Errata

Title & Document Type: 8711B/12B/13B/14B Network Analyzers Programmer's Guide

Manual Part Number: 08713-90004

Revision Date: September1995

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.

Agilent Technologies

Programmer's Guide

HP 8711B/12B/13B/14B RF Network Analyzers

HP part number: 08713-90004
Printed in USA September, 1995

Notice

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

Hewlett-Packard makes no warranty of any kind with regard to this material, including but not limited to, the implied warranties of merchantability and fitness for a particular purpose. Hewlett-Packard shall not be liable for errors contained herein or for incidental or consequential damages in connection with the furnishing, performance, or use of this material.

Firmware Revision

This manual documents analyzers with firmware revisions B.03.50 and above. Some features (and therefore commands) will not be available in analyzers with earlier firmware revisions. For full compatibility, you can upgrade your firmware to the latest version. Contact your nearest Hewlett-Packard sales or service office for information.

HP-IB Programming

This document is an introduction to programming your analyzer over the Hewlett-Packard Interface Bus (HP-IB). Its purpose is to provide concise information about the operation of the instrument under HP-IB control. It provides some background information on the HP-IB and a tutorial introduction using programming examples to demonstrate the remote operation of the HP 8711. The examples are provided on two disks that are included with this guide. Both disks contain the same examples written in HP BASIC; only the disk format is different. These programs can run on the analyzer's internal controller (Option 1C2) or on an external controller.

- *Example Programs Disk — DOS Format* : part number 08712-10001
- *Example Programs Disk — LIF Format* : part number 08712-10002

You should become familiar with the operation of your network analyzer before controlling it over HP-IB. This document is not intended to teach programming or to discuss HP-IB theory except at an introductory level. Related information can be found in the following references. Contact the nearest HP sales office for ordering information. A list of HP sales and service offices can be found in the "Specifications and Characteristics" chapter of the *User's Guide*.

- Information on making measurements with the analyzer is available in the analyzer's *User's Guide*.
- Information on HP Instrument BASIC is available in the *HP Instrument BASIC User's Handbook*.
- Information on HP BASIC programming is available in the manual set for the BASIC revision being used. For example: *BASIC 6.0 Programming Techniques* and *BASIC 6.0 Language Reference*.
- Information on using the HP-IB is available in the *Tutorial Description of the Hewlett-Packard Interface Bus* (HP literature no. 5021-1927).

1

Introduction to HP-IB Programming

Introduction to HP-IB Programming

HP-IB — the Hewlett-Packard Interface Bus — is a high-performance bus that allows individual instruments and computers to be combined into integrated test systems. The bus and its associated interface operations are defined by the IEEE 488.1 standard. The IEEE 488.2 standard defines the interface capabilities of instruments and controllers in a measurement system, including some frequently used commands.

HP-IB cables provide the physical link between devices on the bus. There are eight data lines on each cable that are used to send data from one device to another. Devices that send data over these lines are called **Talkers**. **Listeners** are devices that receive data over the same lines. There are also five control lines on each cable that are used to manage traffic on the data lines and to control other interface operations. **Controllers** are devices that use these control lines to specify the talker and listener in a data exchange. When an HP-IB system contains more that one device with controller capabilities, only one of the devices is allowed to control data exchanges at any given time. The device currently controlling data exchanges is called the **Active Controller**. Also, only one of the controller-capable devices can be designated as the **System Controller**, the one device that can take control of the bus even if it is not the active controller. The network analyzer can act as a talker, listener, active controller or system controller at different times.

HP-IB addresses provide a way to identify devices on the bus. The active controller uses HP-IB addresses to specify which device talks and which device listens during a data exchange. This means that each device's address must be unique. A device's address is set on the device itself, using either a front-panel key sequence or a rear-panel switch.

To set the HP-IB address on the analyzer use the softkeys located in the (SYSTEM OPTIONS) HP-IB menu. The factory default address for the analyzer is 16.

NOTE

Throughout this manual, the following conventions are used:

Square brackets ([ ]) are used to enclose a keyword that is optional or implied when programming the command; that is, the instrument will process the command to have the same effect whether the option node is omitted or not.

Parameter types (< >) are distinguished by enclosing the type name in angle brackets.

A vertical bar (|) can be read as "or" and is used to separate alternative parameter options.

