`. The `STAN` command is appended with the letters A through G, corresponding to the standards listed under softkeys 1 through 7, softkey 1 being the topmost softkey.

The `STAN` command is OPC'able. A command that calls a class is only OPC'able if that class has only one standard in it. If there is more than one standard in a class, the command that calls the class only brings up another menu, and there is no need to OPC it.

Hence, both the manual and HP-IB calibration sequences depend heavily on which calibration kit is active.

Full 2-port calibrations (HP 8753C only)

Each full 2-port measurement calibration is divided into three sub-sequences: transmission, reflection, and isolation. Each subsequence is treated like a calibration in its own right: each must be opened, have all the standards measured, and then be declared done.

The opening and closing statements for the transmission sub-sequence are `TRAN` and `TRAD`. The opening and closing statements for the reflection sub-sequence are `REFL` and `REFD`. The opening and closing statements for isolation are `ISOL` and `ISOD`.

*HP 8753 only.

Example 2A: 1-port calibration

To demonstrate coordinating a calibration over HP-IB, the following program does a 1-port calibration, using the HP 85032B 50 ohm type-N calibration kit. This program simplifies the calibration for the operator by giving explicit directions on the network analyzer display, and allowing the user to continue the program from the network analyzer front panel.

This example program is stored on the Example Programs disk as **IPG2A**.

10	`ABORT 7`	
20	`CLEAR 716`	Prepare for HP-IB control.
30	`OUTPUT 716;"CALKN50; MENUOFF;CLES;ESE 64;"`	This is the minimum instrument set up: the 50 ohm type-N cal kit is selected, the softkey menu is turned off, and the status reporting system is set up so that bit 6, User Request, of the event status register, is summarized by bit 5 of the status byte. This allows us to detect a key press with a serial poll. Refer to Appendix A.
40	`OUTPUT 716;"CALIS111;"`	Open the calibration by calling the S_{11} 1-port calibration.
50	`CALL Waitforkey("CONNECT LOAD AT PORT 1")`	Now ask for the load, and wait for the operator. The Waitforkey subroutine will not return until the operator presses a key on the front panel of the network analyzer.
60	`OUTPUT 716;"OPC?; CLASS11C;"`	There is only one choice in this class, so the CLASS command is OPC'able. Using the OPC? command causes the program to wait until the standard has been measured before continuing. This is very important, because the prompt to connect the next standard should only appear after the first standard is measured.
70	`ENTER 716;Reply`	Wait until the network analyzer is done with the standard.
80	`CALL Waitforkey("CONNECT OPEN AT PORT 1")`	Ask for an open, and wait for the operator to connect it.
90	`OUTPUT 716;"CLASS11A; OPC?; STAN;"`	Measure the open. There is more than one standard in this loads class, so we must identify the specific standard within that class. The female open is the second softkey selection from the top in the menu, so select a lowband load as the standard using the command `STANB`.
100	`ENTER 716;Reply`	Wait for the standard to be measured.
110	`CALL Waitforkey("CONNECT SHORT LOAD AT PORT 1")`	Have the operator connect the short and wait for a reply.
120	`OUTPUT 716;"CLASS11B; OPC?; STANB;"`	There is more than one standard in the short class, too. The specific standard is the female short, or STAN B. Measure the short.
130	`ENTER 716;Reply`	Wait for the standard to be measured.
140	`OUTPUT 717;"PG;"`	The `PG` command sent to the display clears the user graphics, removing the last prompt.

150	`DISP "COMPUTING CALIBRATION COEFFICIENTS"`	
160	`OUTPUT 716;"DONE;OPC?; SAV1;"`	Affirm the completion of the calibration, and save the calibration.
170	`ENTER 716;Reply`	Wait until the network analyzer is done calculating the calibration coefficients before allowing the program to go on.
180	`DISP "1-PORT CAL COMPLETED. CONNECT TEST DEVICE."`	
190	`OUTPUT 716;"MENUON;"`	The calibration is complete, so turn the softkey menu back on.
200	`END`	
210	`SUB Waitforkey(Lab$)`	This subroutine displays the passed message on the network analyzer, and waits for the operator to press a key. It assumes that bit 6, User Request, of the event status register has been enabled.
220	`DISP Lab$`	First, display a message on the computer in case the operator has returned to the computer keyboard.
230	`OUTPUT 717;"PG;PU;PA390, 3600; PD;LB";Lab$;", PRESS ANY KEY WHEN READY;"`	This statement writes on the network analyzer's display. `PG` (page) clears old user graphics. `PU` (pen up) prevents anything from being drawn. `PA390,3600;` moves the logical pen to just above the message area on the display. `PD` (pen down) enables drawing. `LB` (label) writes the message on the display. The label command is terminated with an ETX symbol, which is **[CTRL] [C]** (pressed simultaneously) on the keyboard.
240	`CLEAR 716`	Clear the message line on the network analyzer.
250	`OUTPUT 716;"ESR?;"`	Clear the latched User Request bit so that old key presses will not trigger a measurement.
260	`ENTER 716;Estat`	
270	`Stat=SPOLL(716)`	Now wait for a key press to be reported.
280	`IF NOT BIT(Stat,5) THEN GOTO 340`	
290	`SUBEND`	

Running the program

The program assumes that the port being calibrated is a 50 ohm, type-N female test port. The prompts appear just above the message line on the network analyzer display. Pressing any key on the front panel of the network analyzer continues the program and measures the standard. The program will display a message when the measurement calibration is complete.

Before running the program, set up the desired instrument state. This program does not modify the instrument state in any way. Run the program, and connect the standards as prompted. When the standard is connected, press any key on the network analyzer's front panel to measure it.

Example 2B: Full 2-port measurement calibration (HP 8753C only)

This example shows how to perform a full 2-port measurement calibration using the HP 85032B calibration kit. The main difference between this example and Example 2A is that in this case, the calibration process allows removal of both the forward and reverse error terms, so that all four S-parameters of the device under test can be measured. Port 1 is a female test port and Port 2 is a male test port. This example program is stored on the Example Programs disk as **IPG2B.**

10	`ABORT 7`	
20	`CLEAR 716`	
30	`OUTPUT 716;"CALKN50; MENUOFF;CLES;ESE 64;"`	This is the minimum instrument set up: the 50 ohm type-N kit is selected, the softkey menu is turned off, and the status reporting system is set up so that bit 6, User Request, of the event status register, is summarized by bit 5 of the status byte. This allows us to detect a key press with a serial poll. Refer to Appendix A.
40	`OUTPUT 716;"CALIFUL2;"`	Open the calibration by calling for a full 2-port calibration.
50	`OUTPUT 716;"REFL;"`	Open the reflection calibration subsequence.
60	`CALL Waitforkey("CONNECT OPEN AT PORT 1")`	Ask for the open, and wait for the operator. The Waitforkey subroutine will not return until a key on the front panel of the HP 8753C is pressed.
70	`OUTPUT 716;"CLASS11A; OPC?;STANB;"`	There is more than one standard in the open class, so we must identify the specific standard within that class. The female open selection is the second softkey from the top in the menu, so we select a broadband load as the standard using the command `STANB`.
80	`ENTER 716;Reply`	Wait until the HP 8753C is done with the standard.
90	`CALL Waitforkey("CONNECT SHORT AT PORT 1")`	Ask for a short, and wait for the operator to connect it.
100	`OUTPUT 716;"CLASS11B; OPC?;STANB;"`	Measure the short.
110	`ENTER 716;Reply`	Wait for the standard to be measured.
120	`CALL Waitforkey("CONNECT BROADBAND LOAD AT PORT 1")`	Have the operator connect the broadband load, and wait for his reply.
130	`OUTPUT 716;"OPC?; CLASS11C;"`	There is only one choice in this class, so the `CLASS` command is OPC'able. Using the `OPC?` command causes the program to wait until the standard has been measured before continuing. This is important, because the prompt to connect the next standard should appear only after the first standard is measured.
140	`ENTER 716;Reply`	Wait for the standard to be measured.
150	`CALL Waitforkey("CONNECT OPEN AT PORT 2")`	Ask for the male open for port 2, and wait for the operator.

160	`OUTPUT 716;"CLASS22A; OPC?;STANA;"`	Measure the open.
170	`ENTER 716;`	ReplyWait until the HP 8753C is done with the standard.
180	`CALL Waitforkey("CONNECT SHORT AT PORT 2")`	Ask for a male short, and wait for the operator to connect it.
190	`OUTPUT 716;"CLASS22B; OPC?;STANA;"`	Measure the short.
200	`ENTER 716;Reply`	Wait for the standard to be measured.
210	`CALL Waitforkey("CONNECT LOAD AT PORT 2")`	Have the operator connect the load, and wait for a reply.
220	`OUTPUT 716;"OPC?; CLASS22C;"`	Measure the load.
230	`ENTER 716;Reply`	Wait for the standard to be measured.
240	`OUTPUT 716;"REFD;"`	Close the reflection calibration subsequence.
250	`DISP "COMPUTING REFLECTION CALIBRATION COEFFICIENTS"`	
260	`OUTPUT 716;"TRAN;"`	Open the transmission calibration subsequence.
270	`CALL Waitforkey("CONNECT THRU (PORT 1 TO PORT 2)")`	
280	`DISP "MEASURING FORWARD TRANSMISSION")`	
290	`OUTPUT 716;"OPC?;FWDT;"`	Measure forward transmission.
300	`ENTER 716;Reply`	
310	`OUTPUT 716;"OPC?;FWDM;"`	Measure forward load match.
320	`ENTER 716;Reply`	
330	`DISP "MEASURING REVERSE TRANSMISSION")`	
340	`OUTPUT 716;"OPC?;REVT;"`	Measure reverse transmission.
350	`ENTER 716;Reply`	
360	`OUTPUT 716;"OPC?;REVM;"`	Measure reverse load match.
370	`ENTER 716;Reply`	
380	`OUTPUT 716;"TRAD;"`	Close the transmission calibration sub–sequence.
390	`INPUT "SKIP ISOLATION CAL? Y OR N.", An$`	Ask operator if the isolation cal should be skipped.
400	`"IF An$="Y" THEN`	If the answer is yes, skip the isolation cal and branch to the computation of the calibration coefficients.
410	`OUTPUT 716;"OMII;"`	
420	`GOTO 520`	
430	`END IF`	
440	`CALL Waitforkey("ISOLATE TEST PORTS")`	Ask operator to isolate the test ports.

450	`OUTPUT 716;"ISOL; AVERFACT10;AVEROON;"`	Open the isolation calibration subsequence. Turn on averaging with an averaging factor of 10 for the isolation cal.
460	`DISP "MEASURING REVERSE ISOLATION"`	
470	`OUTPUT 716;"OPC?;REVI;"`	Measure reverse isolation.
480	`ENTER 716;Reply`	
490	`DISP "MEASURING FORWARD ISOLATION"`	
500	`OUTPUT 716;"OPC?;FWDI;"`	Measure forward isolation.
510	`ENTER 716;Reply`	
520	`OUTPUT 716;"ISOD; AVEROOFF;"`	Close the isolation calibration subsequence and turn off averaging.
530	`OUTPUT 717;"PG;"`	The `PG` command sent to the display clears the user graphics, removing the last prompt.
540	`DISP "COMPUTING CALIBRATION COEFFICIENTS"`	
550	`OUTPUT 716;"OPC?;SAV2;"`	
560	`ENTER 716;Reply`	Wait until the HP 8753C is done calculating the calibration coefficients before going on.
570	`DISP "DONE FULL 2-PORT CAL. CONNECT TEST DEVICE."`	
580	`OUTPUT 716;"MENUON;"`	The calibration is completed, so turn the soft key menu back on.
590	`END`	
600	`SUB Waitforkey(Lab$)`	This subroutine displays the passed message on the HP 8753C, and waits for the operator to press a key. It assumes that bit 6, User Request, of the event status register has been enabled.
610	`DISP Lab$`	First, display a message on the computer in case the operator has returned to the computer keyboard.
620	`OUTPUT 717;"PG;PU;PA390, 3600;PD;LB";Lab$;", PRESS ANY KEY*;"`	This statement writes on the HP 8753C's display. `PG` (page) clears old user graphics. `PU` (pen up) prevents anything from being drawn. `PA390,3600;` moves the logical pen to just above the message area on the display. `PD` (pen down) enables drawing. `LB` (label) writes the message on the display. The label command is terminated with an ETX symbol, which is **[CTRL] [C]** on the keyboard.
630	`CLEAR 716`	Clear the message line on the HP 8753C.
640	`OUTPUT 716;"ESR?;"`	Clear the latched User Request bit so that old key presses will not trigger a measurement.
650	`ENTER 716;Estat`	Now wait for a key press to be reported.
660	`Stat$zeSPOLL(716)`	

```
670    IF NOT BIT(Stat,5) THEN
       GOTO 660
680    OUTPUT 717;"PG;"
690    SUBEND
```

680: Clear the prompt from the display.

Running the program

The program assumes that the test ports being calibrated are type-N, port 1 being a female test port and port 2 being a male test port. The HP 85032B 50 ohm type-N calibration kit is to be used. The prompts appear just above the message line on the HP 8753C display. Pressing any key on the front panel of the HP 8753C continues the program and measures the standard. The operator has the option of omitting the isolation cal. If the isolation cal is performed, averaging is automatically employed to ensure a good calibration. The program will display a message when the measurement calibration is complete.

Before running the program, set up the desired instrument state. This program does not modify the instrument state in any way. Run the program, and connect the standards as prompted. When the standard is connected, press any key on the HP 8753C's front panel to measure it.

Data transfer from analyzer to computer

Using markers to obtain trace data at specific points

Trace information can be read out of the network analyzer in several ways. Data can be read off the trace selectively using the markers, or the entire trace can be read out. If only specific information such as a single point off the trace or the result of a marker search is needed, the marker output command can be used to read the information. If all the trace data is needed, seeExamples 3A thru 3C.

To get data off the trace using the marker, the marker first has to be put at the frequency desired. This is done with the marker commands. For example, execute:

```
OUTPUT 716;"MARK1 1.20 GHZ;"
```

This places marker one at 1.20 GHz. If the markers are in continuous mode, the marker value will be linearly interpolated from the two nearest points if 1.2000 GHz was not sampled. This interpolation can be prevented by putting the markers into discrete mode. The key sequence for this is **[LOCAL]** **[MKR]** ***[MARKER MODE MENU]*** ***[MARKERS:DISCRETE]***. To do it overHP-IB, execute:

```
OUTPUT 716;"MARKDISC;"
```

After executing this, note that the marker is may no longer be precisely on 1.20 GHz. (This depends on the start and stop frequencies).

Another way of using the markers is to let the network analyzer pick the stimulus value on the basis of one of the marker searches: max, min, target value, or bandwidths search. For example, execute:

```
OUTPUT 716;"SEAMAX;"
```

This executes a one-time trace search for the trace maximum, and puts the marker at that maximum. In order to continually update the search, turn tracking on. The key sequence is **[MKR FCTN]** ***[MKR SEARCH]*** ***[TRACKING]*** ***[SEARCH: MAX]***. To do it over HP-IB, execute:

```
OUTPUT 716;"TRACKON;SEAMAX;"
```

The trace maximum search will stay on this time, until search is turned off, tracking is turned off, or all markers are turned off. For example, execute:

```
OUTPUT 716;"MARKOFF;"
```

Marker data is read out with the command OUTPMARK. This command causes the network analyzer to transmit three numbers: marker value 1, marker value 2, and marker stimulus value. In this case we get the log magnitude at marker 1, zero, and the marker frequency. See Table 1 for all the different possibilities for values one and two. The third value is frequency in this case, but it could have been time as in time domain (option 010 only) or CW time.

Type **SCRATCH**, press **[EXECUTE]**, type **EDIT**, press **[EXECUTE]**, and then type in the following program:

10	`OUTPUT 716;"SEAMIN; OUTPMARK;"`	Have the network analyzer search out the trace minimum, and then output the marker values at that point.
20	`ENTER 716;Val1,Val2,Stim`	Read marker value 1, marker value 2, and the stimulus value.
30	`DISP Val1,Val2,Stim`	Display the values.
40	`END`	

Run the program. The values displayed by the computer should agree with the marker values displayed on the network analzyer, except that the second value displayed by the computer will be meaningless in phase and log mag formats. To see the possibilities for different values, run the program three times: once in log magnitude format, once in phase format, and once in Smith chart format. To change display format, press **[LOCAL] [FORMAT]** and then select the desired format.

Trace transfer

Getting trace data out of the network analyzer with a 200/300 series computer can be broken down into three steps:

1. Setting up the receive array.
2. Telling the network analyzer to transmit the data.
3. Accepting the transferred data.

Data inside the network analyzer is always stored in pairs, to accommodate real/imaginary pairs, for each data point. Hence, the receiving array has to be two elements wide, and as deep as the number of points. This memory space for this array must be declared before any data is to be transfered from the network analyzer to the computer.

The network analyzer can transmit data over HP-IB in five different formats. The type of format affects what kind of data array is declared (real or integer), since the format determines what type of data is transferred. Examples for data transfers using different formats are given below. The first, Example 3A, illustrates the basic transfer using form 4, an ASCII transfer. For more information on the various data formats, see the section entitled `Data Formats`. For information on the various types of data that can be obtained (raw data, corrected data and so on), see the section entitled `Data Levels`.

Note that Example 9, *Reading disk files into a computer*, allows the operator to access disk files from a computer.

Table 1. Units as a Function of Display Format

DISPLAY FORMAT	MARKER MODE	OUTPMARK value1, value2	OUTPFORM value1, value2	MARKET READOUT value, aux value**
LOG MAG		dB,*	dB,*	dB,*
PHASE		degrees,*	degrees,	degrees,*
DELAY		seconds,*	seconds,*	seconds,*
SMITH CHART	LIN MKR	lin mag, degrees	real, imag	lin mag, degrees
	LOG MKR	dB, degrees	"	dB, degrees
	Re/Im	real, imag	"	real, imag
	R + jX	real, imag ohms	"	real, imag ohms
	G + jB	real, imag Siemens	"	real, imag Siemens
POLAR	LIN MKR	lin mag, degrees	real, imag	lin mag, degrees
	LOG MKR	dB, degrees	"	dB, degrees
	Re/Im	real, imag	"	real, imag
LIN MAG		lin mag,*	lin mag,*	lin mag,*
REAL		real,*	real,*	real,*
SWR		SWR,*	SWR,*	SWR,*

* Value not significant in this format, but is included in data transfers.

** The marker readout values are the marker values displayed in the upper left hand corner of the display. They also correspond to the value and aux value associated with the fixed marker.

Example 3A: Data transfer using form 4 (ASCII transfer)

As detailed in the `HP-IB Quick Reference`, when form 4 is used, each number is sent as a 24 character string, each character being a digit, sign, or decimal point. Since there are two numbers per point, a 201 point transfer in form 4 takes 9,648 bytes. An example simple data transfer using form 4, an ASCII data transfer is shown in this program.

This example program is stored on the Example Programs disk as **IPG3A**.

10	`ABORT 7`	
20	`CLEAR 716`	Prepare for HP-IB control.
30	`OUTPUT 716;"PRES;"`	Preset the analyzer.
40	`DIM Dat(1:11,1:2)`	This line sets up an array to receive the data. The `ENTER 716;Dat(*)` statement in line 60 fills the array `Dat` automatically, changing the second subscript fastest. Since the network analyzer transmits the data as ordered pairs, we make the second dimension two so that the pairs will be properly grouped. The number of points will be set to 11, so we know to make the first dimension 11.
50	`OUTPUT 716;"POIN 11; SING; FORM4;OUTPFORM;"`	Set the number of points, tell the network analyzer to use ASCII transfer format, and request the formatted trace data. Frequency information is not included in the transfer.
60	`ENTER 716;Dat(*)`	The computer takes the data from the instrument and puts it in the receiving array. By specifying `Dat(*)`, we have told the enter statement to fill every location in the array.
70	`DISP DAT(1,1),DAT(1,2)`	This line checks the first data point received. The data is in the current network analyzer display format: see Table 1 for the contents of the array as a function of display format.
80	`END`	

Running the program

The first number of the result is a trace value in dB, and the second is zero. Put a marker at 300 kHz, which was the first point transmitted, to see that the values displayed by the computer agree with the network analyzer. No matter how many digits are displayed, the network analyzer is specified to measure magnitude to a resolution of .001 dB, phase to a resolution of .01 degrees, and group delay to a resolution of .01 psec.

Changing the display format will change the data sent with the `OUTPFORM` transfer. See Table 1 for a list of what data is provided with what formats. The data from `OUTPFORM` reflects all the post processing such as time domain, gating, electrical delay, trace math, and smoothing. If time domain (option 010 only) is on, operation is limited to 201 points in the lowpass mode.

Relating the data from a linear frequency sweep to frequency can be done by interrogating the start frequency, the frequency span, and the number of points. The frequency of point N in a linear frequency sweep is just:

F=Start_frequency +(N–1) X Span/(Points–1)

It is possible to read the frequencies directly out of the instrument with the `OUTPLIML` command. `OUTPLIML` reports the limit test results by transmitting the stimulus point tested, a number indicating the limit test results, and the upper and lower limits at that stimulus point, if available. The number indicating the limit results is a —1 for no test, 0 for fail, and 1 for pass. If there are no limits available, the network analyzer transmits zeros.

For this example, throw away the limit test information and keep the stimulus information. Edit line 40 to read:

```
40      DIM Dat(1:11,1:2), Stim(1:11)
```

And type in:

70	`OUTPUT 716;"OUTPLIML;"`	Request the limit test results.
80	`FOR I=1 TO 11`	Loop 11 times to read in all 11 data points.
90	`ENTER 716;Stim(I), Reslt,Upr,Lwr`	Read the stimulus values in, throw the rest away. Because we are not loading the data into a single array, it is necessary to loop and read every point.
100	`PRINT Stim(I),Dat(I,1), Dat(I,2)`	Print the data value and stimulus value.
110	`NEXT I`	
120	`DISP Reslt,Upr,Lwr`	Show what the last limit test result was, just to see what came out.
130	`END`	

Running this program will print out all the trace data and the stimulus values. Put the instrument into a log frequency sweep by pressing **[LOCAL] [MENU]** ***[SWEEP TYPE MENU] [LOG FREQ]***, and run the program again. If you define a list frequency table with 11 points, this program will still show the sampled frequencies. If you define a limit test table, `Reslt` will hold the limit test results.

Data levels

Different levels of data can be read out of the instrument (see Figure 2). There is available:

- Raw data. The basic measurement data, reflecting the stimulus parameters, IF averaging, and IF bandwidth. If a full 2-port measurement calibration is on, there are four raw arrays kept: one for each raw S-parameter. The data is read out with the commands `OUTPRAW1`, `OUTPRAW2`, `OUTPRAW3`, `OUTPRAW4`. Normally, only raw 1 is available, and it holds the current parameter. If a 2-port calibration is on, the four arrays refer to S_{11}, S_{21}, S_{12}, and S_{22} respectively. This data is in real/imaginary pairs.
- Error Corrected data. This is the raw data with error correction applied. The array is for the currently measured parameter, and is in real/imaginary pairs. The error corrected data is read out with `OUTPDATA`. `OUTPMEMO` reads the trace memory if available, which is also error corrected. Neither raw nor error corrected data reflect such post-processing functions as electrical delay offset, trace math, or time domain gating.
- Formatted data. This is the array of data being displayed. It reflects all post-processing functions such as electrical delay or time domain, and the units of the array read out depends on the current display format. See Table 1 for the various units as a function of display format.
- Calibration coefficients. The results of a calibration are arrays of calibration coefficients which are used in the error correction routines. Each array corresponds to a specific error term in the error model. The `HP-IB Quick Reference` details which error coefficients are used for specific calibration types, and which arrays those coefficients are to be found in. Not all calibration types use all 12 arrays. The data is stored as real/imaginary pairs.

Formatted data is generally the most useful, being the same information seen on the display. However, if the post processing is not necessary, as may be the case with smoothing, error corrected data is more desirable. Error corrected data also gives you the opportunity to put the data into the instrument and apply post-processing at a later time.

As an example of error corrected data, change line 50 to:

```
50      OUTPUT 716;"POIN 11; SING; FORM4; OUTPDATA;"
```

Running the program now displays real and imaginary trace data, regardless of what display format is currently being used. Select the real display format to verify that the data is the real portion.

Data formats

The network analyzer can transmit data over HP-IB in four different formats. Until now, we have been using form 4, an ASCII data transfer. Another option is to use form 3, which is the IEEE 64 bit floating point format. In this mode, each number takes only 8 bytes instead of 24. This means that a 201 point transfer takes only 3,216 bytes. Data is stored internally in the 200/300 series computer with the IEEE 64 bit floating point format, eliminating the need for any reformatting by the computer.

MS-DOS® personal computer format

Use form 5 to transfer data to an MS-DOS® PC. This mode is a modification of IEEE 32 bit floating point format with the byte order reversed. Form 5 also has a four byte header which must be read in so that data order is maintained. In this mode, an MS-DOS® PC can store data internally without reformatting it.

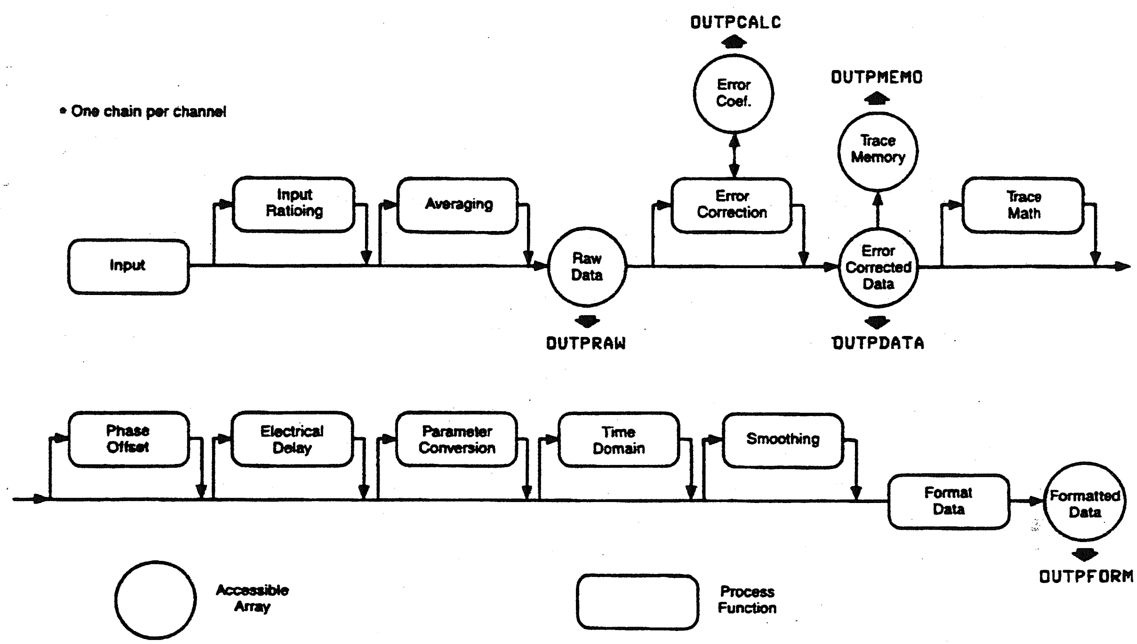


Figure 2. Data processing chain

Example 3B: Data transfer using form 3 (IEEE 64 bit floating point format)

This program illustrates data transfer using form 3, in which data is transmitted in the IEEE 64 bit floating point format.

To use form 3, the computer is told to stop formatting the incoming data with the `ENTER` statement. This is done by defining an I/O path with formatting off. Form 3 also has a four byte header to deal with. The first two bytes are the ASCII characters "#A" that indicate that a fixed length block transfer follows, and the next two bytes form an integer containing number of bytes in the block to follow. The header must read in so that data order is maintained.

This example program is stored on the Example Programs disk as **IPG3B**.

10	`ABORT 7`	
20	`CLEAR 716`	Prepare for HP-IB control.
30	`DIM Dat(1:201,1:2)`	As before, prepare the receiving array.
40	`INTEGER Hdr,Lgth`	Since an integer takes two bytes, `Hdr` and `Lgth` will take care of the four byte header. `Lgth` will hold the number of bytes in the data block.
50	`ASSIGN @Dt TO 716;FORMAT OFF`	This statement defines a data I/O path with ASCII formatting off. The I/O path points to the network analyzer, and can be used to read or write data to the instrument, as long as that data is in binary rather than ASCII format.
60	`OUTPUT 716;"SING; FORM3;OUTPFORM;"`	The analyzer is told to output formatted data using form 3.
70	`ENTER @Dt;Hdr,Lgth,Dat(*)`	The data is read in much as before, but the I/O path has format off to accept the binary data from form 3. The network analyzer and the computer must be in agreement as to the format of the data being transmitted.
80	`DISP Lgth,Dat(1,1),Dat(1,2)`	
90	`END`	

Running the program

Preset the instrument and run the program. The computer displays 3,216 and the trace values at 300 kHz. The number 3,216 comes from 201 points, 2 values per point, 8 bytes per value. This transfer is more than twice as fast as a form 4 transfer.

To illustrate a point, go to the instrument and press **[LOCAL]** **[MENU]** ***[NUMBER of POINTS]***, and key in **101** **[x1]**. Now run the program again: a BASIC error will be generated because the network analyzer ran out of data to transmit before the variable list was full.

Go to the instrument again, and this time change the number of points to 401. Running the program again does not generate an error, but not all of the data was read in. The network analyzer is still waiting to transmit data, but the program has not been designed to detect the situation.

As illustrated above, it is imperative that the receiving array be correctly dimensioned. There are two things that assure correct dimensions. First, the number of points is readily available through `POIN?` or through the header that precedes forms 1, 2 and 3. Second, BASIC allows dimensioning, redimensioning, allocating, and deallocating statements anywhere in a program. We can take advantage of this in simple programs to wait until we know how many points to expect before we dimension.

BASIC offers two options to those who want to dimension an array with a variable expression, such as the number of points in the sweep. One is the REDIM statement, available with AP2_1 or the MAT binary, which redimensions a given array to any size less than or equal to its originally dimensioned size. The other option is to ALLOCATE the array just before using it, and DEALLOCATE when it's no longer needed. ALLOCATE works exactly like DIM, except that when you deallocate, the memory space is returned to general use and you can re–use the variable name. All of the following examples use ALLOCATE.

For example, delete line 30 and type in the following lines over the last program:

70	`ENTER @Dt;Hdr,Lgth`	
80	`ALLOCATE Dat(1:Lgth/16,1:2)`	This guarantees that the receiving array is the correct size. In form 3, each number is 8 bytes, and there are two numbers per point, so we divide `Lgth` by 16 to get number of points.
90	`ENTER @Dt;Dat(*)`	
100	`DISP Dat(Lgth/16,1)`	Display the last number read in.
110	`END`	

Set the number of points to 51 and run the program: this time no errors are generated. Set the number of points to 401, and run the program again. Move a marker to the last point on the trace, and check to see that the last point read in was the last point on the trace, as expected.

There are two other formats available. Form 2 is not used with 200/300 computers, and form 1 is a special high speed transfer. Form 1 is a condensed transfer format that is useful if data is being transferred out of the network analyzer for direct storage and later re–transmission to the network analyzer. Example 3C gives an example of a data transfer using form 1.

Example 3C: Data transfer using form 1 (network analyzer internal format)

In form 1, each data point is sent out as it is stored inside the network analyzer, in a six byte binary string. It is a very fast transfer, using only 1206 bytes to transfer 201 points, but it is difficult to decode. (Real/imaginary data uses the first two bytes for the imaginary fraction mantissa, the middle two bytes for the real fraction mantissa, the fifth byte is used for additional resolution when transferring raw data, and the last byte as the common power of two). The data could be recombined and displayed in the computer, but this requires reformatting time.

In this example, we use form 1 to get data to store on disk. Before running this program, be sure that the mass storage device is a disk drive with a formatted disk in it. We also introduce a method of loading data back into the network analyzer. For most `OUTPxxxx` commands, there is a corresponding `INPUxxxx` command, and here we take advantage of that to load error corrected data back into the instrument.

This example program is stored on the Example Programs disk as **IPG3C**.

10	`ABORT 7`	
20	`CLEAR 716`	Prepare for HP–IB control.
30	`INTEGER Hdr,Lgth`	Set up to integers to take the header, the same as with form 3.
40	`ASSIGN @Dt TO 716;FORMAT OFF`	
50	`OUTPUT 716;"SING;FORM1; OUTPDATA;"`	Have the network analyzer take a sweep, and prepare to transmit the trace data to the computer.
60	`ENTER @Dt;Hdr,Lgth`	
70	`CREATE BDAT "TESTDATA",1, Lgth+4`	This statement creates a disk file to store the form 1 data in. It creates a binary data file name TESTDATA. The file is 1 record long, using a record length of Lgth + 4 bytes. The extra 4 bytes are for the header. This example will not run unless `MASS STORAGE IS` points to a disk drive with a formatted disk it, and that disk cannot have a file named TESTDATA on it.
80	`ASSIGN @Disc TO "TESTDATA"`	This statement creates a data I/O path pointing to the file TESTDATA.
90	`ALLOCATE INTEGER Dat (1:Lgth/6,1:3)`	Create an integer receiving array. There are six bytes per point in form 1, so allocating 3 integers per point will hold the data correctly, since an integer is two bytes.
100	`ENTER @Dt;Dat(*)`	The data is received much as before.
110	`OUTPUT @Disc;Hdr,Lgth, Dat(*)`	Write the data to the disk drive.
120	`INPUT "CHANGE TRACE AND HIT RETURN",Dum$`	Disconnect the test device, and take a sweep. Read the data off the disk, and put it back in the instrument.
130	`OUTPUT 716;"SING;"`	Take one sweep and hold.
140	`ASSIGN @Disc TO "TESTDATA"`	Re–establish the data path. This is necessary to begin reading data from the start of the file, rather than the end of the file where the file pointer was left by line 110.

```
150  ENTER @Disc;Hdr,Lgth, Dat(*)     Get the information.
160  OUTPUT 716;"INPUDATA"
170  OUTPUT @Dt;Hdr,Lgth,Dat(*)       And copy it out to the network analyzer.
180  ASSIGN @Disc TO *                Close the file.
190  DEALLOCATE Dat(*)                Release the memory for the data array.
200  PURGE "TESTDATA"                 And purge the data file.
210  END
```

Running the program

A data file is stored to disk during program execution. Either remove the write-protection from the Example Programs disk or install a blank, formatted data disk. Preset the network analyzer, and run the program. When the program pauses press **[LOCAL]**, change the trace, and press **[RETURN]**. When the data is reloaded into the network analyzer, it will be formatted and displayed as the current trace. This form of data transfer is faster than the transfer using form 3.

Advanced Programming Examples

Using list frequency mode

The network analyzer takes data points spaced at regular intervals across the overall frequency range of the measurement. For a 2 GHz frequency span using 201 points, data will be taken at intervals of 10 MHz. The list frequency mode lets you select the specific points or frequency spacing between points at which measurements are to be made. This allows flexibility in setting up tests to ensure efficient device performance. Sampling specific points reduces measurement time since additional time is not spent measuring device performance at frequencies not needed.

The following examples illustrate the use of the network analyzer's list frequency mode to perform arbitrary frequency testing. Example 4A lets you construct a table of list frequency segments which is then loaded into the network analyzer's list frequency table. Each segment stipulates a start and stop frequency, and the number of data points to be taken over that frequency range. Example 4B lets you select a specific segment to "zoom–in" on. A single instrument can be ready to measure several different devices, each with its own frequency range, using a single calibration performed with all of the segments active. When a specific device is connected, you select the appropriate segment for that device. The list frequency segments can be overlapped, but the number of points in all the segments must not exceed 1632 points.

Example 4A: Setting up a list frequency sweep

This example shows how to create a list frequency table and transmit it to the network analyzer.

The command sequence for entering a list frequency table imitates the key sequence followed when entering a table from the front panel: there is a command for every key press. Editing a segment is also the same as the key sequence, but the network analyzer automatically reorders each edited segment in order of increasing start frequency.

The list frequency table is also carried as part of the learn string. While it cannot be modified as part of the learn string, it can easily be stored and recalled.

This example takes advantage of the computer's capabilities to simplify creating, adding to, and editing the table. The table is entered and completely edited before being transmitted to the network analyzer. To simplify the programming task, options such as entering center/span or step size are not included. For information on reading list frequency data out of the network analyzer, see the section *Data transfer from analyzer to computer.*

This program is stored on the Example Programs disk as **IPG4A** .

10	`ABORT 7`	
20	`CLEAR 716`	Prepare the network analyzer for HP–IB control.
30	`OUTPUT 716;"EDITLIST;"`	Activate the frequency list edit mode.
40	`OUTPUT 716;"CLEL;"`	Delete any existing segments.
50	`INPUT "Number of segments?",Numb`	Find out how many segments to expect.
60	`ALLOCATE Table(1:Numb,1:3)`	Create a table to hold the segments. Keep start frequency, stop frequency, and number of points.
70	`PRINTER IS 1`	Make sure we print on the screen.
80	`OUTPUT 2;CHR$(255)&"K";`	Clear the screen.

90	`PRINT USING "10A,10A,10A, 20A"; "SEGMENT", "START(MHZ)", "STOP(MHZ)", "NUMBER OF POINTS"`	Print the table header.
100	`FOR I=1 TO Numb`	Read in each segment.
110	`GOSUB Loadpoin`	(line 300) reads in the start frequency, stop frequency, and number of points for segment **I**. Since `Loadpoin` is a subroutine, **I** is used as a global variable.
120	`NEXT I`	
130	`LOOP`	Use the `LOOP, EXIT IF, END LOOP` structure to loop and edit the table until editing is no longer needed. This structure sets up a loop with the exit point in the middle of the loop rather than at the beginning (as with `WHILE, END WHILE`), or at the end (as with `REPEAT, UNTIL`).
140	`INPUT "DO YOU WANT TO EDIT? Y OR N",An$`	Edit the table. Editing is re-entering the entire segment. The old segment values are left in place if return is pressed without typing anything.
150	`EXIT IF An$="N"`	Exit the edit loop if editing is finished. Execution is continued at line 210.
160	`INPUT "ENTRY NUMBER?",I`	For editing, get the entry number.
170	`GOSUB Loadpoin`	Re-enter the values.
180	`END LOOP`	
190	`OUTPUT 716;"EDITLIST"`	To begin the table, open the list frequency table for editing. The table must be empty, or these segments will be added on top of the old ones.
200	`FOR I=1 TO Numb`	Loop for each segment.
210	`OUTPUT 716;"SADD;STAR"; Table(I,1);"MHZ;"`	
220	`OUTPUT 716;"STOP"; Table(I,2);"MHZ;"`	
230	`OUTPUT 716;"POIN", Table(I,3),";"`	Enter the segment values.
240	`OUTPUT 716;"SDON;"`	Declare the segment done.
250	`NEXT I`	
260	`OUTPUT 716;"EDITDONE; LISFREQ;"`	Close the table, and turn on list frequency mode.
270	`STOP`	
280	`Loadpoin: !`	Enter in a segment.
290	`INPUT "START FREQUENCY? (MHZ)",Table(I,1)`	
300	`INPUT "STOP FREQUENCY? (MHZ)",Table(I,2)`	
310	`INPUT "NUMBER OF POINTS?",Table(I,3)`	Enter the segment values.

320	`IF Table(I,3)=1 THEN Table(I,2)=Table(I,1)`	If only one point in the segment, make the stop frequency equal to the start frequency to avoid ambiguity.
330	`PRINT TABXY(0,I+1);I; TAB(10);Table(I,1); TAB(20);Table(I,2); TAB(30),Table`	Print the segment out. If a segment is being edited, TABXY, will print over old segments.
340	`RETURN`	
350	`END`	

Running the program

The program displays the frequency list table as it is entered. During editing, the displayed table is updated as each line is edited. The table is not re-ordered. At the completion of editing, the table is entered into the network analyzer, and list frequency mode turned on. During editing, pressing **[RETURN]** leaves an entry at the old value.

Any segments already in the list frequency table in the network analyzer will be deleted by the program. If not wanted, delete lines 40 thru 60. New segments will write over the old ones.

Example 4B: Selecting a single segment from a table of segments

This example program shows how a single segment can be chosen to be the operating frequency range of the network analyzer, out of a table of segments. The program assumes that a list frequency table has already been entered into the network analyzer, either manually, or using the program in Example 4A, `Setting up a list frequency sweep`.

The program first loads the list frequency table into the computer by reading the start and stop frequencies of each segment, and the number of points for each segment. The segments' parameters are then displayed on the computer screen, and the user can choose which segment is to be used by the analyzer. Note that only one segment can be chosen at a time.

This program is stored on the Example Programs disk as **IPG4B**.

10	`ABORT 7`	
20	`CLEAR 716`	Prepare for HP–IB control
30	`PRINTER IS 1`	Make sure we print on the screen.
40	`OUTPUT 2;CHR$(255)$"K";`	Clear the screen.
50	`PRINT USING "10A,15A,15A,20A"; "SEGMENT","START(MHZ)", "STOP(MHZ)","NUMBER OF POINTS"`	Print out the table header.
60	`OUTPUT 716;"EDITLIST; SEDI30;SEDI;"`	Interrogate the number of the highest segment. This allows the program to determine the number of list frequency segments.
70	`ENTER 716;Numsegs`	Read the active parameter (segment number) into the variable `Numsegs`.
80	`ALLOCATE Table(1:Numsegs, 1:3)`	Create an array large enough to hold all the segment parameters.
90	`FOR I=1 to Numsegs`	This `FOR NEXT` loop calls the subroutine `Readlist` which reads in the segment parameters
100	`GOSUB Readlist`	
110	`NEXT I`	
120	`LOOP`	Use the `LOOP` structure to allow continuous selection of the desired segment to be measured.
130	`INPUT "SELECT SEGMENT NUMBER: (0 TO EXIT)",Segment`	
140	`EXIT IF Segment=0`	Allow the operator to exit the loop by entering 0 as the segment number.
150	`OUTPUT 716;"SSEG"; Segment;";EDITDONE;"`	The `SSEG` command causes the specific segment to become the new operating frequency range of the measurement.
160	`END LOOP`	
170	`OUTPUT 716;"ASEG;"`	When the loop is exited, resume operation using all list frequency segments. The `ASEG` command turns on all the segments.
180	`DISP "PROGRAM ENDED"`	
190	`STOP`	

200	`Readlist: !`	This subroutine reads out all the segment parameters.
210	`OUTPUT 716;"EDITLIST; SEDI;",I,";"`	Activate the `I`th segment.
220	`OUTPUT 716;"STAR; OUTPACTI;"`	Make the start frequency active, and output its value using the `OUTPACTI` command.
230	`ENTER 716;Table(I,1)`	Read the start frequency into the list table.
240	`OUTPUT 716;"STOP; OUTPACTI;"`	Make the stop frequency active, and output its value.
250	`ENTER 716;Table(I,2)`	Read the stop frequency value.
260	`OUTPUT 716;"POIN; OUTPACTI;"`	Make the number of points active, and output its value.
270	`ENTER 716;Table(I,3)`	Read the number of points.
280	`IF I=18 THEN INPUT "HIT RETURN FOR MORE",A$`	Stop printing when 17 segments have been listed on the display, this allows the operator to examine the first 17 segments before they are scrolled off the computer display by addition segments (remember, there are up to 30 segments).
290	`IMAGE 4D,6X,4D.6D,3X, 4D.6D,3X,4D`	Specify the print format and margins for the list frequency table.
300	`PRINT USING 290;I;Table (I,1)/1.E+9; Table(I,2)/ 1.E+9;Table(I,3)`	Print out the segment parameters for the `I`th segment.
310	`RETURN`	
320	`END`	

Running the program

The program will read the parameters for each list frequency segment from the network analyzer, and build a table containing all the segments. The parameters of each segment will be printed on the computer screen. If there are more than 17 segments, the program will pause. Press **[RETURN]** to see more segments. The maximum number of segments that can be read is 30 (which is the maximum number of segments that the network analyzer can hold). Use the computer's **[Prev]** and **[Next]**keys to scroll the list of segments back and forth if there are more than 17 segments.

After all the segments are displayed, the program will prompt for a specific segment to be used. Type in the number of the segment, and the network analyzer will then "zoom-in" on that segment. The program will continue looping, allowing continuous selection of different segments. To exit the loop, type **0**. This will restore all the segments (with the command ASEG), allowing the network analyzer to sweep all of the segments, and the program will terminate.

Using limit lines to perform PASS/FAIL tests

There are two steps to performing limit testing on the network analyzer under HP–IB control. First, limit specifications must be specified and loaded into the analyzer. Second, the limits are activated, the device is measured, and its performance to the specified limits is signaled by a pass or fail message on the network analyzer's display.

Example 5A illustrates the first step, setting up limits, and Example 5B performs the limit testing.

Example 5A: Setting up limit lines

This example shows how to create a limit table and transmit it to the network analyzer.

The command sequence for entering a limit table imitates the key sequence followed when entering a table from the front panel: there is a command for every key press. Editing a limit is also the same as the key sequence, but remember that the network analyzer automatically re–orders the table in order of increasing start frequency.

The limit table is also carried as part of the learn string. While it cannot be modified as part of the learn string, it can be stored and recalled with very little effort.