Bus Structure

Data Bus

The data bus consists of eight lines that are used to transfer data from one device to another. Programming commands and data sent on these lines is typically encoded in the ASCII format, 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 talk. When the bus is in remote and a device is addressed, the device receives instructions from HP-IB rather than from its front panel (pressing the `Return to Local` softkey 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.

Sending Commands

Commands are sent over the HP-IB via a controller's language system, such as IBASIC, QuickBASIC or C. The keywords used by a controller to send HP-IB commands vary among systems. When determining the correct keywords to use, keep in mind that there are two different kinds of HP-IB commands:

- Bus management commands, which control the HP-IB interface.
- Device commands, which control analyzer functions.

Language systems usually deal differently with these two kinds of HP-IB commands. For example, HP BASIC uses a unique keyword to send each bus management command, but always uses the keyword `OUTPUT` to send device commands.

The following example shows how to send a typical device command:

```
OUTPUT 716;"CALCULATE:MARKER:MAXIMUM"
```

This sends the command within the quotes (`CALCULATE:MARKER:MAXIMUM`) to the HP-IB device at address 716. If the device is an analyzer, the command instructs the analyzer to set a marker to the maximum point on the data trace.

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.

Interface Capabilities

The analyzer has the following interface capabilities, as defined by the IEEE 488.1 standard:

SH1	full Source handshake capability
AH1	full Acceptor handshake capability
T6	basic Talker, Serial Poll, no Talk Only, unaddress if MLA
TE0	no Extended Talker capability
L4	basic Listener, no Listen Only, unaddress if MTA
LE0	no Extended Listener capability
SR1	full Service Request capability
RL1	full Remote/Local capability
DC1	full Device Clear capability
C1	System Controller capability
C2	send IFC and take charge Controller capability
C3	send REN Controller capability
C4[1]	respond to SRQ
C8[1]	send IFC, receive control, pass control, pass control to self
C12[2]	send IF messages, receive control, pass control
E2	tri-state drivers
DT1	full device trigger capability
PP0	no parallel poll capability

1 only when an HP Instrument BASIC program is running

2 only when an HP Instrument BASIC program is not running

Programming Fundamentals

This section includes specific information for programming your network analyzer. It includes how the analyzer interacts with a controller, how data is transferred between the analyzer and a controller, and how to use the analyzer's status register structure to generate service requests.

Controller Capabilities

The analyzer can be configured as an HP-IB system controller or as a talker/listener on the bus. To configure the analyzer, select either the `System Controller` or the `Talker/Listener` softkey in the (SYSTEM OPTIONS) `HP-IB` menu.

The analyzer is not usually configured as the system controller unless it is the only controller on the bus. This setup would be used if the analyzer only needed to control printers or plotters. It would also be used if HP Instrument BASIC was being used to control other test equipment.

When the analyzer is used with another controller on the bus, it is usually configured as a talker/listener. In this configuration, when the analyzer is passed control it can function as the active controller.

Response to Bus Management Commands

The HP-IB contains an attention (ATN) line that determines whether the interface is in command mode or data mode. When the interface is in command mode (ATN TRUE) a controller can send bus management commands over the bus. Bus management commands specify which devices on the interface can talk (send data) and which can listen (receive data). They also instruct devices on the bus, either individually or collectively, to perform a particular interface operation.

This section describes how the analyzer responds to the HP-IB bus management commands. The commands themselves are defined by the IEEE 488.1 standard. Refer to the documentation for your controller's language system to determine how to send these commands.

Device Clear (DCL)

When the analyzer receives this command, it:

- Clears its input and output queues.
- Resets its command parser (so it is ready to receive a new program message).
- Cancels any pending `*OPC` command or query.

The command does not affect:

- Front panel operation.
- Any analyzer operations in progress (other than those already mentioned).
- Any instrument settings or registers (although clearing the output queue may indirectly affect the Status Byte's Message Available (MAV) bit).

Go To Local (GTL)

This command returns the analyzer to local (front-panel) control. All keys on the analyzer's front-panel are enabled.

Interface Clear (IFC)

This command causes the analyzer to halt all bus activity. It discontinues any input or output, although the input and output queues are not cleared. If the analyzer is designated as the active controller when this command is received, it relinquishes control of the bus to the system controller. If the analyzer is enabled to respond to a Serial Poll it becomes Serial Poll disabled.

Local Lockout (LLO)

This command causes the analyzer to enter the local lockout mode, regardless of whether it is in the local or remote mode. The analyzer only leaves the local lockout mode when the HP-IB's Remote Enable (REN) line is set FALSE.