This example takes advantage of the computer's capabilities to simplify creating and editing the table. The table is entered and completely edited before being transmitted to the network analyzer. To simplify the programming task, options such as entering offsets are not included.

This program is stored as **IPG5A** on the Example Programs disk.

10	`ABORT 7`	
20	`CLEAR 716`	Prepare the network analyzer for HP–IB control.
30	`OUTPUT 716;"EDITLIML; CDEL;"`	Delete any existing limits.
40	`INPUT "Number of limits?",Numb`	Find out how many limits to expect.
50	`ALLOCATE Table(1:Numb,1:3)`	Create a table to hold the limits. It will contain stimulus value (frequency), upper limit value, and the lower limit value.
60	`ALLOCATE Limtype$(Numb)(2)`	Create a string array to indicate the limit types.
70	`PRINTER IS 1`	Make sure we print on the screen.
80	`OUTPUT 2;CHR$(255)&"K";`	Clear the screen.
90	`PRINT USING "10A,20A, 15A,20A";"SEG", "STIMULUS(MHZ)", "UPPER (dB)","LOWER (dB)","TYPE"`	Print the table header.
100	`FOR I=1 TO Numb`	Read in each segment.
110	`GOSUB Loadlimit`	Loadlimit (line 310) reads in the stimulus value (frequency), upper value, lower value, and the limit type for limit `I`. Since `Loadlimit` is a subroutine, `I` is used as a global variable.

120	`NEXT I`	Use the `LOOP`, `EXIT IF`, `END LOOP` structure to loop and edit the table until the operator indicates that editing is no longer desired. This structure sets up a loop with the exit point in the middle of the loop rather than at the beginning (as with `WHILE`, `END WHILE`), or at the end (as with `REPEAT`, `UNTIL`).
130	`LOOP`	
140	`INPUT "DO YOU WANT TO EDIT? Y OR N",An$`	Edit the table. Editing is re–entering the entire limit. The old limit values are left in place if return is pressed without typing anything.
150	`EXIT IF An$="N"`	Exit the edit loop if editing is finished. Execution is continued at line 190.
160	`INPUT "ENTRY NUMBER?",I`	For editing, get the entry number.
170	`GOSUB Loadlimit`	And have `Loadlimit` re–enter the values.
180	`END LOOP`	
190	`OUTPUT 716;"EDITLIML;"`	Begin the table entry by opening the limit table for editing. The limit table must be empty, or these limits will just be added on top of the old ones.
200	`FOR I=1 TO Numb`	Loop for each limit.
210	`OUTPUT 716;"SADD;LIMS"; Table(I,1);"MHZ;"`	Enter the stimulus value.
220	`OUTPUT 716;"LIMU";Table (I,2);"DB;"`	Enter the upper limit value.
230	`OUTPUT 716;"LIML",Table (I,3),"DB;"`	Enter the lower limit value.
240	`IF Limtype$(I)="FL" THEN OUTPUT 716;"LIMTFL;"`	Set flat limit type.
250	`IF Limtype$(I)="SL" THEN OUTPUT 716;"LIMTSL;"`	Set sloped limit type.
260	`IF Limtype$(I)="SP" THEN OUTPUT 716;"LIMTSP;"`	Set point limit type.
270	`OUTPUT 716;"SDON;"`	Declare the limit done.
280	`NEXT I`	
290	`OUTPUT 716;"EDITDONE; LIMILINEON;LIMITESTON;"`	Close the table, display the limits, and activate limit testing.
300	`STOP`	
310	`Loadlimit: !`	Enter in a segment.
320	`INPUT "STIMULUS VALUE? (MHZ)",Table(I,1)`	
330	`INPUT "UPPER LIMIT VALUE (DB)?",Table(I,2)`	
340	`INPUT "LOWER LIMIT VALUE (DB)?",Table(I,3)`	Enter the limit values.
350	`INPUT "LIMIT TYPE" (FL=FLAT, SL=SLOPED, SP=SINGLE POINT)",Limtype$(I)`	Enter the limit type.

```
360    PRINT TABXY(0,I+1);I;
       TAB(10); Table(I,1);
       TAB(30); Table(I,2);
       TAB(45),Table(I,3),
       TAB(67); Limtype$(I)
```

Print the limit values out. Because of the TABXY, this will print over old limits if a limit is being edited.

```
370    RETURN

380    END
```

Running the program

The program displays the limit table as it is entered. During editing, the displayed table is updated as each line is edited. The table is not reordered. When editing is done, the table is entered into the network analyzer, and limit testing mode turned on. During editing, pressing **[RETURN]** leaves an entry at the old value.

This example program will delete any existing limit lines before entering the new limits. If this is not wanted, omit lines 30 through 50.

Example 5B: Performing PASS/FAIL tests while tuning

The purpose of this example is to demonstrate the use of the limit/search fail bits in event status register B, to determine whether a device passes the specified limits. Limits can be entered manually, or using the Example 5A.

The limit/search fail bits are set and latched when limit testing or a marker search fails. There are four bits, one for each channel for both limit testing and marker search. Their purpose is to allow the computer to determine whether the test/search just executed was successful. The sequence of their use is to clear event status register B, trigger the limit test or marker search, and then check the appropriate fail bit.

In the case of limit testing, the best way to trigger the limit test is to trigger a single sweep. By the time the `SING` command finishes, limit testing will have occurred. A second consideration when dealing with limit testing is that if the device is tuned during the sweep, it may be tuned into and then out of limit, causing a limit test pass when the device is not in fact within limits.

In the case of the marker searches (max, min, target, and widths), outputting marker or bandwidth values automatically triggers any related searches. Hence, all that is needed is to check the fail bit after reading the data.

In this example, the requirement that several sweeps in a row must pass is used in order to give confidence that the limit test pass was not extraneous due to the device settling or the operator tuning during the sweep. Upon running the program, the number of passed sweeps for qualification is entered. For very slow sweeps, a small number of sweeps such as two is appropriate. For very fast sweeps, where the device needs time to settle after tuning and the operator needs time to get away from the device, as many sweeps as six or more sweeps might be appropriate.

A limit test table can be entered over HP-IB: the sequence is very similar to that used in entering a list frequency table and is shown in Example 5A. The manual sequence is closely followed.

This program is stored under **IPG5B** on the Example Programs disk.

10	`ABORT 7`	
20	`CLEAR 716`	Prepare the network analyzer for remote control.
30	`INPUT "Number of consecutive passed sweeps for qualification?",Qual`	Find out how many sweeps must pass before the device is considered to have passed the limit test.
40	`DISP "TUNE DEVICE"`	Tell operator to begin tuning.
50	`Reap=0`	Reap is a counter holding how many sweeps have passed the limit test.
60	`OUTPUT 716;"OPC?;SING;"`	Take a sweep. When it is done, limit test will have occurred.
70	`ENTER 716;Reply`	Wait for the end of the sweep.
80	`OUTPUT 716;"ESB?;"`	Check to see if the fail bit is set.
90	`ENTER 716;Estat`	
100	`IF BIT(Estat,4) THEN`	If the fail bit for channel one is set, reset the number of sweeps passed counter.
110	`IF Reap<$GTO THEN BEEP 1200,.05`	If sweeps had been passing, warn the operator that the device is now failing.
120	`Reap=0`	
130	`GOTO 40`	

140	`END IF`	If the fail bit was not set, tell the operator.
150	`BEEP 2500,.01`	
160	`Reap=Reap+1`	Increment the sweeps passed counter.
170	`DISP "STOP TUNING"`	Encourage the operator to stop tuning the device.
180	`IF Reap<Qual THEN GOTO 60`	If not enough sweeps have passed, loop.
190	`DISP "DEVICE PASSED!"`	The device has passed.
200	`FOR I=1 TO 10`	Warble, telling the operator the device has passed, using an audible signal.
210	`BEEP 1000,.05`	
220	`BEEP 2000,.01`	
230	`NEXT I`	
240	`INPUT "HIT RETURN FOR NEXT DEVICE",Dum$`	Wait for the next device.
250	`GOTO 40`	
260	`END`	

Running the program

Set up a limit table on channel 1 for a specific device either manually, or using the program in Example 5A. Run the program, and enter the number of passed sweeps desired for qualification. After entering the qualification number, connect the filter. When a sweep passes, the computer beeps. When enough sweeps in a row pass to qualify the device, the computer warbles and asks for a new device.

The program assumes a response calibration (thru calibration) or full 2-port calibration has been performed prior to running the program. Try causing the DUT to fail by loosening the cables connecting the DUT to the network analyzer, and then retightening them.

Storing and recalling instrument states

This example demonstrates ways of storing and recalling entire instrument states over HP-IB. The methods discussed are to use the learn string, and the computer to coordinate direct store/load of instrument states to disk.

Using the learn string is a quick way of saving the instrument state, but using direct disk access will automatically store calibrations, cal kits, and data along with the instrument state.

Example 6A: Using the learn string

The learn string is a fast and easy way to read an instrument state. The learn string includes all front panel settings, the limit table for each channel, and the list frequency table. The learn string is read out with `OUTPLEAS`, and put back into the instrument with `INPULEAS`. The string is in form 1, and is no longer than 3000 bytes long.

This example program is stored on the Example Programs disk as **IPG6A**.

10	`DIM State$(3000)`	Set up the receive string.
20	`OUTPUT 716;"OUTPLEAS;"`	Request the learn string.
30	`ENTER 716 USING "-K";State$`	Read in the learn string. Normally, the enter statement will terminate if a line feed is received, so `USING "-K"` is used, which allows termination only on End Or Identify.
40	`LOCAL 716`	Put the analyzer in LOCAL mode.
50	`INPUT "CHANGE STATE AND HIT RETURN",Dum$`	Give the operator a chance to modify the state or connect a new analyzer.
60	`OUTPUT 716;"INPULEAS"; State$`	Transmit the state back to the HP 8753B.
70	`DISP "INITIAL INSTRUMENT STATE RESTORED"`	
80	`END`	

Running the program

Run the program. When the program stops, change the instrument state and press **[RETURN]**. The network analyzer will return its original state.

When using a learn string from an HP 8753B, additional commands are needed because the "old" string is shorter. Therefore, you must tell the network analyzer when the transfer of the string has reached completion (sending EOI concurrently with the last byte of information).

The following program will output an HP 8753B learn string (previously obtained and put in State$) to an HP 8753C:

10	`OTHER CODE WHICH HAS State$`	
	`.` `.` `.`	
100	`EOI$=";"`	
120	`ASSIGN@Ana to 716; EOL EOI$ END`	This syncs EOI (End or Identify) with a semicolon as the last character in the string.
130	`OUTPUT@Ana; "INPULEAS";State$`	
140	`DISP "8753B Learn String State"`	
150	`END`	

Example 6B: Coordinating disk storage

To have the network analyzer store an instrument state on disk, specify the state name by titling a file using `TITFn`, then specify a `STORn` of that file, where `n` is the file number, 1 to 5. On receipt of the store command, the network analyzer will request active control. When control is received, the network analyzer will store the instrument state on disk as defined under the *[DEFINE STORE]* menu.

To have the network analyzer load a file from disk, specify the state name, and then request a `LOADn` of that file. The best way of learning what the register titles on the disk are, is to use the *[READ FILE TITLES]* under the **[RECALL]** key.

This example program is stored on the Example Programs disk as **IPG6B**.

10	`ABORT 7`	
20	`CLEAR 716`	Prepare the network analyzer for remote control.
30	`INPUT "STATE TITLE? PRESS RETURN",Nam$`	Get the name of the file to create.
40	`OUTPUT 716;"USEPASC;"`	Tell the network analyzer to use pass control mode.
50	`OUTPUT 716;"TITF1""";Nam$;""";STOR1;"`	Title register 1, and store it. The title must be preceded and followed by double quotation marks, and the only way to do that with an output statement is to use two sets of quotation marks: `""`.
60	`DISP "SAVING ON DISC"`	
70	`SEND 7;TALK 16 CMD 9`	Pass control to the network analyzer, assuming it has interpreted the `STOR1` command and set the request control bit.
80	`STATUS 7,6;Stat`	
90	`IF NOT BIT(Stat,6) THEN GOTO 80`	Wait for active control to return.
100	`INPUT "STATE STORED. HIT RETURN TO RECALL",Dum$`	
110	`INPUT "STATE TITLE?",Nam$`	Get the name of the file to read.
120	`OUTPUT 716;"TITF1""";Nam$;""";LOAD1;"`	Title register one, and request a load.
130	`DISP "READING DISC"`	
140	`SEND 7;TALK 16 CMD 9`	Pass control.
150	`STATUS 7,6;Stat`	
160	`IF NOT BIT(Stat,6) THEN GOTO 150`	Wait for control to return.
170	`DISP "DONE"`	The program is done, and the state has been loaded back into the instrument.
180	`END`	

Running the program

Put a formatted disk in the disk drive, and point the network analyzer's disk address, unit number, and volume number toward that drive. Run the example, and when the program pauses, change the instrument state so that a change will be noticeable. Pressing return will recall the state just stored, or a completely different state can be recalled.

Example 6C: Reading calibration data

This example demonstrates how to read measurement calibration data out of the network analyzer, how to put it back into the instrument, and how to determine which calibration is active.

The data used to perform measurement error correction is stored inside the network analyzer in up to twelve calibration coefficient arrays. Each array is a specific error coefficient, and is stored and transmitted as an error corrected data array: each point is a real/imaginary pair, and the number of points in the array is the same as the number of points in the sweep. The four data formats also apply to the transfer of calibration coefficient arrays. Appendix C, `Calibration`, of the `HP-IB Quick Reference` specifies where the calibration coefficients are stored for different calibration types.

A computer can read out the error coefficients using the commands `OUTPCALC01`, `OUTPCALC02`, ... `OUTPCALC12`. Each calibration type uses only as many arrays as needed, starting with array 1. Hence, it is necessary to know the type of calibration about to be read out: attempting to read an array not being used in the current calibration causes the "REQUESTED DATA NOT CURRENTLY AVAILABLE" warning.

A computer can also store calibration coefficients in the network analyzer. To do this, declare the type of calibration data about to be stored in the network analyzer just as if you were about to perform that calibration. Then, instead of calling up different classes, transfer the calibration coefficients using the `INPUCALCnn` commands. When all the coefficients are in the network analyzer, activate the calibration by issuing the mnemonic `SAVC`, and have the network analyzer take a sweep.

This example reads the calibration coefficients into a very large array, from which they can be examined, modified, stored, or put back into the instrument. If the data is to be directly stored onto disk, it is usually more efficient to use form 1 (network analyzer internal binary format), and to store each coefficient array as it is read in.

This program is stored on the Example Programs disk as **IPG6C**.

10	`ABORT 7`	
20	`CLEAR 716`	Prepare the network analyzer for HP-IB control.
30	`DATA "CALIRESP",1, "CALIRAI",2, "CALIS111",3`	
40	`DATA "CALIS221",3, "CALIFUL2",12`	
50	`DATA "NOOP",0`	Set up the data base of possible calibrations, and the number of arrays associated with each calibration.
60	`INTEGER Hdr,Lgth,I,J`	Define integers to hold the header, and to act as counters.
70	`ASSIGN @Dt TO 716;FORMAT OFF`	
80	`READ Calt$,Numb`	Get a calibration type and the number of associated arrays.
90	`IF Numb=0 THEN GOTO 360`	If correction was not on, stop the program.
100	`OUTPUT 716;Calt$;"?;"`	Interrogate the network analyzer to see if this calibration is active.
110	`ENTER 716;Active`	
120	`IF NOT Active THEN GOTO 80`	If the calibration was not active, loop.
130	`DISP Calt$,Numb`	Show the operator that we have found the calibration and number of arrays.
140	`OUTPUT 716;"FORM3;POIN?;"`	Find out how many points to expect.

150	`ENTER 716;Poin`	
160	`ALLOCATE Cal(1:Numb,1: Poin,1:2)`	Create a very large array to hold all the coefficients.
170	`FOR I=1 TO Numb`	Loop once for each calibration coefficient.
180	`OUTPUT 716 USING "K,ZZ";"OUTPCALC",I`	Request the calibration coefficient. The `K` transmits `OUTPCALC` literally, and `ZZ` transmits `I` as two digits, using a leading zero if needed.
190	`ENTER @Dt;Hdr,Lgth`	`Read the header.`
200	`FOR J=1 TO Poin`	
210	`ENTER @Dt; Cal(I,J,1),Cal(I,J,2)`	Since we are not filling the entire array, we have to read each point individually.
220	`NEXT J`	
230	`NEXT I`	The calibration data is now all in the computer.
240	`INPUT "HIT RETURN TO RE-TRANSMIT CALIBRATION", Dum$`	
250	`OUTPUT 716;Calt$,";"`	Begin the calibration retransmission by declaring what calibration type is about to be loaded.
260	`FOR I=1 TO Numb`	Now load each calibration coefficient.
270	`DISP "TRANSMITTING ARRAY: ",I`	
280	`OUTPUT 716 USING "K,ZZ"; "FORM3;INPUCALC",I`	
290	`OUTPUT @Dt;Hdr,Lgth`	
300	`FOR J=1 TO Poin`	
310	`OUTPUT @Dt;Cal(I,J,1),Cal(I,J,2)`	
320	`NEXT J`	
330	`NEXT I`	All of the calibration data has been loaded.
340	`OUTPUT 716;"SAVC;"`	End the sequence by activating the calibration.
350	`OUTPUT 716;"CONT;"`	Trigger a sweep so the calibration becomes active.
360	`DISP "DONE"`	
370	`END`	

Running the program

Before executing the program, perform a calibration.

The program is able to detect what calibration is active, and with that information it predicts how many arrays to read out. When all the arrays are inside the computer, the program prompts the user. At this point, turn calibration off, or perform a completely different calibration on the network analyzer. Then press continue on the computer, and the computer will reload the old calibration.

Note that the retransmitted calibration is associated with the current instrument state: the instrument has no way of knowing the original state associated with the calibration data. For this reason, it is recommended that the learn string be used to store the instrument state whenever calibration data is stored. See Example 6A, *Using the learn string*.

Miscellaneous Programming Examples

Controlling peripherals

The purpose of this section is to demonstrate how to coordinate printers, plotters, power meters, and disk drives with the network analyzer.

The network analyzer has three operating modes with respect to HP–IB, as set under the **[LOCAL]** menu. System controller mode is used when no computer is present. The other two modes allow the computer to coordinate certain actions: in talker/listener mode the computer can control the network analyzer, as well as coordinate plotting and printing, and in pass control mode the computer can pass active control to the network analyzer so that the network analyzer can plot, print, control a power meter, or load/store to disk. Peripheral control is the major difference between the two modes.

Note that the network analyzer assumes that the address of the computer is correctly stored in its HP–IB addresses menu under the ***[ADDRESS: CONTROLLER]*** entry. If this address is incorrect, control will not return to the computer. If control is passed to the network analyzer while it is in talker/listener mode, control will not return to the computer.

Example 7: Operation using pass control mode

If the network analyzer is in pass control mode and receives a command telling it to plot, print, control a power meter, or store/load to disk, it sets bit 1 in the event status register to indicate that it needs control of the bus. If the computer then uses the HP–IB control command to pass control to the network analyzer, the network analyzer will take control of the bus, and access the peripheral. When the network analyzer no longer needs control, it will pass it back to the computer. When performing a power meter cal over HP–IB, the network analyzer requests control at each measurement point in a sweep which is typically $ae3* the number of readings.

Control should not be passed to the network analyzer before it has set event status register bit 1, Request Active Control. If the network analyzer receives control before the bit is set, control is passed immediately back.

While the network analyzer has control, it is free to address devices to talk and listen as needed. The only functions denied it are the ability to assert the interface clear line (IFC), and the remote line (REN). These are reserved for the system controller. As active controller, the network analyzer can send messages to and read replies back from printers, plotters, and disk drives.

This example prints the display. It is stored on the Example Programs disk as **IPG7B**. The program could request a plot with `PLOT`, or a disk access with a command such as `REFT` (read file titles.)

10	`OUTPUT 716;"CLES;ESE2;"`	Clear the status reporting system, and enable the Request Active Control bit in the event status register.
20	`OUTPUT 716;"USEPASC; PRINALL;"`	Put the network analyzer in pass control mode, and request a print.
30	`Stat=SPOLL(716)`	Get the status byte of the network analyzer.
40	`IF NOT BIT(Stat,5) THEN GOTO 30`	If the network analyzer is not requesting control, loop and wait. If using color printer, use COLOP.
50	`SEND 7;TALK 16 CMD 9`	This is the bus command to pass active control to device 16. With BASIC 3.0 or higher, or 2.0 with extensions 2.1, the command PASS CONTROL 716 can be used instead.

`60 DISP "PRINTING"`	
`70 STATUS 7,6;Hpib`	To determine when the print is finished, watch for return of active control. The `STATUS` command loads the interface 7 (HP–IB) register 6, the computer's status with respect to HP–IB, into the variable `Hpib`. Bit 6 tells if the computer is the active controller: it will be set when the network analyzer returns control.
`80 IF NOT BIT(Hpib,6) THEN GOTO 70`	If control has not returned, loop and wait.
`90 DISP "DONE"`	Control has returned.
`100 END`	

Running the program

The network analyzer will briefly flash the message WAITING FOR CONTROL, before receiving control and making the print. The computer will display the PRINTING message.

When the print is complete, the network analyzer passes control back to the address stored as the controller address under the **[LOCAL]** ***[SET ADDRESSES]*** menu. The computer will detect the return of active control and exit the wait loop.

Because the program waits for the network analyzer's request for control, it can be used to respond to front panel requests as well. Delete `PRINALL;` from line 20, and run the program. Nothing will happen until you go to the front panel of the network analyzer and request a print, plot, or disk access. For example, press **[LOCAL]** **[COPY]** and ***[PRINT]***.

Example 8: Creating a user interface

This example shows how to create a custom user interface involving only the front panel keys and display of the network analyzer.

User graphics

The network analyzer's display can be treated as an HP–GL plotter. The BASIC graphics commands can be used to create a custom display. Some of the more useful commands are as follows. `VIEWPORT` defines what area of the display is to be plotted on. `WINDOW` allows you to specify the plotting units (i.e. how many units per axis) in the `VIEWPORT` defined area. `DRAW` draws lines from point to point. `MOVE` moves the logical pen without drawing anything. `GCLEAR` clears the graphics display area. `PEN` selects the line color.

All of the BASIC graphics statements are accepted. The LABEL statement is not recommended because it fills the display memory up very rapidly as opposed to when the HP–GL `LB` command is used. See the Waitforkey subroutine of Example 2A for an example of the `LB` command.

HP–GL (Hewlett–Packard Graphics Language) commands, such as the `LB` command mentioned above, can be directly sent to the network analyzer display with the `OUTPUT` statement. See Appendix D, *Display Graphics*, of the *HP–IB Quick Reference* for a list of the HP–GL commands accepted, and their functions.

Front panel control

It is possible to take over the front panel keys. The user request bit in the event status register is set whenever a front panel key is pressed or the knob is turned, whether the instrument is in remote or local mode. Each key has a number associated with it, as shown in Figure E.4, *Front Panel Keycodes* of the *HP–IB Quick Reference*. The number of the key last pressed can be read with the `KOR?` and the `OUTPKEY?` commands. With `KOR?`, a knob turn is reported as a negative number encoded with the number of counts turned. With `OUTPKEY?`, a knob turn is always reported as a negative one.

The keycode encoding with `KOR?` is as follows. Clockwise rotations are reported as numbers from —1 to —64, —1 being a very small rotation. Counter–clockwise rotations are reported as the numbers —32,767 to —32,703, —32,767 being a very small rotation. Hence, clockwise rotations don't need any decoding at all, and counter–clockwise rotations can be decoded by adding 32,768.

There are approximately 120 counts per knob rotation, and sign of the count depends on the direction the knob was turned.

This example uses the knob and the up and down keys on the network analyzer to position a grid on the display. Pressing **[ENTRY OFF]** on the network analyzer causes the computer to put a trace on the grid.

This example program is stored on the Example Programs disk as **IPG8**.

10	`INTEGER Hdr,Lgth,Keyc`	Declare variables to hold the header and the key code.
20	`ASSIGN @Dt TO 716;FORMAT OFF`	Define an IO path with formatting off, to receive the form 3 trace data for plotting.
30	`OUTPUT 716;"HOLD;AUTO;` `CLES;ESE 64;POIN?;"`	Prepare the instrument. `HOLD;AUTO;` freezes and scales the trace for plotting. `CLES;ESE 64;` clears the status reporting system and enables the User Request bit in the event status register. Lastly, `POIN?;` requests the number of points.

40	`ENTER 716;POIN`	Read in the number of points.
50	`GINIT`	Initialize the graphics functions in the computer.
60	`PLOTTER IS 717,"HPGL"`	Specify the network analyzer display as the plotting device.
70	`OUTPUT 717;"CS;SP3;"`	Turn off the measurement display and set the rectangle color to that of channel 2 data.
80	`Cx=55`	Initialize the x position of the center of the rectangle.
90	`Cy=60`	Initialize the y position of the center of the rectangle.
100	`S=20`	Set the size of the rectangle.
110	`REPEAT`	The `REPEAT`, `UNTIL` structure sets up a loop that keeps repeating until the condition specified in the `UNTIL` statement is found to be true. The condition is checked at the end of the loop. In this case, loop and redraw the rectangle until **[ENTRY OFF]** has been pressed.
120	`GCLEAR`	Clear the graphics area on the network analyzer.
130	`IF Cx$GT160 THEN Cx=160`	Prevent box from going off the screen.
140	`IF Cx<-17 THEN Cx=-17`	Note that these values are linked to the increments set in lines 270/310 and 320!
150	`IF Cy$GT115 THEN Cy=115`	
160	`IF Cy<-15 THEN Cy=-15`	
170	`VIEWPORT Cx-S,Cx+S,Cy-S,Cy+S`	Define the area of the rectangle, which will become the plotting area for the grid and trace.
180	`WINDOW 0,Poin-1,0,1`	Define the units along the edges of the rectangle. In this case, the horizontal edge has as many units as points in the sweep, and the vertical edge is simply unity.
190	`FRAME`	Draw the rectangle around the plotting area.
200	`Stat=SPOLL(716)`	Read the status byte.
210	`IF NOT BIT(Stat,5) THEN GOTO 200`	If bit 5 is not set, a key has not been pressed, so loop and wait.
220	`OUTPUT 716;"ESR?;"`	A key press has occurred, so read the event status register in order to clear the latched bit.
230	`ENTER 716;Estat`	Read in the register value, but do nothing with it.
240	`OUTPUT 716;"KOR?;"`	Now read in the key or knob count.
250	`ENTER 716;Keyc`	
260	`IF Keyc=26 THEN Cy=Cy+5`	Key 26 is the up key, so shift the rectangle up.
270	`IF Keyc=18 THEN Cy=Cy-5`	Key 18 is the down key, so shift the rectangle down.

280	`IF Keyc<0 THEN`	If the keycode was negative, then it is a knob count.
290	`Knb=Keyc`	Decode the knob count into the variable Knb.
300	`IF Knb<-64 THEN Knb=Knb+32768`	If the count is less then —64, add 32768 (2^{15}) to recover the knob count. If the count is more than —64, then no decoding is needed.
310	`x=Cx-Knb*3C`	Shift the rectangle according the knob count, multiplying the knob count to make the rectangle move farther.
320	`END IF`	
330	`UNTIL Keyc=34`	This is the end of the REPEAT, UNTIL structure. Leave the loop only when key 34, **[ENTRY OFF]** has been pressed.
340	`GRID (Poin-1)/10,.1`	**[ENTRY OFF]** has been pressed, so draw the grid and the trace. This statement draws a grid with 10 divisions on each axis.
350	`OUTPUT 717;"SP1;"`	Set the trace color to that of channel 1 data.
360	`OUTPUT 716;"FORM3;OUTPFORM;"`	Now get the trace data.
370	`ENTER @Dt;Hdr,Lgth`	Get the header information.
380	`ALLOCATE Dat(1:Poin,1:2)`	Define the receiving array.
390	`ENTER @Dt;Dat(*)`	And read in the data.
400	`OUTPUT 716;"SCAL?;"`	Instead of scaling the data in this program, interrogate the scale factor the network analyzer was using.
410	`ENTER 716;Scal`	
420	`OUTPUT 716;"REFV?;"`	Similarly, use the value at the reference position to decide where to draw the trace.
430	`ENTER 716;Ref`	
440	`OUTPUT 716;"REFP?;"`	Interrogate the current reference position being used.
450	`ENTER 716;Refp`	
460	`Bot=Rev-Refp*Scal`	Calculate the value of the bottom grid line.
470	`Full=10*Scal`	And define the full scale span across the grid.
480	`MOVE 0,(Dat(1,1)-Bot)/Full`	Go to the first point on the trace without drawing anything.
490	`FOR I=1 TO Poin-1`	And draw all the rest of the points in the trace.
500	`DRAW I,(Dat(I,1)-Bot)/Full`	
510	`NEXT I`	The trace is drawn, so end the program.
520	`END`	

Running the program

Set the instrument up to make a measurement. The network analyzer will not accept a graphics dump of a trace of greater than 1601 points.

Run the program, and go to the front panel of the network analyzer. The measurement display has been turned off, and there is a box on the screen. The knob moves the box left and right, and the up/down keys move the box up and down. When you are satisfied with the position of the box, press **[ENTRY OFF]**. The computer will fill the box with a grid, and plot the current measurement data on the grid.

Transferring disk data files

An external disk drive is often used to store data files in addition to instrument states (see Example 6B). Instrument states, graphics, data trace, calibration data, and memory trace files can be stored on disk. The file name is then appended with up to two characters to indicate what is in the file. For example, if channel 2 error-corrected data is saved to disk as DEVICE, the actual error-corrected data would be stored in DEVICED2. As with all data files stored on disk, they are stored in form 3. See Appendix E.3: *Disk file names* in the *HP-IB Quick Reference* for a complete list of the types of files saved to disk and the corresponding appendages to file names.

Example 9: Reading data files into a computer

This example demonstrates how to recall a specific disk file into a computer. First, EXT-MADTAON defines the storage of the current trace as error-corrected data. After the file is stored to disk, the computer reads the error-corrected data into an array. The program can easily be modified to read and transfer raw data, memory traces, and formatted data.

10	`ABORT 7`	
20	`CLEAR 716`	Prepare the network analyzer for remote control.
30	`INPUT "STATE TITLE?",Nam$`	Get the name of the file to create.
40	`OUTPUT 716;"USEPASC;"`	Tell the network analyzer to use pass control mode.
50	`OUTPUT 716;"TITF1""";` `Nam$;""";EXTMDATAON; STOR1;"`	Title register 1, and store the instrument state and error-corrected data. The title must be preceded and followed by double quotation marks. The only way to do this within an output statement is to use two sets of quotation marks: " ".
60	`DISP "SAVING ON DISC"`	
70	`SEND 7;TALK 16 CMD 9`	Pass control to the network analyzer, assuming it has interpreted the STOR 1 command and set the request control bit.
80	`STATUS 7,6;Stat`	
90	`IF NOT BIT(Stat,6) THEN GOTO 80`	Wait for active control to return.
100	`DISP "READING DATA INTO Disc_dat ARRAY"`	
110	`ASSIGN @Dt TO Nam$&"D1";FORMAT OFF`	This statement defines an I/O path with ASCII formatting off. The I/O path points to the chosen error-corrected data file, and can be used to read or write data from the file, since it is in binary rather than ASCII format.
120	`ALLOCATE Disc$zedat(1:201,1:2)`	Allocate an array for a 201 point data trace. Real and imaginary pairs will be transferred for each data point.
130	`ENTER @Dt;Disc_dat(*)`	The computer takes the data from disk and transfers it into the receiving array. By specifying Disc_dat (*), the ENTER statement will fill every location in the array.
140	`ASSIGN @Dt TO *`	Close the I/O path.

150	`DISP Disc_dat(1,1), Disc_dat(1,2)`	Show the first real imaginary pair.
160	`END`	End program execution.

Running the program

A data file is stored to disk during program execution. Either remove the write–protection from the Example Programs disk or install a blank, formatted data disk. Perform a measurement calibration with 201 points. Connect a test device and run the program. The first/real imaginary pair will be displayed. Place a marker at the beginning of the trace and look at both real and imaginary formats to verify this point.

Appendix A: Status Reporting

The network analyzer has a status reporting mechanism that gives information about specific functions and events inside the network analyzer. The status byte is an 8 bit register with each bit summarizing the state of one aspect of the instrument. For example, the error queue summary bit will always be set if there are any errors in the queue. The value of the status byte can be read with the `SPOLL(716)` statement. This command does not automatically put the instrument in remote mode, thus giving the operator access to the network analyzer front panel functions. The status byte can also be read by sending the command `OUTPSTAT`. Reading the status byte does not affect its value. The sequencing bit can be set by the operator during execution of a test sequence.

The status byte summarizes the error queue, as mentioned before. It also summarizes two event status registers that monitor specific conditions inside the instrument. The status byte also has bits that are set when the front panel preset key has been pressed, when the instrument is issuing a service request over HP–IB, and when the network analyzer has data to send out over HP–IB. See Figure A.1 for a definition of the status registers.

Example A1: Using the error queue

The error queue holds up to 20 instrument errors and warnings in the order that they occurred. Each time the network analyzer detects an error condition and displays a message on the CRT, it also puts the error in the error queue. If there are any errors in the queue, bit 3 of the status byte will be set. The errors can be read from the queue with the `OUTPERRO` command, which causes the network analyzer to transmit the error number and the error message of the oldest error in the queue.

This example program is stored on the Example Programs disk as **IPGA1**.

10	`DIM Err$(50)`	Prepare a string to hold the error message.
20	`Stat=SPOLL(716)`	Use the serial poll statement to read the status byte into the variable `Stat`. Serial poll is an HP–IB function dedicated specifically to getting the status byte of an instrument quickly, and does not cause the network analyzer to go into remote.
30	`IF NOT BIT(Stat,3) THEN GOTO 20`	If the error queue summary bit is not set, loop until it is set.
40	`OUTPUT 716;"OUTPERRO;"`	If the error queue has something in it, we instruct the network analyzer to output the error number and the error message. This communication with the network analyzer will put it in remote mode.
50	`ENTER 716;Err,Err$`	Err holds the error number, `Err$` the error message.
60	`PRINT Err,Err$`	
70	`LOCAL 716`	Return the network analyzer to local mode so that the front panel is available to the operator.
80	`BEEP 600,.2`	Give an audible signal that there is a problem.
90	`GOTO 20`	
100	`END`	

Running the program

Preset the network analyzer and run the program. Nothing should happen at first. To get something to happen, press a blank softkey. The message "CAUTION: INVALID KEY" will appear on the network analyzer, the computer will beep and print two lines. The first line will be the invalid key error, and the second message will be the "NO ERRORS" message. Hence, to clean the error queue, you can either loop until the no errors message is received, or until the bit in the status register is cleared. In this case, we wait until the status bit is clear. Note that all through this, the front panel of the network analyzer is in local mode.

Because the error queue will keep up to 20 errors until either all the errors are read out or the instrument is preset, it is important to clear out the error queue whenever errors are detected so that old errors are not associated with the current instrument state.

Not all messages displayed by the network analyzer are put in the error queue: operator prompts and cautions are not included.

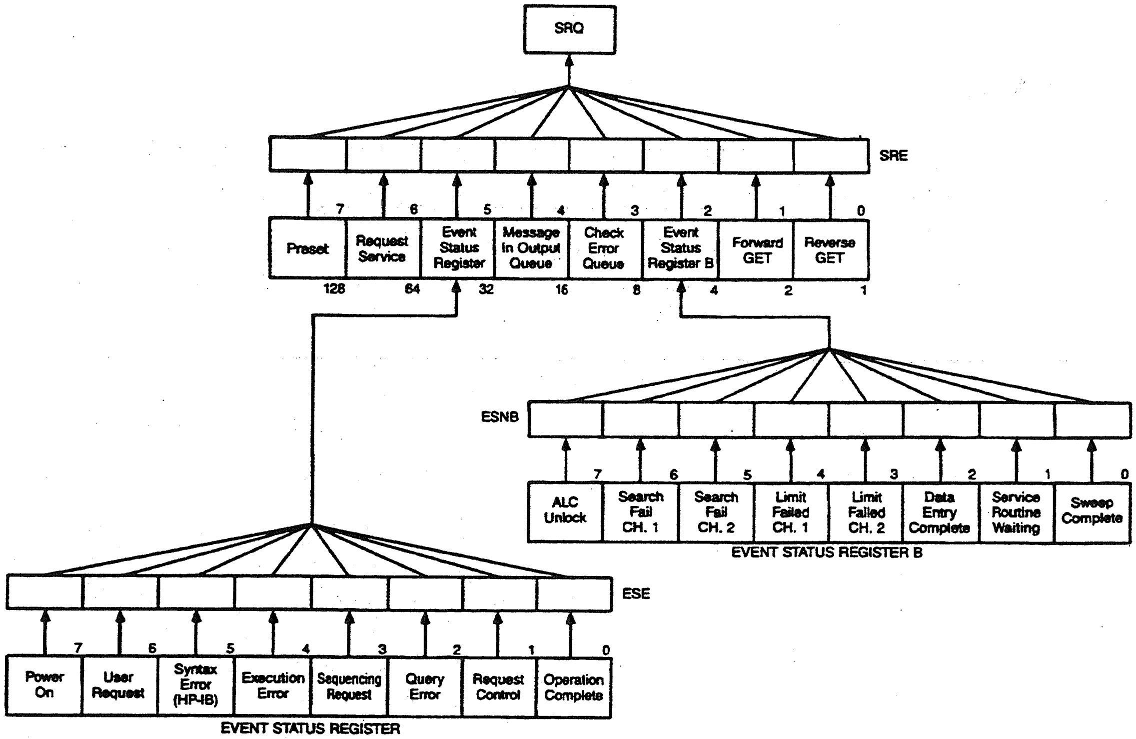


Figure A.1. Status reporting system

Example A2: Using the status registers

The other key components of the status reporting system are the event status register, and event status register B. These 8 bit registers consist of latched event bits. A latched bit is set at the onset of the monitored condition, and is cleared by a read of the register or by clearing the status registers with CLES.

This example program is stored on the Example Programs disk as **IPGA2**

10	`CLEAR 716`	Clear out any old conditions.
20	`OUTPUT 716;"ESR?;"`	Read out the event status register.
30	`ENTER 716;Estat`	
40	`IF NOT BIT(Estat,6) THEN GOTO 20`	If the user request bit of the event status register is not set, loop back.
50	`OUTPUT 716;"KOR?;"`	If the user request bit has been set, there has been some front panel activity, and we read out the key code. The network analyzer's reply to KOR?; includes the knob count if the knob was turned. The information comes as a negative number, and has to be decoded.
60	`ENTER 716;Keyc`	
70	`IF Keyc.≥0 then PRINT "KEY ";`	If the code was positive, it was a key press rather than a knob turn, and print the leader KEY. Placing a semicolon after the statement, suppresses the carriage return, line feed, allowing the code to be printed on the same line.
80	`IF Keyc<-400 THEN Keyc=Keyc+32768`	If the keycode is negative, it is a knob count. If it isn't less than —400, then the count is a clockwise rotation and needs no modification. If the count is less than —400, we have to add 32,768 (2^{15}) to get the counter-clockwise count.
90	`PRINT "CODE =",Keyc`	Print the decoded key code.
100	`GOTO 20`	Wait for the next key press.
110	`END`	

Running the program

Run the program. Pressing a key on the network analyzer causes the computer to display the keycode associated with that key. Note that since the network analyzer is in remote mode, the normal function of the key is not executed. In effect, we have taken over the front panel and can now redefine the keys.

Example A3: Using the preset bit

The purpose of the preset bit is to aid test software in determining if and when the front panel preset key was pressed. To use this feature, bit 7 of the SRQ enable emask must be set.

Internally, the analyzer detects thhat bit 7 is being enabaled and sets a flag in non–volitile memory. On any subsequent front panel preset, the analyzer will execute the normal preset functions, set the preset bit in the status register, check the non-volatile memory flag, and if it is set, generates an SRQ. Thus, bit 7 of the SRQ mask survives a preset. It is, however, reset on a power cycle.

This example is for clarification of the preset bit use. It is not contained on the Example Programs disk.

```
10        SRQmask=BINIOR(Srqmask,128)              Set bit 7 in srqmask variable.
20        OUTPUT 716;"CLES;"                       Clear status registers.
30        OUTPUT 716;"SRE";Srqmask                 Enable srq on preset.
40        ON INTR 7 GOSUB Isr                      Setup interrupt routine.
50        ENABLE INTR 7;2                          Allow the interrupt.
60        LOCAL 716                                Allow control of front panel.
70    Idle:GOTO Idle                               Wait in busy loop for preset.
80        STOP
90    Isr:
100       IF BIT(SPOLL(716),7) THEN                Check reason for srq.
110        PRINT "someone pressed the preset key"
120       END IF
130       OUTPUT 716;"CLES;"                       Clear the interrrupt.
140       ENABLE INTR 7;2                          Re–enable interrupt.
150       LOCAL 716
160       RETURN
170       END
```

Example A4: Generating interrupts

It is also possible to generate interrupts using the status reporting mechanism. The status byte bits can be enabled to generate a service request (SRQ) when set. The 200/300 series computers can in turn be set up to generate an interrupt on the SRQ.

To be able to generate an SRQ, a bit in the status byte has to be enabled using `SREn`. A one in a bit position enables that bit in the status byte. Hence, `SRE 8` enables an SRQ on bit 3, check error queue, since 8 equals 00001000 in binary representation. That means that whenever an error is put into the error queue and bit 3 gets set, the SRQ line is asserted, and the (S) indicator on the front panel of the network analyzer comes on. The only way to clear the SRQ is to disable bit 3, re-enable bit 3, or read out all the errors from the queue.

A bit in the event status register can be enabled so that it is summarized by bit 5 of the status byte. If any enabled bit in the event status register is set, bit 5 of the status byte will also be set. For example `ESE 66` enables bits 1 and 6 of the event status register, since in binary, 66 equals 01000010. Hence, whenever active control is requested or a front panel key is pressed, bit five of the status byte will be set. Similarly, `ESNBn` enables bits in event status register B so that they will be summarized by bit 2 in the status byte.

To generate an SRQ from an event status register, enable the desired event status register bit. Then enable the status byte to generate an SRQ. For instance, `ESE 32;SRE 32;` enables the syntax error bit, so that when the syntax error bit is set, the summary bit in the status byte will be set, and it enables an SRQ on bit 5 of the status byte, the summary bit for the event status register.

The following example program is stored on the Example Programs disk as **IPGA3**.

10	`OUTPUT 716;"CLES; ESE 32; SRE 32;"`	Clear the status reporting system, and then enable bit 5 of the event status register, and bit 5 of the status byte so that an SRQ will be generated on a syntax error.
20	`ON INTR 7 GOTO Err`	Tell the computer where to branch it gets the interrupt.
30	`ENABLE INTR 7;2`	Tell the 200/300 series to enable an interrupt from interface 7 (HP-IB) when bit 1 (value 2, the SRQ bit) of the interrupt register is set. If there is more than one instrument on the bus capable of generating an SRQ, it is necessary to use serial poll to determine which device has issued the SRQ. In this case, we assume the network analyzer did it. A branch to `Err` will disable the interrupt, so the return from `Err` re-enables it.
40	`GOTO 40`	Do nothing loop.
50	`Err:!`	
70	`OUTPUT 716;"ESR?;"`	The interrupt has come in! Read the register to clear the bit.
80	`ENTER 716;Estat`	
90	`PRINT "SYNTAX ERROR DETECTED"`	
100	`ENABLE INTR 7`	
110	`GOTO 30`	
120	`END`	

Running the program

Preset the instrument, and run the program. The computer will do nothing. With the program still running, execute:

```
OUTPUT 716;"STIP 1 GHZ;"
```

The computer will display `SYNTAX ERROR DETECTED`, and the network analyzer will display CAUTION: SYNTAX ERROR, and display the incorrect command, pointing at the first character it did not understand.

The SRQ can be cleared by reading the event status register and hence clearing the latched bit, or by clearing the enable registers with `CLES`. The syntax error message on the network analyzer display can only be cleared by `CLEAR 7` or `CLEAR 716`. `CLEAR 7` is not commonly used because it clears every device on the bus.

Note that an impossible data condition does not generate a syntax error. For example, execute:

```
CLEAR 716
OUTPUT 716;"STAR 10 HZ;"
```

The network analyzer simply sets the start frequency to 300 kHz, without generating a syntax error.

For more information, call your local HP sales office listed in your telephone directory or an HP regional office listed below for the location of your nearest sales office.

United States:
Hewlett-Packard Company
4 Choke Cherry Road
Rockville, MD 20850
(301) 670-4300

Hewlett-Packard Company
5201 Tollview Drive
Rolling Meadows, IL 60008
(312) 255-9800

Hewlett-Packard Company
5161 Lankershim Blvd.
No. Hollywood, CA 91601
(818) 505-5600

Hewlett-Packard Company
2015 South Park Place
Atlanta, GA 30339
(404) 955-1500

Canada:

Hewlett-Packard Ltd.
6877 Goreway Drive
Mississauga, Ontario L4V1M8
(416) 678-9430

Australia/New Zealand:
Hewlett-Packard Australia Ltd.
31-41 Joseph Street,
Blackburn, Victoria 3130
Melbourne, Australia
(03) 895-2895

Europe/Africa/Middle East:
Hewlett-Packard S.A.
Central Mailing Department,
P.O. Box 529
1180 AM Amstelveen,
The Netherlands
(31) 20/547 9999

Far East:
Hewlett-Packard Asia Ltd.
22/F Bond Centre
West Tower
89 Queensway
Central, Hong Kong
(5) 8487777

Japan:
Yokogawa-Hewlett-Packard Ltd.
29-21, Takaido-Higashi 3-chome
Suginami-ku, Tokyo 168
(03) 331-6111

Latin America:
Latin American Region Headquarters
Monte Pelvoux Nbr. 111
Lomas de Chapultepec
11000 Mexico, D.F. Mexico
(905) 596-79-33

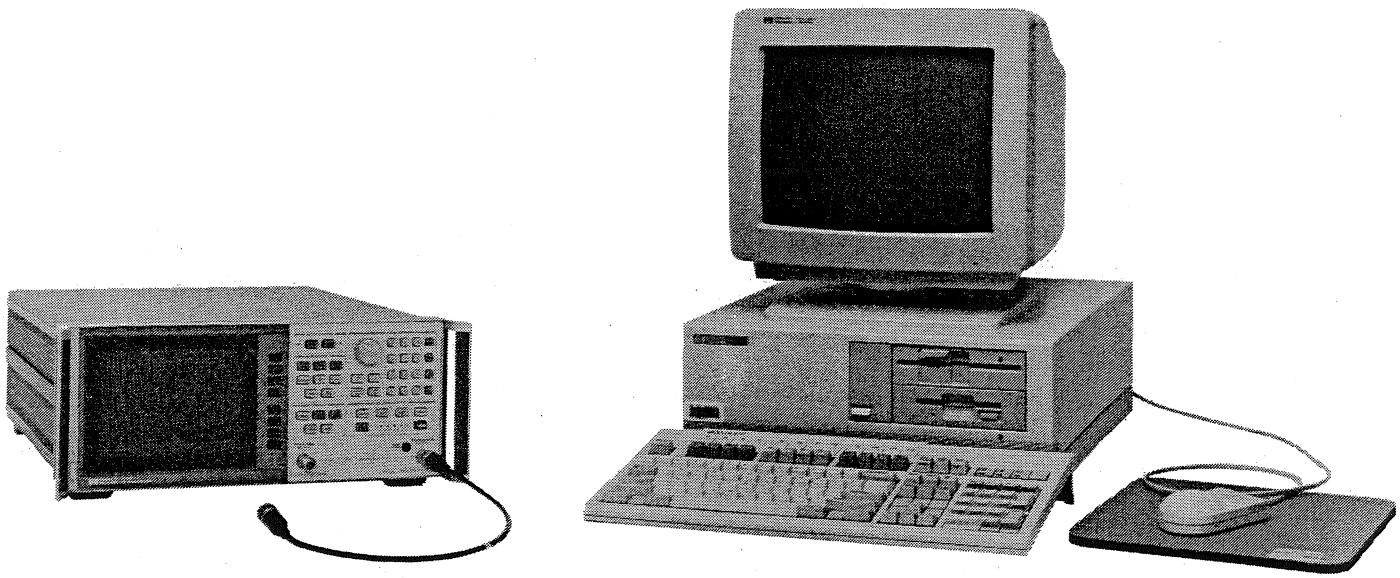
January 15, 1992

HEWLETT PACKARD

HP-IB Programming Guide

HEWLETT PACKARD

For the HP 8752A and HP 8753C Network Analyzers with the HP Vectra Personal Computer using Microsoft® QuickBASIC 4.5



Introduction

This programming guide is an introduction to remote operation of the HP 8752A and 8753C Network Analyzers with an HP Vectra Personal Computer (or IBM compatible) using the HP 82335A HP-IB Command Library and Microsoft QuickBASIC 4.5. This is a tutorial introduction, using programming examples to demonstrate the control of network analyzers with HP-IB commands. The example programs are on the Example Programs disk (part number 08753-10020) included with the operating manual. This document is closely associated with the *HP-IB Quick Reference* for the HP 8700-series network analyzers, which provides complete programming information in a concise format. Included in the *HP-IB Quick Reference* is an alphabetical list of HP-IB mnemonics and their explanations.

This note assumes that the reader is familiar with the operation of the network analyzer and the HP Vectra Personal Computer (or compatible), particularly HP-IB operation using the HP 82335A Command Library. This document is not intended to teach QuickBASIC programming or to discuss HP-IB theory except at an introductory level. See the section entitled *Reference information* for documents better suited to these tasks.

Reference information

HP 8752A/8753C Network Analyzer literature

User's Guide
Quick Reference
Operating Manual

HP-IB and HP Vectra Personal Computer literature

Tutorial Description of the Hewlett-Packard Interface Bus
Condensed Description of the Hewlett-Packard Interface Bus
HP 82335A HP-IB Command Library Manual

Microsoft QuickBASIC 4.5 literature

Microsoft QuickBASIC: BASIC Language Reference
Microsoft QuickBASIC: Learning and Using Microsoft QuickBASIC
Microsoft QuickBASIC: Programming in BASIC: Selected Topics

Table of contents

Equipment

To run the examples in this Programming Guide, the following equipment is required:

- HP 8752A or 8753C Network Analyzer.
- HP Vectra Personal Computer (or compatible) with Microsoft QuickBASIC 4.5, HP 82335A HP-IB Interface Card, MS-DOS® 3.2 or higher, and at least 320 Kbytes of memory.
- HP 10833A/B/C/D HP-IB cables to interconnect the computer, the network analyzer, and any peripherals.

The following equipment is optional:

- HP 85032B 50 ohm type-N calibration kit.
- HP 11857D 7 mm test port return cables (HP 8753C only).
- A test device such as a filter to use in the example measurement programs.

Preparation

1. **System.** Connect the network analyzer to the computer with an HP-IB cable. The network analyzer has only one HP-IB interface, but it occupies two addresses: one for the instrument and one for the display. The display address is the instrument address with the least significant bit complemented. The default addresses are 16 for the instrument and 17 for the display. Other devices on the bus cannot occupy the same address as the network analyzer.

2. **Computer.** Turn on the computer and load QuickBASIC by typing `QB /L QBHPIB` at the MS-DOS prompt. Invoking QuickBASIC in this way will load the Quick library `QBHPIB.QLB`, making its contents available for use.

3. **Network analyzer.** Turn on the network analyzer and verify its address by pressing **[LOCAL]** *[SET ADDRESSES]* and *[ADDRESS: 875x]*. If the address has been changed from the default value (16), return it to 16 to perform the examples in this document by pressing **[1] [6] [x1] [PRESET]**. Make sure the instrument is in *[TALKER/LISTENER]* mode, as indicated under the **[LOCAL]** key, since this is the only mode in which the network analyzer and an HP Vectra can communicate over HP-IB.

4. **Connection.** Type the following on the computer in the immediate portion of the display and all on one line:

```
CALL IOOUTPUTS(716&, "PRES;",
LEN("PRES;")): IF PCIB.ERR <> NOERR THEN
ERROR PCIB.BASERR
```

This presets the network analyzer. If a preset does not occur, there is a problem. Since many HP-IB problems are caused by an incorrect address or bad or loose HP-IB cables, check all HP-IB addresses and connections.

Notes on QuickBASIC

In QuickBASIC, multiple statements are allowed per line, and line numbers are not required. In the examples in this programming guide, line numbers are included for clarity. Each line is preceded by a line number, and each line number is followed by a complete one-line statement. No carriage returns are used in the statements although it may appear that way on the following pages.
The following error trapping line should follow every call to an I/O routine:

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

In the following example programs, this line is generally made into a separate routine that can easily be executed after every call to an I/O routine:

```
CALL IOXXXX: GOSUB ERRORTRAP
```

If an error occurs, the number corresponding to that error is assigned to the variable `PCIB.ERR` and the program branches to an HP-IB Command Library subprogram for error handling which displays a message on the computer screen stating the error number and type.

Since the IOOUTPUTS command library routine to send a command from the computer to the analyzer is called so often and is so long, it is worthwhile to make it into a separate routine (called IOOUTS here) that can be executed with a GOSUB statement. If this is done, the line to preset the analyzer becomes

```
A$ = "PRES;": GOSUB IOOUTS
```

and the program END is followed by the ERRORTRAP and IOOUTS routines.

```
:
END

ERRORTRAP:
   IF PCIB.ERR <>NOERR THEN ERROR PCIB.BASERR
RETURN

IOOUTS:
   CALL IOOUTPUTS(716&, A$, LEN(A$)): GOSUB
   ERRORTRAP
RETURN
```

The construction of the IOOUTPUTS call is as follows:

```
CALL IOOUTPUTS(716&, A$, LEN(A$)): GOSUB
ERRORTRAP
```

CALL IOOUTPUTS: command. Execute the HP-IB string data output command.

716&: address. The data is directed to interface 7 (HP-IB) and out to the device at address 16 (the network analyzer). The appended "&" is required by the IO routine, which expects a long-integer.

A$: HP-IB command string. A$ should be set equal to the mnemonic corresponding to the desired operation before the GOSUB IOOUTS command that will execute the call to IOOUTPUTS is given.

LEN(A$): length. The IOOUTPUTS routine must know the length (in characters) of the command string it is sending so that it can append an appropriate line terminator.

GOSUB ERRORTRAP: error trap. The call to an error trapping routine that must follow every call to an I/O routine.

Just as there are I/O commands to send data to the analyzer, there are I/O commands to receive data from the analyzer. For more information on this topic, see the section entitled *Transferring Data.*

Basic Instrument Control

Preparation for HP-IB control

At the beginning of a program, the network analyzer has to be taken from an unknown state and brought under computer control. One way to do this is with an abort/clear sequence, which prepares the bus for activity and the analyzer for receiving HP-IB commands. In addition, a time-out should be set (IOTIMEOUT), and, if the program will be transferring data, the end-or-identify mode should be disabled (IOEOI). Because a known initial instrument state makes programs more reliable, the next step is generally to put the network analyzer into a known state. The most convenient way to do this is to send PRES, which returns the analyzer to the preset state. If preset is not desired and the status reporting mechanism is going to be used, CLES can be sent to clear all of the status reporting registers and their enabled bits.

For an example of the necessary preparation for HP-IB control in QuickBASIC programs, load the following program (stored on the Example Programs disk as **IPGI1.BAS**). Note that the first four I/O commands are to the address 7&, the interface bus. Only the IOOUTPUTS command is actually to the analyzer, address 716&.

```
10   CALL IOTIMEOUT(7&, 10!): GOSUB ERRORTRAP
```

Define a system time-out of 10 seconds. (This value is chosen because most sweeps and calibration calculations are completed in under 10 seconds.) Time-out allows recovery from I/O operations that are not completed in the allowed number of seconds.

```
20   CALL IOABORT(7&): GOSUB ERRORTRAP
```

Halt any bus activity and return active control to the computer.

```
30   CALL IOCLEAR(7&): GOSUB ERRORTRAP
```

Clear syntax errors, the input command buffer, and any messages waiting to be sent out. This command does not affect the status reporting system.

```
40   CALL IOEOI(7&, 0): GOSUB ERRORTRAP
```

Disable the end-or-identify mode for transferring data. This prevents both a write operation from setting the EOI line on the last byte of the write and a read operation from terminating upon sensing that the EOI line has been set.

```
50   A$ = "PRES;": GOSUB IOOUTS
```

Send the HP-IB mnemonic PRES to the network analyzer (address = 716) via the IOOUTS subroutine. This presets the instrument, clears the status reporting system, and resets all front panel settings except the HP-IB mode and the HP-IB addresses.

```
60   END
```

End program execution.

```
70   ERRORTRAP:
80   IF PCIB.ERR <> NOERR THEN ERROR
     PCIB.BASERR
90   RETURN
100  IOOUTS:
110  CALL IOOUTPUTS(716&, A$, LEN(A$)):
     GOSUB ERRORTRAP
120  RETURN
```

This program brings the network analyzer to a known state and prepares it to respond to HP-IB control.

The network analyzer will not respond to HP-IB commands unless the remote line is asserted. When the remote line is asserted and the analyzer is addressed to listen, it automatically goes into remote mode. Remote mode means that all front panel keys except **[LOCAL]** and the line power switch are disabled. The command `IOABORT` asserts the remote line, which remains asserted until the command `IOLOCAL` is executed. Another way to assert the remote line is to execute

```
CALL IOREMOTE(716&): GOSUB ERRORTRAP
```

This statement asserts the remote line and addresses the network analyzer to listen, thereby putting it into remote mode. Now no front panel key will respond until **[LOCAL]** is pressed.

The local key can also be disabled with the following sequence:

```
CALL IOREMOTE(716&): GOSUB ERRORTRAP
CALL IOLLOCKOUT(7&): GOSUB ERRORTRAP
```

Now no front panel key (including **[LOCAL]**) except the line power switch will respond. The analyzer can be returned to local mode temporarily with the following command:

```
CALL IOLOCAL(716&): GOSUB ERRORTRAP
```

However, as soon as the analyzer is next addressed to listen, it goes back into local lockout. The only way to clear local lockout, other than cycling power, is to execute

```
CALL IOLOCAL(7&): GOSUB ERRORTRAP
```

This disables the remote line on the interface, puts the instrument into local mode, and clears local lockout.

Commands

A computer controls the network analyzer by sending it commands over HP-IB. Each command is specific to the network analyzer and is executed automatically, taking precedence over analyzer manual control. A command applies only to the active channel unless functions are coupled between channels, just as with front panel operation. Most commands are equivalent to front panel functions.

No operand commands

The simplest command that the network analyzer accepts is one that requires no operand. For example, `AUTO` is a no operand command. Leave the previous program in the main window and put the cursor in the immediate window. Now execute

```
A$ = "AUTO;": GOSUB IOOUTS
```

In response, the network analyzer autoscales the active channel just as it would if **[SCALE REF]** *[AUTO SCALE]* were pressed on the analyzer's front panel.

The semicolon following `AUTO` terminates the command inside the network analyzer. It clears the active entry area and prepares the network analyzer for the next command. If there is a syntax error in a command, the network analyzer will ignore the command and look for the terminating semicolon. When it finds this terminator, the network analyzer starts processing incoming commands normally. Characters between the syntax error and the next terminator are lost. A line feed can also act as terminator. The QuickBASIC `IOOUTPUTS` routine, which is called from the user-defined subprogram `IOOUTS`, automatically transmits a carriage return/line feed following the data if there is not a semicolon at the end of the statement.

The `IOOUTPUTS` routine will transmit all commands listed, as long as they are separated by commas or semicolons. All the information enclosed in quotes will be transmitted literally. A carriage return/line feed is transmitted after each command, but this can be prevented by separating commands with semicolons instead of commas.

The network analyzer does not distinguish between upper and lower case letters. For example, execute

```
A$ = "auto;": GOSUB IOOUTS
```

On/off commands

The network analyzer also accepts a command that turns a function on and off. Execute

```
A$ = "DUACON;": GOSUB IOOUTS
```

This activates dual channel display mode on the network analyzer. To restore single channel display mode, execute

```
A$ = "DUACOFF;": GOSUB IOOUTS
```

The command is composed of the root mnemonic `DUAC` (dual channel) and `ON` or `OFF`.

In addition, the network analyzer has a debug mode to aid in troubleshooting systems. When debug mode is on, the network analyzer scrolls incoming HP-IB commands across the display. To turn this mode on manually, press **[LOCAL]** *[HP-IB DIAG ON]*. To turn it on over HP-IB, execute

```
A$ = "DEBUON;": GOSUB IOOUTS
```

Parameter setting commands

The analyzer also accepts commands that set parameters. For example, execute

```
A$ = "STAR 10 MHZ;": GOSUB IOOUTS
```

The network analyzer now has a start frequency of 10 MHz. The STAR 10 MHZ command performs the same function as keying in **[START] [1] [0] [M/u]** from the network analyzer's front panel. STAR is the root mnemonic for the start key, 10 is the data, and MHZ is the units. The network analyzer's root mnemonics are derived from the equivalent key label if possible and from the common name for the function if not. The *HP-IB Quick Reference* lists all the root mnemonics and all the different units accepted.

Notice that the front panel remote (R) and listen (L) HP-IB status indicators are on. The network analyzer automatically goes into remote mode when it is sent a command with the IOOUTPUTS statement.

Interrogate instrument state commands

Each instrument parameter can be interrogated to find its current state or value with query commands. If a question mark is appended to the root mnemonic of a command, the network analyzer will send out the value of that parameter. For example, the command POWE 5 DB sets the analyzer's output power to +5 dBm, and the command POWE? tells the analyzer to send out the current RF output power value at the test port to the computer. The program in the main window can be modified to show the use of this command by deleting line 50 and inserting the following lines before the END at line 60.

```
45   A$ = "POWE?;": GOSUB IOOUTS
50   CALL IOENTER(716&, REPLY!): GOSUB
     ERRORTRAP
55   PRINT REPLY!
```

This modified program is stored on the Example Programs disk as **IPGI2.BAS**.

Now run the program, and the computer will display the source power level in dBm. The preset level is 0 dBm for the 8753C and −10 dBm for the 8752A. Next change the power level by pressing **[LOCAL] [MENU]** *[POWER]* **[1] [x1]**, and run the program again.

When the network analyzer receives the command POWE?, it prepares to send out the current RF source power level. The QuickBASIC statement CALL IOENTER(716&, REPLY!): GOSUB ERRORTRAP addresses the analyzer to talk, thereby allowing it to transmit information to the computer. This turns the network analyzer front panel talk light (T) on. The computer places the data transmitted by the network analyzer into the variable listed in the IOENTER statement. In this case, the network analyzer transmits the output power value, and this gets placed in the real number variable REPLY!.

The IOENTER statement takes the binary data sent out from the network analyzer and formats it into a real number. There are other I/O routines for entering a string (IOENTERS), an array of real numbers (IOENTERA), and unformatted data (IOENTERAB, IOENTERB). The data being requested is determined by the I/O routine and must correspond to the variable being received.

On/off commands can be also be interrogated. The reply is 1 if the function is on and 0 if it is off. Similarly, if a command controls a function that is underlined on the network analyzer display when active, interrogating that command yields 1 if the command is underlined and 0 if it is not. For example, there are nine options in the format menu, and only one is underlined at a time. Of the nine, only the underlined option will return 1 when interrogated.

For instance, rewrite line 45 as

```
45   A$ = "DUAC?;": GOSUB IOOUTS
```

Run the program once and note the result. Then press **[LOCAL] [DISPLAY]** *[DUAL CHAN]* to toggle the display mode, and run the program again to observe the difference.

Another example is to rewrite line 45 as

```
45   A$ = "PHAS?;": GOSUB IOOUTS
```

In this case, the computer will display 1, only if phase is currently being displayed on the network analyzer. Since the command only applies to the active channel, the response to the PHAS? inquiry depends on which channel is active.

Held commands

A held command is one that cannot be interrupted during its execution. When the network analyzer is executing a held command, it holds off processing new HP-IB commands, halting HP-IB operation until the held command completes execution. Some examples of held commands are DONE, PRES, and SING.

While a held command is executing, the network analyzer will still service the HP-IB interface routines, such as IOSPOLL, IOCLEAR, and IOABORT, all of which must be called and followed by error trapping. Executing a call to IOCLEAR will abort a held command, leaving its execution to be completed as if it had been begun from the front panel. These routines (IOSPOLL, IOCLEAR, and IOABORT) also clear the input buffer, destroying any commands received after the held command. If the network analyzer has halted the bus because its input buffer was full, executing a call to the routine IOABORT will release the bus.

Operation complete (OPC)

The operation complete (OPC) function allows synchronization of the program by requiring the current command to complete execution before the next command can begin. For instance, a program should not have the operator connect the next calibration standard while the network analyzer is still measuring the current one. To provide OPC information, the network analyzer uses its OPC reporting mechanism, which indicates when the execution of certain key commands has been completed. The function is activated by sending either `OPC` or `OPC?` immediately before an OPC'able command. When the command completes execution, bit 0 of the Event Status Register is set. If `OPC?` is interrogated, the network analyzer outputs 1 when the command completes execution.

The program in the main window can be modified to show the use of the `OPC?` command by deleting lines 45 through 55 and inserting the following lines before the `END` at line 60.

```
44   A$ = "SWET 3 S; OPC?; SING;": GOSUB
     IOOUTS
```

Set the sweep time to 3 seconds, and `OPC?` a single sweep.

```
48   PRINT "SWEEPING"
52   CALL IOENTER(716&,REPLY!): GOSUB
     ERRORTRAP
```

The program will halt until the network analyzer completes the sweep and sends out 1.

```
56   PRINT "DONE"
```

The modified program is stored on the Example Programs disk as **IPGI3.BAS**.

When it is run, the computer displays the sweeping message as the analyzer executes the sweep, and the computer displays `DONE` when the analyzer finishes the sweep. When `DONE` appears, the program can continue with a valid data trace ensured in the analyzer. Without a single sweep, it takes more than one sweep time to ensure good data.

Measurement Programming

The previous section of this document outlined the process to get commands into the network analyzer. The next step is to organize the commands into a measurement sequence. A typical measurement sequence consists of the following steps:

1. Prepare the instrument.
2. Calibrate the instrument.
3. Connect the device under test.
4. Make the measurement.
5. Process the data.
6. Transfer the data.

Prepare the instrument

Define the measurement by setting the basic measurement parameters. These include all the stimulus parameters (sweep type, span, sweep time, number of points, and RF power level) as well as the parameter to be measured, IF averaging, and IF bandwidth. These parameters define how data is gathered and processed within the instrument. Changing any parameter requires that a new sweep be taken.

Other parameters can be set within the instrument, such as smoothing, trace scaling, or trace math, that do not directly affect data gathering. These functions are classified as post processing functions: they can be changed with the instrument in hold mode, and the data will correctly reflect the new state.

The save/recall registers and the learn string are two rapid ways of setting up an entire instrument state. The learn string is a string summary of the instrument state that can be read into and sent out from the computer, as shown in Example 6A: *Using the learn string.*

Calibrate the instrument

Measurement calibration is normally performed once the instrument state has been defined. Although it is not required to make a measurement, calibration improves the accuracy of the data.

There are several ways to calibrate the instrument. The simplest way is to stop the program and have the operator perform the calibration from the front panel. Alternatively, the computer can be used to guide the operator through the calibration, as shown in Examples 2A: *1-port calibration* and 2B: *Full 2-port calibration (HP 8753C only)*. Lastly, calibration data saved from a previous calibration can be transmitted back into the instrument, as shown in Example 6B: *Reading calibration data.* This should only be done if the hardware configuration has not changed.

Connect the device under test

The computer can be used to verify that the device is connected properly and to speed up the adjustment process. Useful functions for this purpose include limit testing, bandwidth searches, and trace statistics. All device adjustments should take place at this stage and be finished before taking data.

Make the measurement

Once the device is connected and adjusted, measure its frequency response and hold the data within the instrument so that there is a valid trace to analyze. The single sweep command `SING` is designed to do this. All stimulus changes are completed before the sweep is started, and the HP-IB hold state is not released until the formatted trace is displayed. When the sweep is complete, the instrument is put into hold mode, which freezes the data inside the instrument. Because single sweep is OPC'able, it is easy to determine when the sweep has been completed.

The number of groups command NUMGn is similar to SING, but it triggers n sweeps. This is useful, for example, in making a measurement with an averaging factor n (n can range from 1 to 999). Both SING and NUMGn commands restart averaging.

Process the data

With valid data to operate on, the post-processing functions can be used. Referring ahead to the data processing chain in Figure 1 (page 20), notice that any function that affects the data after the error correction stage can be used. The most useful functions are trace statistics, marker searches, electrical delay offset, time domain, and gating. If a 2-port calibration is active, then any of the four S-parameters can be viewed without taking a new sweep.

Transfer the data

Lastly, transmit the results out of the instrument. Each data output command is designed to ensure that transmitted data reflects the current state of the instrument.

- The commands OUTPDATA, OUTPRAWn, and OUTPFORM will transmit data only after all formatting functions have completed.
- The commands OUTPLIML, OUTPLIMM, and OUTPLIMF will transmit data only after a limit test has occurred (if limit testing is on).
- The command OUTPMARK will activate a marker (if one is not already selected) and will transmit data only after any current marker searches have completed.
- The command OUTPMSTA will transmit data only after marker statistics for the current trace have been calculated. If the statistics function is not on, it will be turned on to update the current values and then turned off.
- The command OUTPMWID will transmit data only after a bandwidth search has been executed for the current trace. If the bandwidth search function is not on, it will be turned on to update the current values and then turned off.

Data transfer is discussed further in Examples 3A through 3D: *Transferring data.*

Basic Programming Examples

Making measurements

The procedure for setting up measurements on the network analyzer via HP-IB follows the same sequence as when the setup is performed manually. As long as the desired frequency range, number of points, and power level are set prior to performing the calibration, there is no required order.

Example 1: Setting up a basic measurement

The following program illustrates how to set up a basic measurement on the network analyzer. The program will select the desired parameter, measurement format, and frequency range. Performing calibrations is described in later examples.

This example program is stored on the Example Programs disk as **IPG1.BAS**.

`10 REM $INCLUDE: 'QBSETUP'`	Call the QuickBASIC initialization file `QBSETUP`, the setup program for the MS-DOS HP-IB Command Library. This command should appear before the body of the program whenever calls to the HP-IB Command Library are to be made.
`20 CLS`	Clear the computer CRT.
`30 ISC& = 7`	Assign the interface select code to a variable. This select code is set on the HP 82335A HP-IB interface card.
`40 VNA& = 716`	Assign the address of the HP 8753C/8752A to a variable.
`50 CALL IOTIMEOUT(ISC&, 10!): GOSUB ERRORTRAP`	Define a system time-out of 10 seconds and perform error trapping. Time-out allows recovery from I/O operations that are not completed in under 10 seconds.
`60 CALL IOABORT(ISC&): GOSUB ERRORTRAP`	Abort any HP-IB transfers and perform error trapping.
`70 CALL IOCLEAR(ISC&): GOSUB ERRORTRAP`	Clear the analyzer's HP-IB interface and perform error trapping.
`80 CALL IOEOI(ISC&, 0): GOSUB ERRORTRAP`	Disable the End-Or-Identify mode for transferring data and perform error trapping.
`90 A$ = "PRES; MENUOFF;": GOSUB IOOUTS`	Preset the network analyzer and turn its softkey menu off.
`100 A$ = "CHAN1; S21; LOGM;": GOSUB IOOUTS`	Make channel 1 the active channel and measure the forward transmission parameter, displaying its magnitude in decibels. The mnemonic for this parameter is the same for both analyzers (`S21`) although it is called TRANSMISSION on the HP 8752A.
`110 A$ = "CHAN2; S21; PHAS;": GOSUB IOOUTS`	Make channel 2 the active channel and measure the phase of the forward transmission parameter.
`120 A$ = "DUACON;": GOSUB IOOUTS`	Display both channels simultaneously.
`130 LOCATE 1, 1: INPUT "ENTER START FREQUENCY (MHz): ", F.START!`	Position the cursor on the computer CRT at (row,column) = (1,1), and read in a real start frequency, `F.START!`.
`140 LOCATE 1, 41: INPUT "ENTER STOP FREQUENCY (MHz): ", F.STOP!`	Read in a real stop frequency, `F.STOP!`.

`150 A$ = "STAR" + STR$(F.START!) + "MHz;": GOSUB IOOUTS`	Set the start frequency on the network analyzer to `F.START!`. In QuickBASIC, the "+" is used to concatenate strings.
`160 A$ = "STOP" + STR$(F.STOP!) + "MHz;": GOSUB IOOUTS`	Set the stop frequency on the network analyzer to `F.STOP!`.
`170 A$ = "AUTO;": GOSUB IOOUTS`	Autoscale the network analyzer's active channel (2).
`180 A$ = "CHAN1; AUTO;": GOSUB IOOUTS`	Activate and autoscale channel 1.
`190 A$ = "MENUON;": GOSUB IOOUTS`	Turn the network analyzer's softkey menu back on.
`200 CALL IOLOCAL(ISC&): GOSUB ERRORTRAP`	Return the network analyzer to local mode and perform error trapping.
`210 END`	End program execution.
`220 ERRORTRAP:`	Define a routine to trap errors.
`230 IF PCIB.ERR <> NOERR THEN ERROR PCIB.BASERR`	Perform error trapping.
`240 RETURN`	Return from the `ERRORTRAP` routine.
`250 IOOUTS:`	Define a routine to send a command string from the computer to the analyzer.
`260 CALL IOOUTPUTS(VNA&, A$, LEN(A$)): GOSUB ERRORTRAP`	Send the command string `A$` out to the analyzer and perform error trapping.
`270 RETURN`	Return from the `IOOUTS` routine.

Running the program

1. The computer sets up a measurement of transmission log magnitude on channel 1 and transmission phase on channel 2, displaying both measurements simultaneously by using the dual channel display mode.
2. Enter any valid value in MHz when prompted for start and stop frequencies.
3. The computer will enter the specified start and stop frequencies into the network analyzer, and they will be the frequency limits of the analyzer's display.

Performing calibrations

Coordinating a measurement calibration over HP-IB follows the keystrokes required to calibrate from the front panel in that there is a command for every step. The general key sequence is to select the calibration, to measure the calibration standards, and then to declare the calibration done. The actual sequence depends on the calibration kit and changes slightly for 2-port calibrations*, which are divided into three calibration sub-sequences.

The calibration kit tells the network analyzer which standards to expect at each step of the calibration. The set of standards associated with a given calibration is termed a class. For example, measuring the short during a 1-port calibration is one calibration step. All of the shorts that can be used for this calibration step make up the class, which is called class S11B. For the 7 mm and the 3.5 mm cal kits, class S11B has only one standard, so selecting *[SHORT]* automatically measures the short. For type-N cal kits, however, class S11B has two standards: male and female test ports. Selecting *[SHORTS]* brings up a second menu, allowing the operator to select which standard in the class is to be measured. The sex listed refers to the test port.

To do a 1-port calibration over HP-IB using the 7 mm or 3.5 mm cal kits, sending the command `CLASS11B` will automatically measure the short. For the type-N cal kit, sending `CLASS11B` brings up the menu with the male and female test port options. To select one of these standards, use either the command `STANA` or the command `STANB`. The `STAN` command can be appended with the letters `A` through `G`, corresponding to the standards listed under softkeys 1 through 7, softkey 1 being the uppermost softkey. The `STAN` command is always OPC'able, but a `CLASS` command is OPC'able only if the class has just one standard in it, which is then automatically measured. This is because when there is more than one standard in a class, the command that calls the class simply brings up another menu.

Each full 2-port measurement calibration is divided into three subsequences: transmission, reflection and isolation. Each subsequence is treated like a calibration in its own right: each must be opened, all of its standards must be measured, and then it must be declared done. The opening and closing commands for the subsequences are similar.

Transmission subsequence: `TRAN` and `TRAD`
Reflection subsequence: `REFL` and `REFD`
Isolation subsequence: `ISOL` and `ISOD`

*HP 8753C only.

Example 2A: 1-port calibration

The following program illustrates how to perform a 1-port measurement calibration on the network analyzer over HP-IB. The program does the calibration using the HP 85032B 50 ohm type-N calibration kit. It steps the operator through the calibration by giving explicit directions on the network analyzer display and allowing the user to continue the program from the network analyzer front panel. The desired instrument state should be set up before the program is run.

This example program is stored on the Example Programs disk as **IPG2A.BAS**.

`10 DECLARE SUB ERRORTRAP ()`	Define a subroutine to trap errors.
`20 DECLARE SUB IOOUTS (A$, ADDRESS&)`	Define a subroutine to send a command string from the computer to the analyzer.
`30 DECLARE SUB WAITFORKEY (LABEL$, VNA&, DISPLAY&, ISC&)`	Define a subroutine to display a message on the analyzer and wait for the operator to press a key.
`40 REM $INCLUDE: 'QBSETUP'`	Call the QuickBASIC initialization file `QBSETUP`.
`50 CLS`	Clear the computer CRT.
`60 ISC& = 7`	Assign the interface select code to a variable.
`70 VNA& = 716`	Assign the analyzer's address to a variable.
`80 DISPLAY& = 717`	Assign the analyzer's display address to a variable.
`90 CALL IOTIMEOUT(ISC&, 10!): CALL ERRORTRAP`	Define a system time-out of ten seconds and perform error trapping.
`100 CALL IOABORT(ISC&): CALL ERRORTRAP`	Abort any HP-IB transfers and perform error trapping.
`110 CALL IOCLEAR(ISC&): CALL ERRORTRAP`	Clear the analyzer's HP-IB interface and perform error trapping.
`120 CALL IOEOI(ISC&, 0): CALL ERRORTRAP`	Disable the End-Or-Identify mode for transferring data and perform error trapping.
`130 CALL IOOUTS("CALKN50; MENUOFF; CLES; ESE64;", VNA&)`	Select the 50 ohm type-N cal kit, turn off the softkey menu, clear the status byte, and set up the status reporting system so that bit 6, User Request, of the Event Status Register is summarized by bit 5 of the status byte, allowing a key press to be detected by a serial poll. For more information about setting up status reporting systems, refer to *Example 7: Interrupt generation*.
`140 CALL IOOUTS("WAIT;", VNA&)`	Wait for a clean sweep on the analyzer so that the following command will have the proper effect.
`150 CALL IOOUTS("ENTO;", VNA&)`	Clear the analyzer's entry area.
`160 CALL IOOUTS("CALIS111;", VNA&)`	Open the calibration by calling the S11 1-port calibration.
`170 CALL WAITFORKEY("CONNECT OPEN AT PORT 1", VNA&, DISPLAY&, ISC&)`	Ask for an open and wait for the operator to connect it.
`180 CALL IOOUTS("CLASS11A; OPC?; STANB;", VNA&)`	Measure the open. Identify the specific standard (female test port) within the class using the command `STANB`, indicating the option at the second softkey from the top.
`190 CALL IOENTER(VNA&, REPLY!): CALL ERRORTRAP`	Wait for the standard to be measured.

`200 CALL WAITFORKEY("CONNECT SHORT AT PORT 1", VNA&, DISPLAY&, ISC&)`	Ask for a short and wait for the operator to connect it.
`210 CALL IOOUTS("CLASS11B; OPC?; STANB;", VNA&)`	Measure the short. Identify the specific standard (female test port) within the class.
`220 CALL IOENTER(VNA&, REPLY!): CALL ERRORTRAP`	Wait for the standard to be measured.
`230 CALL WAITFORKEY("CONNECT LOAD AT PORT 1", VNA&, DISPLAY&, ISC&)`	Ask for a load and wait for the operator to connect it.
`240 CALL IOOUTS("OPC?; CLASS11C;", VNA&)`	Measure the load. There are no options within this class, so `OPC?`, which always precedes the last command, comes first.
`250 CALL IOENTER(VNA&, REPLY!): CALL ERRORTRAP`	Wait for the standard to be measured.
`260 CALL IOOUTS("PG;", DISPLAY&)`	Clear the user graphics by removing the last prompt.
`270 CLS : PRINT "COMPUTING CALIBRATION COEFFICIENTS"`	Display program progress on the computer CRT.
`280 CALL IOOUTS("DONE; OPC?; SAV1;", VNA&)`	Complete the calibration and save it.
`290 CALL IOENTER(VNA&, REPLY!): CALL ERRORTRAP`	Wait until the network analyzer has computed the calibration coefficients before continuing.
`300 CLS : PRINT "1-PORT CALIBRATION COMPLETED. CONNECT TEST DEVICE."`	Display program progress and instructions on the computer CRT.
`310 CALL IOOUTS("MENUON;", VNA&)`	Turn the softkey menu back on.
`320 CALL IOLOCAL(ISC&): CALL ERRORTRAP`	Return the analyzer to local mode and perform error trapping.
`330 END`	End program execution.
`340 SUB ERRORTRAP`	Define a subroutine to trap errors.
`350 IF PCIB.ERR <> NOERR THEN ERROR PCIB.BASERR`	Perform error trapping.
`360 END SUB`	Return from the `ERRORTRAP` subroutine.
`370 SUB IOOUTS (A$, ADDRESS&) STATIC`	Define a subroutine to send a command string from the computer to the analyzer.
`380 CALL IOOUTPUTS(ADDRESS&, A$, LEN(A$)): CALL ERRORTRAP`	Send the command string `A$` out to the analyzer and perform error trapping.
`390 END SUB`	Return from the `IOOUTS` subroutine.
`400 SUB WAITFORKEY (LABEL$, VNA&, DISPLAY&, ISC&) STATIC`	Define a subroutine to display a message on the analyzer and wait for the operator to press a key.
`410 CLS : PRINT LABEL$`	Display instructions on the computer CRT.

```
420 CALL IOOUTS("PG; PU; PA
    390,3600; PD; LB" +
    LABEL$ + "; PRESS ANY KEY
    WHEN READY." + CHR$(3),
    DISPLAY&)
```

Write on the network analyzer's display:
`PG` : PaGe; clears old user graphics.
`PU` : Pen Up; prevents anything from being drawn.
`PA` : Pen At; positions the logical pen.
`PD` : Pen Down; enables drawing.
`LB` : LaBel; writes the message on the display. The label must always be terminated by the ETX symbol, CHR$(3).

```
430 CALL IOOUTS("ESR?;",
    VNA&)
```

Request the Event Status Register value from the analyzer.

```
440 CALL IOENTER(VNA&,
    ESTAT!): CALL ERRORTRAP
```

Receive the Event Status Register value from the analyzer, thereby clearing the latched User Request bit so that old key presses will not trigger a measurement.

```
450 CALL IOOUTS("ESE64;",
    VNA&)
```

Ensure that the proper status reporting system is still in effect.

```
460 STAT% = 0
```

Initialize `STAT%` for entry into the `DO UNTIL` loop.

```
470 DO UNTIL ((STAT% MOD 64)
    >31)
```

Wait for a key press to be indicated by the setting of bit 5 of the status byte. `MOD 64` removes the effect of all higher value bits (bit 6 is equivalent to 64 in decimal), and `>31` ensures that bit 5, which is equivalent to 32 in decimal, is set.

```
480  CALL IOSPOLL(VNA&, STAT%):
     CALL ERRORTRAP
```

Read in the status byte as an integer.

```
490 LOOP
500 END SUB
```

Return from the `WAITFORKEY` subroutine.

Running the program

1. The computer assumes that the port being calibrated is a 50 ohm type-N female test port and prompts the operator to connect each standard.
2. Connect the standards as prompted, and press any key on the front panel of the network analyzer to continue the program and measure the standard.
3. The program will display a message when the measurement calibration is complete.

Example 2B: Full 2-port calibration (HP 8753C only)

The following program illustrates how to perform a full 2-port measurement calibration on the network analyzer over HP-IB. The program does the calibration using the HP 85032B calibration kit. It steps the operator through the calibration by giving explicit directions on the network analyzer display and allowing the user to continue the program from the network analyzer front panel. The desired instrument state should be set up before the program is run. The main difference between this example and Example 2A is that in this case the calibration process allows removal of both the forward and reverse error terms. This permits measurement of all four S-parameters of the device under test. Port 1 is a female test port and port 2 is a male test port.

This example program is stored on the Example Programs disk as **IPG2B.BAS**.

`10 DECLARE SUB ERRORTRAP ()`	Define a subroutine to trap errors.
`20 DECLARE SUB IOOUTS (A$, ADDRESS&)`	Define a subroutine to send a command string from the computer to the analyzer.
`30 DECLARE SUB WAITFORKEY (LABEL$, VNA&, DISPLAY&, ISC&)`	Define a subroutine to display a message on the analyzer and wait for the operator to press a key.
`40 REM $INCLUDE: 'QBSETUP'`	Call the QuickBASIC initialization file `QBSETUP`.
`50 CLS`	Clear the computer CRT.
`60 ISC& = 7`	Assign the interface select code to a variable.
`70 VNA& = 716`	Assign the analyzer's address to a variable.
`80 DISPLAY& = 717`	Assign the analyzer's display address to a variable.
`90 CALL IOTIMEOUT(ISC&, 10!): CALL ERRORTRAP`	Define a system time-out of ten seconds and perform error trapping.
`100 CALL IOABORT(ISC&): CALL ERRORTRAP`	Abort any HP-IB transfers and perform error trapping.
`110 CALL IOCLEAR(ISC&): CALL ERRORTRAP`	Clear the analyzer's HP-IB interface and perform error trapping.
`120 CALL IOEOI(ISC&, 0): CALL ERRORTRAP`	Disable the End-Or-Identify mode for transferring data and perform error trapping.
`130 CALL IOOUTS("CALKN50; MENUOFF; CLES; ESE64;", VNA&)`	Select the 50 ohm type-N cal kit, turn off the softkey menu, clear the status byte, and set up the status reporting system so that bit 6, User Request, of the Event Status Register is summarized by bit 5 of the status byte, allowing a key press to be detected by a serial poll.
`140 CALL IOOUTS("CALIFUL2;", VNA&)`	Open the calibration by calling for a full two-port calibration.
`150 CALL IOOUTS("REFL;", VNA&)`	Open the reflection calibration subsequence.
`160 CALL WAITFORKEY("CONNECT OPEN AT PORT 1", VNA&, DISPLAY&, ISC&)`	Ask for an open at port 1 and wait for the operator to connect it.
`170 CALL IOOUTS("CLASS11A; OPC?; STANB;", VNA&)`	Measure the open. Identify the specific standard (female test port) within the class using the command `STANB`, indicating the option at the second softkey from the top.
`180 CALL IOENTER(VNA&, REPLY!): CALL ERRORTRAP`	Wait for the standard to be measured.
`190 CALL WAITFORKEY("CONNECT SHORT AT PORT 1", VNA&, DISPLAY&, ISC&)`	Ask for a short at port 1 and wait for the operator to connect it.

`200 CALL IOOUTS("CLASS11B; OPC?; STANB;", VNA&)`	Measure the short. Identify the specific standard (female test port) within the class.
`210 CALL IOENTER(VNA&, REPLY!): CALL ERRORTRAP`	Wait for the standard to be measured.
`220 CALL WAITFORKEY("CONNECT LOAD AT PORT 1", VNA&, DISPLAY&, ISC&)`	Ask for a load at port 1 and wait for the operator to connect it.
`230 CALL IOOUTS("OPC?; CLASS11C;", VNA&)`	Measure the load. There are no options within this class, so `OPC?`, which always precedes the last command, comes first.
`240 CALL IOENTER(VNA&, REPLY!): CALL ERRORTRAP`	Wait for the standard to be measured.
`250 CALL WAITFORKEY("CONNECT OPEN AT PORT 2", VNA&, DISPLAY&, ISC&)`	Ask for an open at port 2 and wait for the operator to connect it.
`260 CALL IOOUTS("CLASS22A; OPC?; STANA;", VNA&)`	Measure the open. Identify the specific standard (male test port) within the class using the command `STANA`, indicating the option at the first softkey from the top.
`270 CALL IOENTER(VNA&, REPLY!): CALL ERRORTRAP`	Wait for the standard to be measured.
`280 CALL WAITFORKEY("CONNECT SHORT AT PORT 2", VNA&, DISPLAY&, ISC&)`	Ask for a short at port 2 and wait for the operator to connect it.
`290 CALL IOOUTS("CLASS22B; OPC?; STANA;", VNA&)`	Measure the short. Identify the specific standard (male test port) within the class.
`300 CALL IOENTER(VNA&, REPLY!): CALL ERRORTRAP`	Wait for the standard to be measured.
`310 CALL WAITFORKEY("CONNECT LOAD AT PORT 2", VNA&, DISPLAY&, ISC&)`	Ask for a load at port 2 and wait for the operator to connect it.
`320 CALL IOOUTS("OPC?; CLASS22C;", VNA&)`	Measure the load, noting that there are no options within this class.
`330 CALL IOENTER(VNA&, REPLY!): CALL ERRORTRAP`	Wait for the standard to be measured.
`340 CALL IOOUTS("OPC?; REFD;", VNA&)`	Close the reflection calibration subsequence.
`350 CLS : PRINT "COMPUTING REFLECTION CALIBRATION COEFFICIENTS"`	Display program progress on the computer CRT.
`360 CALL IOENTER(VNA&, REPLY!): CALL ERRORTRAP`	Wait for the analyzer to finish calculating the reflection calibration coefficients before continuing.
`370 CALL IOOUTS("TRAN;", VNA&)`	Open the transmission calibration subsequence.
`380 CLS : PRINT "OPENING TRANSMISSION CALIBRATION SUBSEQUENCE"`	Display program progress on the computer CRT.
`390 CALL WAITFORKEY("CONNECT THRU (PORT 1 TO PORT 2)", VNA&, DISPLAY&, ISC&)`	Ask for a thru and wait for the operator to connect it.
`400 CLS : PRINT "MEASURING FORWARD TRANSMISSION"`	Display program progress on the computer CRT.

`410 CALL IOOUTS("OPC?; FWDT;", VNA&)`	Measure forward transmission.
`420 CALL IOENTER(VNA&, REPLY!): CALL ERRORTRAP`	Wait for the standard to be measured.
`430 CALL IOOUTS("OPC?; FWDM;", VNA&)`	Measure forward load match.
`440 CALL IOENTER(VNA&, REPLY!): CALL ERRORTRAP`	Wait for the standard to be measured.
`450 CLS : PRINT "MEASURING REVERSE TRANSMISSION"`	Display program progress on the computer CRT.
`460 CALL IOOUTS("OPC?; REVT;", VNA&)`	Measure reverse transmission.
`470 CALL IOENTER(VNA&, REPLY!): CALL ERRORTRAP`	Wait for the standard to be measured.
`480 CALL IOOUTS("OPC?; REVM;", VNA&)`	Measure reverse load match.
`490 CALL IOENTER(VNA&, Reply!): CALL ERRORTRAP`	Wait for the standard to be measured.
`500 CALL IOOUTS("TRAD;", VNA&)`	Close the transmission calibration subsequence.
`510 CLS : INPUT "SKIP ISOLATION CALIBRATION? (Y/N) ", ANSWER$`	Ask the operator if the isolation part of the calibration is to be skipped.
`520 IF ((ANSWER$ = "Y") OR (ANSWER$ = "y")) THEN`	Skip the isolation part of the calibration.
`530 CALL IOOUTS("OMII;", VNA&)`	Tell the analyzer to omit the isolation part of the calibration.
`540 ELSE`	Do the isolation part of the calibration.
`550 CALL WAITFORKEY("ISOLATE TEST PORTS", VNA&, DISPLAY&, ISC&)`	Ask the operator to isolate the test ports.
`560 CALL IOOUTS("ISOL; AVERFACT10; AVERON;", VNA&)`	Open the isolation calibration subsequence. Turn averaging on with an averaging factor of ten.
`570 CLS : PRINT "MEASURING REVERSE ISOLATION"`	Display program progress on the computer CRT.
`580 CALL IOOUTS("OPC?; REVI;", VNA&)`	Measure reverse isolation.
`590 CALL IOENTER(VNA&, REPLY!): CALL ERRORTRAP`	Wait for the standard to be measured.
`600 CLS : PRINT "MEASURING FORWARD ISOLATION"`	Display program progress on the computer CRT.
`610 CALL IOOUTS("OPC?; FWDI;", VNA&)`	Measure forward isolation.
`620 CALL IOENTER(VNA&, REPLY!): CALL ERRORTRAP`	Wait for the standard to be measured.
`630 END IF`	
`640 CALL IOOUTS("ISOD; AVEROFF;", VNA&)`	Close the isolation calibration subsequence. Turn off averaging.
`650 CALL IOOUTS("PG;", DISPLAY&)`	Ensure that the user graphics are cleared by removing the last prompt.
`660 CLS : PRINT "COMPUTING CALIBRATION COEFFICIENTS"`	Display program progress on the computer CRT.

`670 CALL IOOUTS("DONE; OPC?; SAV2;", VNA&)`	Affirm the completion of the calibration and save it.
`680 CALL IOENTER(VNA&, REPLY!): CALL ERRORTRAP`	Wait until the network analyzer has computed the calibration coefficients before continuing.
`690 CLS : PRINT "FULL 2-PORT CALIBRATION COMPLETED. CONNECT TEST DEVICE.";`	Display program progress and instructions on the computer CRT.
`700 CALL IOOUTS("MENUON;", VNA&)`	Turn the softkey menu back on.
`710 CALL IOLOCAL(ISC&): CALL ERRORTRAP`	Return the analyzer to local mode and perform error trapping.
`720 END`	End program execution.
`730 SUB ERRORTRAP`	Define a subroutine to trap errors.
`740 IF PCIB.ERR <> NOERR THEN ERROR PCIB.BASERR`	Perform error trapping.
`750 END SUB`	Return from the `ERRORTRAP` subroutine.
`760 SUB IOOUTS (A$, ADDRESS&) STATIC`	Define a subroutine to send a command string from the computer to the analyzer.
`770 CALL IOOUTPUTS(ADDRESS&, A$, LEN(A$)): CALL ERRORTRAP`	Send the command string `A$` out to the analyzer and perform error trapping.
`780 END SUB`	Return from the `IOOUTS` subroutine.
`790 SUB WAITFORKEY (LABEL$, VNA&, DISPLAY&, ISC&) STATIC`	Define a subroutine to display a message on the analyzer and wait for the operator to press a key.
`800 CLS : PRINT LABEL$`	Display instructions on the computer CRT.
`810 CALL IOOUTS("PG; PU; PA 390,3600; PD; LB" + LABEL$ + "; PRESS ANY KEY WHEN READY." + CHR$(3), DISPLAY&)`	Write on the network analyzer's display: `PG` : PaGe; clears old user graphics. `PU` : Pen Up; prevents anything from being drawn. `PA` : Pen At; positions the logical pen. `PD` : Pen Down; enables drawing. `LB` : LaBel; writes the message on the display. The label must always be terminated by the ETX symbol, CHR$(3).
`820 CALL IOOUTS("ENTO;", VNA&)`	Clear the analyzer's entry area.
`830 CALL IOCLEAR(VNA&): CALL ERRORTRAP`	Clear the analyzer's HP-IB interface and perform error trapping.
`840 CALL IOOUTS("ESR?;", VNA&)`	Request the Event Status Register value from the analyzer.
`850 CALL IOENTER(VNA&, ESTAT!): CALL ERRORTRAP`	Receive the Event Status Register from the analyzer, thereby clearing the latched User Request bit so that old key presses will not trigger a measurement.
`860 CALL IOOUTS("ESE64;", VNA&)`	Ensure that the proper status reporting system is still in effect.
`870 STAT% = 0`	Initialize `STAT%` for entry into the `DO UNTIL` loop.
`880 DO UNTIL ((STAT% MOD 64) >31)`	Wait for a key press to be indicated by the setting of bit 5 of the status byte. `MOD 64` removes the effect of all higher value bits (bit 6 is equivalent to 64 in decimal), and `>31` ensures that bit 5, which is equivalent to 32 in decimal, is set.

890 CALL IOSPOLL(VNA&, STAT%): CALL ERRORTRAP	Read in the status byte as an integer.
900 LOOP	
910 CALL IOOUTS("PG;", DISPLAY&)	Clear the user graphics on the analyzer.
920 END SUB	Return from the WAITFORKEY subroutine.

Running the program

1. The computer assumes that the test ports being calibrated are 50 ohm type-N, port 1 being a female test port and port 2 being a male test port. Prompts to connect each standard appear just above the message line on the HP 8753C display.
2. Connect the standards as prompted, and press any key on the front panel of the network analyzer to continue the program and measure the standard. When the option of omitting the isolation calibration is given, press "Y" or "N" on the computer keyboard. If the isolation cal is performed, averaging is automatically employed to ensure a good calibration.
3. The program will display a message when the measurement calibration is complete.

Transferring data

Trace information can be read out of the analyzer in two ways. First, trace data can be read selectively using markers. This is preferable if only specific information is needed. Secondly, the entire trace can be read out. This is only necessary if all the trace data is needed. The process of transferring data can be divided into the following three steps:

1. Set up the receiving array. Trace data is represented inside the network analyzer as a real/ imaginary component pair for each point. The receiving array for marker data must store three values: this real/imaginary component pair as well as a stimulus value. See Table 1 to identify the first two values according to the current display format and marker mode. The receiving array for reading in an entire trace must be two components wide and the number of points long in order to accommodate all of the trace data. Since QuickBASIC stores data by column and therefore fills the first array dimension first, make the first dimension of the receiving array correspond to the number of elements per point (e.g. 2) and the second dimension correspond to the number of points (e.g. 201). In addition, because a four-byte header is sent out before the trace data when reading in an entire trace in all formats except *form 4*, at least one extra real number or two extra integers must be allocated at the beginning of the receiving array in order to maintain data order. Although this four-byte header can be read in as one real number or as two integers, the four bytes are actually meant to be two ASCII characters and one integer. The first two bytes are the ASCII characters "#A" that indicate that a fixed length block transfer follows. The last two bytes form an integer containing the number of bytes in the block to follow.

2. Request the data from the network analyzer. For marker data, this is always done by the command `OUTPMARK`. For an entire trace, the desired data format and level must be specified. The analyzer can transmit data over HP-IB in five different formats, three of which are shown in the following example programs. The level of the data is determined by the `OUTPxxxx` command used. (See Figure 1.) The different data levels are as follows:

   - Raw data is the basic measurement data. It reflects the stimulus parameters, IF averaging, and IF bandwidth, and is read out with the four `OUTPRAWx` commands. Normally, only `OUTPRAW1` is available, and it sends out the current active parameter; however, if a full 2-port measurement calibration is on, all four `OUTPRAWx` commands are available. The four arrays correspond to S11, S21, S12, and S22, respectively, and the data is in real/imaginary component pairs.

   - Error-corrected data is the raw data with error correction applied. This data is read out with the command `OUTPDATA`, which reads active trace data, or the command `OUTPMEMO`, which reads the error corrected trace memory, if available. The data is for the current active parameter and is in real/imaginary component pairs. Neither raw nor error-corrected data reflect such post-processing functions as electrical delay offset, trace math, or time domain gating.

   - Formatted data, read out by the command `OUTPFORM`, is the data being displayed by the analyzer and reflects all post-processing functions. See Table 1 to identify the array values according to the current display format and marker mode.

   - Calibration coefficient data is the error correction arrays resulting from a calibration. Each array corresponds to a specific error term in the error model, and the data is stored as real/imaginary component pairs. The HP-IB Quick Reference details which error coefficients are used for specific calibration types and which arrays those coefficients are to be found in. Not all calibration types use all twelve arrays.

Because formatted data is seen on the analyzer display, it is generally the most useful. However, if post-processing is not necessary, as may be the case with smoothing, error-corrected data is more desirable.

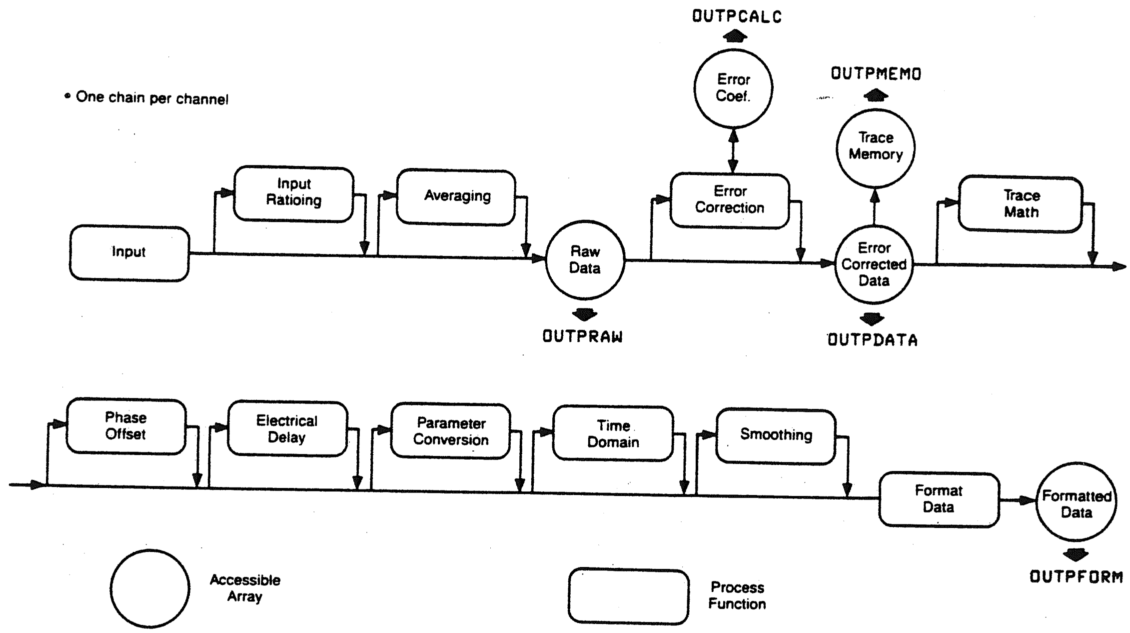


Figure 1. Data processing chain

3. Set all receiving parameters, and receive the data into the array. The receiving parameters and the type of data read in depend on which I/O routine will be used to receive the array. The three parameters in the computer that it may be necessary to initialize are as follows:

   - `MAX%`: the maximum number of items to be read. This includes the data and the header for all data formats except *form 4*. See Table 2 to determine whether `MAX%` is to specify a number of real numbers or a number of bytes according to the entering I/O routine used.

   - `ACTUAL%`: the actual number of items read. This is set by the I/O routine and should be initialized to zero.

   - `FLAG%`: the code set to indicate how transferred bytes are to be placed into memory. For example, `FLAG% = 1` means that bytes will be put into consecutive memory locations; `FLAG% = 4` means that every four bytes will be reversed in memory. See Table 2 to identify the entering I/O routines that use `FLAG%` as a parameter.

In general, the entering I/O routine must be sent a segment address indicating the place in memory to start storing data. If there is a four-byte header to be read in, this address should be one real number or two integers (four bytes) before the desired destination of the true data. For example, an array to hold the data for a 201-point trace with two real numbers per point might be allocated as `DAT!(1 TO 2, 1 TO 201)`. In order to account for the header, it should instead be dimensioned as `DAT!(1 TO 2, 0 TO 201)`, which will add two real numbers to the beginning of the array. Since only one of these is needed to store the four-byte header, the starting address specified in the entering I/O routine should only include one of them in the array: `SEG DAT!(2, 0)`. The result of this is that `DAT!(1, 0)` will be empty, `DAT!(2, 0)` will store the header, and `DAT!(1, 1)` will store the first real number of the data. See Table 2 for a summary of all entering I/O routines. For more information, refer to the *HP-IB Command Library Manual*.

Table 1. Units as a Function of Display Format

DISPLAY FORMAT	MARKER MODE	OUTPMARK value 1, value 2	OUTPFORM value 1, value 2	MARKET READOUT** value, aux value
LOG MAG		dB,*	dB,*	dB,*
PHASE		degrees,*	degrees,*	degrees,*
DELAY		seconds,*	seconds,*	seconds,*
SMITH CHART	LIN MKR	lin mag, degrees	real, imag	lin mag, degrees
	LOG MKR	dB, degrees	"	dB, degrees
	Re/Im	real, imag	"	real, imag
	R + jX	real, imag ohms	"	real, imag ohms
	G + jB	real, imag Siemens	"	real, imag Siemens
POLAR	LIN MKR	lin mag, degrees	real, imag	lin mag, degrees
	LOG MKR	dB, degrees	"	dB, degrees
	Re/Im	real, imag	"	real, imag
LIN MAG		lin mag,*	lin mag,*	lin mag,*
REAL		real,*	real,*	real,*
SWR		SWR,*	SWR,*	SWR,*

* Value not significant in this format, but is included in data transfers.

** The marker readout values are the marker values displayed in the upper right-hand corner of the display. They also correspond to the value and aux value associated with the fixed marker.

Table 2. Entering IO Routine Summary

ROUTINE	DATA TYPE	MAX%	FLAG %
IOENTER	one real	—	no
IOENTERA	array of reals	number of reals	no
IOENTERAB	unformatted	number of bytes*	yes
IOENTERB	unformatted	number of bytes	yes
IONETERS	character string	number of characters	no

* IOENTERAB will only read out as many bytes as are indicated by the last two bytes of the header (the number of bytes in the block to follow). However, if MAX% is less than this number, the transfer will terminate once MAX% bytes have been read out (MAX% is used as a safeguard to prevent longer-than-anticipated data from over-running the data array).

Example 3A: Data transfer using form 4, ASCII transfer format

The following program illustrates how to transfer data using *form 4*. *Form 4* transfers two numbers for each trace point, each number of the transfer data as a 24-character string, each character being a digit, sign, or decimal point. *Form 4* does not use a header. The first of two eleven-point transfers uses `OUTPFORM` to read out magnitude data. This eleven-point transfer with two real numbers per point and 24 bytes per point takes 528 (11*2*24) bytes. The second transfer uses `OUTPLIML` to read out limit data. (`OUTPLIML` reads out the stimulus frequency, result, upper limit, and lower limit of limit data.) Note that stimulus values can be read using this command even though no limits have been set. This eleven-point transfer with four real numbers per point and 24 bytes per point takes 1056 (11*4*24) bytes.

This example program is stored on the Example Programs disk as **IPG3A.BAS**.

`10 REM $INCLUDE: 'QBSETUP'`	Call the QuickBASIC initialization file `QBSETUP`.
`20 CLS`	Clear the computer CRT.
`30 ISC& = 7`	Assign the interface select code to a variable.
`40 VNA& = 716`	Assign the analyzer's address to a variable.
`50 CONST SIZE% = 11`	Set a constant to the number of points to be used in the trace.
`60 CALL IOTIMEOUT(ISC&, 10!): GOSUB ERRORTRAP`	Define a system time-out of ten seconds and perform error trapping.
`70 CALL IOABORT(ISC&): GOSUB ERRORTRAP`	Abort any HP-IB transfers and perform error trapping.
`80 CALL IOCLEAR(ISC&): GOSUB ERRORTRAP`	Clear the analyzer's HP-IB interface and perform error trapping.
`90 CALL IOEOI(ISC&, 0): GOSUB ERRORTRAP`	Disable the End-Or-Identify mode for transferring data and perform error trapping.
`100 A$ = "PRES;": GOSUB IOOUTS`	Preset the network analyzer.
`110 DIM DAT!(1 TO 2, 1 TO SIZE%), STIM!(1 TO 4, 1 TO SIZE%)`	Prepare arrays to receive the data. All `IOENTER` routines that fill arrays do so column by column. For example DAT! will be filled in the order `DAT!(1,1)`, `DAT!(2,1)`, `DAT!(1,2)`, etc. Noting this, dimension the array such that the data will be properly grouped.
`120 A$ = "POIN " + STR$(SIZE%) + "; SING; FORM4; OUTPFORM;": GOSUB IOOUTS`	Set the number of points in the trace to `SIZE%`, sweep once, and then hold. Tell the analyzer to send out formatted data in *form 4*, the ASCII transfer format.
`130 MAX% = 2 * SIZE%`	The maximum number of real numbers to be read in is two per point with `SIZE%` points.
`140 ACTUAL% = 0`	Initialize the actual number of real numbers read in. This variable is given a value by `IOENTERA`.
`150 CALL IOENTERA(VNA&, SEG DAT!(1, 1), MAX%, ACTUAL%): GOSUB ERRORTRAP`	Read the trace data into the array. The first field is the magnitude in dB.
`160 A$ = "OUTPLIML;": GOSUB IOOUTS`	Tell the analyzer to send out the limit test data for each point.
`170 MAX% = 4 * SIZE%`	The maximum number of real numbers to be read in during the next transfer is four per point with `SIZE%` points.
`180 ACTUAL% = 0`	Re-initialize the actual number of real numbers read in.

`190 CALL IOENTERA(VNA&, SEG STIM!(1, 1), MAX%, ACTUAL%): GOSUB ERRORTRAP`	Read the trace data into the array. The first field is the frequency in Hz.
`200 PRINT TAB(5); "#"; TAB(13); "MAGNITUDE"; TAB(27); "FREQUENCY"`	
`210 PRINT TAB(15); "(dB)"; TAB(29); "(Hz)": PRINT`	Display the table heading.
`220 FOR I% = 1 TO SIZE%`	Display the data for each trace point in a table on the computer CRT.
`230 PRINT USING "######"; I%;`	Display the trace point index in the desired format. For an explanation of QuickBASIC format statements, see the section entitled *Formatting Numbers* in *Microsoft QuickBASIC: Basic Language Reference*.
`240 PRINT "  "; : PRINT USING "+###.######";DAT!(1,I%);`	Display the trace point magnitude in the desired format.
`250 PRINT "  "; : PRINT USING "##.##^^^^";STIM!(1,I%)`	Display the trace point frequency in the desired format.
`260 NEXT I%`	
`270 CALL IOLOCAL(ISC&): GOSUB ERRORTRAP`	Return the network analyzer to local mode and perform error trapping.
`280 END`	End program execution.
`290 ERRORTRAP:`	Define a routine to trap errors.
`300 IF PCIB.ERR <> NOERR THEN ERROR PCIB.BASERR`	Perform error trapping.
`310 RETURN`	Return from the `ERRORTRAP` routine.
`320 IOOUTS:`	Define a routine to send a command string from the computer to the analyzer.
`330 CALL IOOUTPUTS(VNA&, A$, LEN(A$)): GOSUB ERRORTRAP`	Send the command string `A$` out to the analyzer and perform error trapping.
`340 RETURN`	Return from the `IOOUTS` routine.

Running the program

1. The computer presets the analyzer and resets the trace to eleven points.
2. The computer reads in the trace data requested by `OUTPFORM`. The first number for each point is the magnitude in dB. Regardless of the number of significant digits transmitted, the network analyzer only measures magnitude to a resolution of 0.001 dB, phase to 0.01 degrees, and group delay to 0.01 psec.
3. The computer reads in the trace data read out by `OUTPLIML`. The first number for each point is the frequency in Hz.
4. The computer displays the magnitude and frequency at the eleven points of the trace in a table.

Example 3B: Data transfer using form 5, PC-DOS 32-bit floating point format

The following program illustrates how to transfer data using *form 5*. *Form 5* transfers two numbers for each trace point, each number as a four-byte real number, and it uses a header, so the receiving array DAT! is set up to accommodate it. One 201-point transfer is done using OUTPFORM to read out magnitude data. This 201-point transfer with two real numbers per point and four bytes per point plus a four-byte header takes 1612 (201*2*4+4) bytes. Note that this same transfer in *form 4* would take 9648 (201*2*24) bytes.

This example program is stored on the Example Programs disk as **IPG3B.BAS**.

`10 REM $INCLUDE: 'QBSETUP'`	Call the QuickBASIC initialization file QBSETUP.
`20 CLS`	Clear the computer CRT.
`30 ISC& = 7`	Assign the interface select code to a variable.
`40 VNA& = 716`	Assign the analyzer's address to a variable.
`50 CALL IOTIMEOUT(ISC&, 10!): GOSUB ERRORTRAP`	Define a system time-out of ten seconds and perform error trapping.
`60 CALL IOABORT(ISC&): GOSUB ERRORTRAP`	Abort any HP-IB transfers and perform error trapping.
`70 CALL IOCLEAR(ISC&): GOSUB ERRORTRAP`	Clear the analyzer's HP-IB interface and perform error trapping.
`80 CALL IOEOI(ISC&, 0): GOSUB ERRORTRAP`	Disable the End-Or-Identify mode for transferring data and perform error trapping.
`90 DIM DAT!(1 TO 2, 0 TO 201)`	Prepare an array to receive the data, leaving at least four bytes of space before the desired data destination to account for the two-integer header.
`100 A$ = "SING; FORM5; OUTPFORM;": GOSUB IOOUTS`	Sweep once and then hold. Tell the analyzer to send out formatted data in *form 5*, PC-DOS 32-bit floating point.
`110 MAX% = 201 * 4 * 2 + 4`	The maximum number of bytes to be read in is two 4-byte real numbers per point with 201 points plus a four-byte (two-integer) header.
`120 ACTUAL% = 0`	Initialize the actual number of bytes read in. This variable is given a value by IOENTERB.
`130 FLAG% = 1`	No swapping of bytes is desired.
`140 CALL IOENTERB(VNA&, SEG DAT!(2, 0), MAX%, ACTUAL%, FLAG%): GOSUB ERRORTRAP`	Read in the data, specifying the beginning array address as one real number (four bytes) before the desired destination of the true data in order to account for the header and therefore maintain data grouping.
`150 PRINT USING "+###.#####"; DAT!(1, 1); DAT!(1, 201)`	Display the first and last data point values. Only the first value of the pair of numbers for each point (the magnitude in dB) is significant.
`160 A$ = "CONT;": GOSUB IOOUTS`	Restore continuous sweep trigger mode to the analyzer.
`170 CALL IOLOCAL(ISC&): GOSUB ERRORTRAP`	Return the network analyzer to local mode and perform error trapping.
`180 END`	End program execution.
`190 ERRORTRAP:`	Define a routine to trap errors.
`200 IF PCIB.ERR <> NOERR THEN ERROR PCIB.BASERR`	Perform error trapping.
`210 RETURN`	Return from the ERRORTRAP routine.
`220 IOOUTS:`	Define a routine to send a command string from the computer to the analyzer.

`230 CALL IOOUTPUTS(VNA&, A$, LEN(A$)): GOSUB ERRORTRAP`	Send the command string `A$` out to the analyzer and perform error trapping.
`240 RETURN`	Return from the `IOOUTS` routine.

Running the program

1. The computer reads in the trace data requested by `OUTPFORM` in *form 5*. The first number for each point is the magnitude in dB.
2. The computer displays the first and last magnitude values read in.

   Now go to the analyzer and press **[MENU]** *[NUMBER OF POINTS]* **[4] [0] [1] [x1]**. Run the program again. Note that although the program does not generate an error, only half of the data was read in since the computer only expected the data for 201 points. In this case the analyzer is still waiting to transfer data.

   Now change the number of points to 101. Run the program again. Note that a QuickBASIC error was generated since the analyzer ran out of data to transmit before the computer received the data from 201 points that it was expecting.

   It is imperative that the receiving array be correctly dimensioned. Fortunately, this is easy to ensure because not only is the number of points in the analyzer's trace readily available through `POIN?`, but the size of the transfer block is also easily determined from the header. In addition, QuickBASIC allows dimension statements anywhere in a program, so it is possible to wait until the size of the transfer is known to dimension the receiving array.

The above example program can be modified to take advantage of this by making the following changes:

- Change line 90 to the following:

`90 DIM HEADER%(0 TO 1)`	Prepare an array to receive the two-integer header.

- Delete line 110.
- Insert the following lines between lines 100 and 120:

`102 MAX% = 4`	The maximum number of bytes to be read in is only the four byte header.
`105 ACTUAL% = 0`	Initialize the actual number of bytes read in. This variable is given a value by `IOENTERB`.
`108 FLAG% = 1`	No swapping of bytes is desired.
`110 CALL IOENTERB(VNA&, SEG HEADER%(0), MAX%, ACTUAL%, FLAG%): GOSUB ERRORTRAP`	Read in the header as two integers. The second integer is the number of bytes of the trace data that would follow if `MAX%` were not set to read in only the header.
`112 DIM DAT!(1 TO 2, 0 TO HEADER%(1) / 8)`	Prepare an array to receive the data. The necessary size of the array can be determined from the known number of bytes of the trace data. (There are `HEADER%(1)` bytes with four bytes per real number and two real numbers per point.)
`115 A$ = "OUTPFORM;": GOSUB IOOUTS`	Tell the analyzer to send out data formatted data in *form 5*, PC-DOS 32-bit floating point.
`118 MAX% = HEADER%(1) + 4`	The maximum number of bytes to be read in is the number of bytes following the header, given by `HEADER%(1)`, plus the four bytes in the header.

This modified program is stored on the Example Programs disk as **IPG3BX.BAS**.

Two transfers are done using `OUTPFORM`. The first transfer reads in only the four-byte header (as two integers) before it terminates. The second of these integers is the size in bytes of the block of data to follow, and with this the receiving array can be correctly dimensioned regardless of the number of points in the trace.

Example 3C: Data transfer using form 1, network analyzer internal format

The following program illustrates how to transfer data using *form 1. Form 1* transfers a six-byte binary string of data for each trace point. The six bytes can be represented as three integers, and *form 1* uses a four-byte header, which can be read in as two integers, so the receiving array `DAT!` is set up to accommodate this. One transfer is done using `OUTPDATA` to determine the size of the data block. The receiving array is then correctly dimensioned, and a second transfer is done using `OUTPDATA` to receive all of the trace data. If there is a 201-point trace, with six-bytes per point plus a four-byte header, this transfer takes only 1210 (201*6+4) bytes. This is considerably faster than the same transfer in either *form 4* or *form 5.*

However, the data received in *form 1* is difficult to decode. Real/imaginary data uses the first two bytes for the imaginary fraction mantissa, the middle two bytes for the real fraction mantissa, the fifth byte for additional resolution when transferring raw data, and the last byte as the common power of two. The data could be recombined and displayed on the computer, but since this requires reformatting time, *form 1* is most useful for getting data to store on disk, as shown in the following program.

This example program is stored on the Example Programs disk as **IPG3C.BAS**.

`10 REM $INCLUDE: 'QBSETUP'`	Call the QuickBASIC initialization file `QBSETUP`.
`20 CLS`	Clear the computer CRT.
`30 ISC& = 7`	Assign the interface select code to a variable.
`40 VNA& = 716`	Assign the analyzer's address to a variable.
`50 CALL IOTIMEOUT(ISC&, 10!): GOSUB ERRORTRAP`	Define a system time-out of ten seconds and perform error trapping.
`60 CALL IOABORT(ISC&): GOSUB ERRORTRAP`	Abort any HP-IB transfers and perform error trapping.
`70 CALL IOCLEAR(ISC&): GOSUB ERRORTRAP`	Clear the analyzer's HP-IB interface and perform error trapping.
`80 CALL IOEOI(ISC&, 0): GOSUB ERRORTRAP`	Disable the End-Or-Identify mode for transferring data and perform error trapping.
`90 DIM HEADER%(0 TO 1)`	Prepare an array to receive the four-byte header as two integers.
`100 A$ = "SING; FORM1; OUTPDATA;": GOSUB IOOUTS`	Sweep once and then hold. Tell the analyzer to send out corrected data in *form 1,* instrument internal binary.
`110 MAX% = 4`	The maximum number of bytes to be read in is only the four-byte header.
`120 ACTUAL% = 0`	Initialize the actual number of bytes read in. This variable is given a value by `IOENTERB`.
`130 FLAG% = 4`	Reverse every four bytes.
`140 CALL IOENTERB(VNA&, SEG HEADER%(0), MAX%, ACTUAL%, FLAG%): GOSUB ERRORTRAP`	Read in the header as two integers. The first integer is the number of bytes of the trace data that would follow if `MAX%` were not set to read in only the header.
`150 DIM DAT%(1 TO 3, 0 TO HEADER%(0) / 6)`	Prepare an array to receive the data. The necessary size of the array can be determined from the known number of bytes of the trace data. (In addition to one four-byte header, there are six bytes per point in *form 1,* so allocate three integers per point.)
`160 A$ = "OUTPDATA;": GOSUB IOOUTS`	Tell the analyzer to send out corrected data in *form 1,* instrument internal binary.

`170 MAX% = HEADER%(0) + 4`	The maximum number of bytes to be read in is the number of bytes following the header, given by `HEADER%(0)`, plus four bytes in the header.
`180 ACTUAL% = 0`	Re-initialize the actual number of bytes read in.
`190 FLAG% = 1`	Because the data is only going to be stored in a file and not seen, no swapping of bytes is necessary.
`200 CALL IOENTERB(VNA&, SEG DAT%(2, 0), MAX%, ACTUAL%, FLAG%): GOSUB ERRORTRAP`	Read in the data, specifying the beginning array address as two integers (four bytes) before the desired destination of the true data in order to account for the header and therefore maintain data grouping.
`210 OPEN "TESTDATA" FOR BINARY AS #1`	Open the binary storage file.
`220 PUT #1, , HEADER%(0)`	Store the number of bytes of the trace data in the storage file.
`230 PUT #1, , DAT%(2, 0)`	Store the four-byte header in the storage file as two integers.
`240 PUT #1, , DAT%(3, 0)`	
`250 FOR I% = 1 TO HEADER%(0) / 6`	
`260  PUT #1, , DAT%(1, I%)`	Store the trace data in the storage file.
`270  PUT #1, , DAT%(2, I%)`	
`280  PUT #1, , DAT%(3, I%)`	
`290 NEXT I%`	
`300 CLOSE #1`	Close the storage file.
`310 PRINT "CHANGE SETUP AND PRESS <ENTER>."`	Display instructions on the computer CRT.
`320 DO UNTIL INKEY$ = CHR$(13): LOOP`	Wait for the operator to change the trace.
`330 OPEN "TESTDATA" FOR BINARY AS #1`	Open the binary storage file.
`340 GET #1, , HEADER%(0)`	Read the number of bytes of trace data from the storage file.
`350 GET #1, , DAT%(2, 0)`	Read the header from the storage file.
`360 GET #1, , DAT%(3, 0)`	
`370 FOR I% = 1 TO (HEADER%(0) / 6)`	
`380  GET #1, , DAT%(1, I%)`	Read the trace data from the storage file.
`390  GET #1, , DAT%(2, I%)`	
`400  GET #1, , DAT%(3, I%)`	
`410 NEXT I%`	
`420 CLOSE #1`	Close the storage file.
`430 A$ = "SING;": GOSUB IOOUTS`	Sweep once to view the current setup's trace on the analyzer and then hold.
`440 PRINT "PRESS <ENTER> TO CONTINUE.": DO UNTIL INKEY$ = CHR$(13): LOOP`	Allow the operator to view the current setup's trace before continuing.
`450 A$ = "INPUDATA;": GOSUB IOOUTS`	Prepare the analyzer to read in corrected data.
`460 MAX% = HEADER%(0) + 4`	The maximum number of bytes to be sent out is the number of bytes following the header, given by `HEADER%(0)`, plus the four bytes in the header.
`470 FLAG% = 1`	No swapping of bytes is desired.

`480 CALL IOOUTPUTB(VNA&, SEG DAT%(2, 0), MAX%, FLAG%): GOSUB ERRORTRAP`	Send out the data, specifying the beginning array address as two integers (four bytes) before the address where the true data is stored in order to account for the header.
`490 KILL "TESTDATA"`	Delete the data file.
`500 CALL IOLOCAL(ISC&): GOSUB ERRORTRAP`	Return the network analyzer to local mode and perform error trapping.
`510 END`	End program execution.
`520 ERRORTRAP:`	Define a routine to trap errors.
`530 IF PCIB.ERR <> NOERR THEN ERROR PCIB.BASERR`	Perform error trapping.
`540 RETURN`	Return from the `ERRORTRAP` routine.
`550 IOOUTS:`	Define a routine to send a command string from the computer to the analyzer.
`560 CALL IOOUTPUTS(VNA&, A$, LEN(A$)): GOSUB ERRORTRAP`	Send the command string `A$` out to the analyzer and perform error trapping.
`570 RETURN`	Return from the `IOOUTS` routine.

Running the program

1. The computer initiates a transfer using `OUTPDATA`, reads in the four-byte header as two integers, and terminates the transfer. The second of these integers is the size in bytes of the block of data to follow, and with this, the receiving array is correctly dimensioned.
2. The computer reads in all the trace data requested by `OUTPDATA`.
3. The computer stores the size of the block of data and the data in the hard disk file `TESTDATA`. If a hard disk is not available, change the file name on lines 210 and 330 to `A:TESTDATA`, and make sure that there is a formatted non-write-protected) disk in the A: drive.
4. Change the setup on the analyzer as prompted by the computer by, for example, disconnecting the test device.
5. The computer reads the trace data back in from the storage file, sends the data out to the analyzer, and deletes the storage file.

Example 3D: Data transfer using markers

The following program illustrates how to transfer data using markers and the command `OUTPMARK`. In order to read data off a trace using a marker, the marker must first be made active and put at the desired frequency using a command to select a specific stimulus value, like `MARK1 133.15MHZ`, or a command to do a marker search, like `MARK3; SEAMIN`. The command `OUTPMARK` tells the network analyzer to transmit three numbers: marker value one, marker value two, and marker stimulus value. See Table 1 (page 20) to identify the first two marker values according to the current display format. The third marker value, the stimulus value, is either frequency or time, depending on the network analyzer's active domain. These three values can be read in as an array of real numbers using the routine `IOENTERA`. In this case, there is no header, and `MAX%` is the maximum number of real numbers to read in (3).

This Example Program is stored on the Example Programs disk as **IPG3D.BAS**.

`10 REM $INCLUDE: 'QBSETUP'`	Call the QuickBASIC initialization file `QBSETUP`.
`20 CLS`	Clear the computer CRT.
`30 ISC& = 7`	Assign the interface select code to a variable.
`40 VNA& = 716`	Assign the analyzer's address to a variable.
`50 DISPLAY& = 717`	Assign the analyzer's display address to a variable.
`60 CALL IOTIMEOUT(ISC&, 10!): GOSUB ERRORTRAP`	Define a system time-out of ten seconds and perform error trapping.
`70 CALL IOABORT(ISC&): GOSUB ERRORTRAP`	Abort any HP-IB transfers and perform error trapping.
`80 CALL IOCLEAR(ISC&): GOSUB ERRORTRAP`	Clear the analyzer's HP-IB interface and perform error trapping.
`90 CALL IOEOI(ISC&, 0): GOSUB ERRORTRAP`	Disable the End-Or-Identify mode for transferring data and perform error trapping.
`100 DIM VALU!(0 TO 2)`	Allocate space to hold data read in from the analyzer.
`110 ADDRESS& = VNA&`	Initialize the output address to the address of the network analyzer.
`120 A$ = "PRES;": GOSUB IOOUTS`	Preset the network analyzer.
`130 A$ = "CHAN1; S21; LOGM;": GOSUB IOOUTS`	Make channel 1 the active channel and measure the magnitude of forward transmission parameter S21 in decibels.
`140 A$ = "CENT 134MHz;": GOSUB IOOUTS`	Set the center frequency to 134 MHz.
`150 A$ = "SPAN 25MHz;": GOSUB IOOUTS`	Set the frequency span to 25 MHz.
`160 A$ = "AUTO;": GOSUB IOOUTS`	Autoscale the resulting trace.
`170 A$ = "SING; MARK3; SEAMIN;": GOSUB IOOUTS`	Sweep once, hold, and set marker three at the minimum magnitude value of the trace.
`180 A$ = "MARK4; SEAMAX;": GOSUB IOOUTS`	Set marker four at the maximum magnitude value of the trace.
`190 A$ = "MARK1 133.15MHz; OUTPMARK;": GOSUB IOOUTS`	Set marker one at 133.15 MHz, sweep once, and request marker data from marker one. Since the format is log magnitude, only the first value (the magnitude at the marker in dB) and the third value (the frequency in Hz) read in are significant. → See Table 1.

`200 MAX% = 3`	Set the maximum number of real numbers to be read in from the analyzer.
`210 ACTUAL% = 0`	Initialize the actual number of real numbers read in. This variable is given a value by `IOENTERA`.
`220 CALL IOENTERA(VNA&, SEG VALU!(0), MAX%, ACTUAL%): GOSUB ERRORTRAP`	Read in marker data from the analyzer.
`230 PRINT "  MARKER AT 133.15 MHz:"`	
`240 PRINT "     FROM LOG MAGNITUDE PLOT:"`	Display a heading.
`250 PRINT TAB(15); VALU!(0); " DB"`	Display the magnitude value just read in.
`260 GOSUB WAITING`	Wait for the user to press any network analyzer key before continuing.
`270 A$ = "PHAS; AUTO;": GOSUB IOOUTS`	Display the phase of the active transmission parameter and autoscale the resulting trace.
`280 A$ = "MARK1; OUTPMARK;": GOSUB IOOUTS`	Request marker data from marker one. Since the format is phase, only the first value (the phase at the marker in degrees) and the third value (the frequency in Hz) read in are significant. → See Table 1. Note that a single sweep / hold is not necessary here because only format has changed.
`290 ACTUAL% = 0`	Re-initialize the actual number of real numbers read in.
`300 CALL IOENTERA(VNA&, SEG VALU!(0), MAX%, ACTUAL%): GOSUB ERRORTRAP`	Read in marker data from the analyzer.
`310 PRINT "     FROM PHASE PLOT:"`	Display a heading.
`320 PRINT TAB(15); VALU!(0); " DEGREES"`	Display the phase value just read in.
`330 GOSUB WAITING`	Wait for the user to press any network analyzer key before continuing.
`340 A$ = "LINM; AUTO;": GOSUB IOOUTS`	Display the linear magnitude of the active transmission parameter and autoscale the resulting trace.
`350 A$ = "MARK1; OUTPMARK;": GOSUB IOOUTS`	Request marker data from marker one. Since the format is linear magnitude, only the first value (the linear magnitude) and the third value (the frequency in Hz) read in are significant. → See Table 1.
`360 ACTUAL% = 0`	Re-initialize the actual number of real numbers read in.
`370 CALL IOENTERA(VNA&, SEG VALU!(0), MAX%, ACTUAL%): GOSUB ERRORTRAP`	Read in marker data from the analyzer.
`380 PRINT "     FROM LINEAR MAGNITUDE PLOT:"`	Display a heading.
`390 PRINT TAB(15); VALU!(0); " UNITS"`	Display the magnitude value just read in.
`400 GOSUB WAITING`	Wait for the user to press any network analyzer key before continuing.

410 A$ = "SMIC; AUTO; SMIMRX;": GOSUB IOOUTS	Display the Smith chart of the active transmission parameter and autoscale the trace. Set the marker data to be given in the form R + jX.
420 A$ = "MARK1; OUTPMARK;": GOSUB IOOUTS	Request marker data from marker one. In this configuration, the first value (real in ohms), the second value (imaginary in ohms), and the third value (the frequency in Hz) read in are significant. → See Table 1.
430 ACTUAL% = 0	Re-initialize the actual number of real numbers read in.
440 CALL IOENTERA(VNA&, SEG VALU!(0), MAX%, ACTUAL%): GOSUB ERRORTRAP	Read in marker data from the analyzer.
450 PRINT " FROM SMITH CHART:"	Display a heading.
460 PRINT TAB(15); VALU!(0); " + j "; VALU!(1); " OHMS"	Display the normalized impedance values just read in.
470 GOSUB WAITING	Wait for the user to press any network analyzer key before continuing.
480 A$ = "POLA; AUTO; POLMRI;": GOSUB IOOUTS	Display the active transmission parameter in polar form and autoscale the trace. Set the marker data to be in the form real/imaginary.
490 A$ = "MARK1; OUTPMARK;": GOSUB IOOUTS	Request marker data from marker one. In this configuration, the first value (real), the second value (imaginary), and the third value (the frequency in Hz) read in are significant. → See Table 1.
500 ACTUAL% = 0	Re-initialize the actual number of real numbers read in.
510 CALL IOENTERA(VNA&, SEG VALU!(0), MAX%, ACTUAL%): GOSUB ERRORTRAP	Read in marker data from the analyzer.
520 PRINT " FROM POLAR PLOT:"	Display a heading.
530 PRINT TAB(15); VALU!(0); " + j "; VALU!(1); " UNITS"	Display the values just read in.
540 CALL IOLOCAL(ISC&): GOSUB ERRORTRAP	Return the network analyzer to local mode and perform error trapping.
550 END	Perform error trapping.
560 ERRORTRAP:	Define a routine to trap errors.
570 IF PCIB.ERR <> NOERR THEN ERROR PCIB.BASERR	Perform error trapping.
580 RETURN	Return from the ERRORTRAP routine.
590 IOOUTS:	Define a routine to send a command string from the computer to the analyzer.
600 CALL IOOUTPUTS(ADDRESS&, A$, LEN(A$)): GOSUB ERRORTRAP	Send the command string A$ out to the analyzer and perform error trapping.
610 RETURN	Return from the IOOUTS routine.
620 WAITING:	Define a routine to display a prompt on the network analyzer's display and wait for the user to press any key before continuing.
630 ADDRESS& = DISPLAY&	Reset the output address to the network analyzer's display.

`640 A$ = "PU; PA 390,3600; PD; LBPRESS ANY KEY TO CONTINUE" + CHR$(3): GOSUB IOOUTS`	Write a prompt on the network analyzer's display.
`650 ADDRESS& = VNA&`	Return the output address to the network analyzer.
`660 A$ = "CLES; ESE64;": GOSUB IOOUTS`	Set up the status reporting system so that bit 6, User Request, of the Event Status Register is summarized by bit 5 of the Status Byte, allowing a key press to be detected by a serial poll.
`670 A$ = "ESR?;": GOSUB IOOUTS`	Request the Event Status Register value from the analyzer.
`680 CALL IOENTER(VNA&, ESTAT!): GOSUB ERRORTRAP`	Receive the Event Status Register value from the analyzer, thereby clearing the latched User Request bit so that old key presses will not trigger a measurement.
`690 STAT% = 0`	Initialize `STAT%` for entry into the `DO UNTIL` loop.
`700 DO UNTIL ((STAT% MOD 64) > 31)`	Wait for a key press to be indicated by the setting of bit 5 in the status byte. `MOD 64` removes the effect of all higher value bits (bit 6 is equivalent to 64 in decimal), and `> 31` ensures that bit 5, which is equivalent to 32 in decimal, is set.
`710 CALL IOSPOLL(VNA&, STAT%): GOSUB ERRORTRAP`	Read in the status byte as an integer.
`720 LOOP`	
`730 ADDRESS& = DISPLAY&`	Reset the output address to the network analyzer's display.
`740 A$ = "PG;": GOSUB IOOUTS`	Clear old user graphics from the network analyzer's display.
`750 ADDRESS& = VNA&`	Return the output address to the network analyzer.
`760 RETURN`	Return from the `WAITING` routine.

Running the program

1. The computer sets up a trace on the analyzer and puts markers at the maximum and minimum log magnitudes of the trace as well as at a specific frequency.

2. The computer reads in the data from marker one read out by `OUTPMARK`. Press any key on the analyzer front panel to continue the program, go on to a new display format, and read in its data from marker one. Note that only the identity of the first two marker data values varies with the current display format and marker mode; the command to read out the marker data, `OUTPMARK` and the number of values to be read (3) is always the same.