Local Lockout ensures that the analyzer's remote softkey menu (including the `Return to LOCAL` softkey) is disabled when the analyzer is in the remote mode. When the key is enabled, it allows a front-panel operator to return the analyzer to local mode, enabling all other front-panel keys. When the key is disabled, it does not allow the front-panel operator to return the analyzer to local mode.

Parallel Poll

The analyzer ignores all of the following parallel poll commands:

- Parallel Poll Configure (PPC).
- Parallel Poll Unconfigure (PPU).
- Parallel Poll Enable (PPE).
- Parallel Poll Disable (PPD).

Remote Enable (REN)

REN is a single line on the HP-IB. When it is set TRUE, the analyzer will enter the remote mode when addressed to listen. It will remain in remote mode until it receives the Go to Local (GTL) command or until the REN line is set FALSE.

When the analyzer is in remote mode and local lockout mode, all front panel keys are disabled. When the analyzer is in remote mode but not in local lockout mode, all front panel keys are disabled except for the softkeys. The remote softkey menu includes seven keys that are available for use by a program. The eighth softkey is the `Return to LOCAL` key which allows a front-panel operator to return the analyzer to local mode, enabling all other front-panel keys.

Selected Device Clear (SDC)

The analyzer responds to this command in the same way that it responds to the Device Clear (DCL) command.

When the analyzer receives this command it:

- Clears its input and output queues.
- Resets its command parser (so it is ready to receive a new program message).
- Cancels any pending `*OPC` command or query.

The command does not affect:

- Front-panel operation.
- Any analyzer operations in progress (other than those already mentioned).
- Any analyzer settings or registers (although clearing the output queue may indirectly affect the Status Byte's MAV bit).

Serial Poll

The analyzer responds to both of the serial poll commands. The Serial Poll Enable (SPE) command causes the analyzer to enter the serial poll mode. While the analyzer is in this mode, it sends the contents of its Status Byte register to the controller when addressed to talk.

When the Status Byte is returned in response to a serial poll, bit 6 acts as the Request Service (RQS) bit. If the bit is set, it will be cleared after the Status Byte is returned.

The Serial Poll Disable (SPD) command causes the analyzer to leave the serial poll mode.

Take Control Talker (TCT)

If the analyzer is addressed to talk, this command causes it to take control of the HP-IB. It becomes the active controller on the bus. The analyzer automatically passes control back when it completes the operation that required it to take control. Control is passed back to the address specified by the `*PCB` command (which should be sent prior to passing control).

If the analyzer does not require control when this command is received, it immediately passes control back.

Message Exchange

The analyzer communicates with the controller and other devices on the HP-IB using program messages and response messages. Program messages are used to send commands, queries, and data to the analyzer.

Response messages are used to return data from the analyzer. The syntax for both kinds of messages is discussed in Chapter 10.

There are two important things to remember about the message exchanges between the analyzer and other devices on the bus:

- The analyzer only talks after it receives a terminated query (see "Query Response Generation" later in this section).
- Once it receives a terminated query, the analyzer expects to talk before it is told to do something else.

HP-IB Queues

Queues enhance the exchange of messages between the analyzer and other devices on the bus. The analyzer contains:

- An input queue.
- An error queue.
- An output queue.

Input Queue.

The input queue temporarily stores the following until they are read by the analyzer's command parser:

- Device commands and queries.
- The HP-IB END message (EOI asserted while the last data byte is on the bus).

The input queue also makes it possible for a controller to send multiple program messages to the analyzer without regard to the amount of time required to parse and execute those messages. The queue holds up to 128 bytes. It is cleared when:

- The analyzer is turned on.
- The Device Clear (DCL) or Selected Device Clear (SDC) command is received.

Error Queue.

The error queue temporarily stores up to 20 error messages. Each time the analyzer detects an error, it places a message in the queue. When you send the SYST:ERR? query, one message is moved from the error queue to the output queue so it can be read by the controller. Error messages are delivered to the output queue in the order they were received.

The error queue is cleared when:

- All the error messages are read using the SYST:ERR? query.
- The analyzer is turned on.
- The *CLS command is received.

Output Queue.

The output queue temporarily stores a single response message until it is read by a controller. It is cleared when:

- The message is read by a controller.
- The analyzer is turned on.
- The Device Clear (DCL) or Selected Device Clear (SDC) command is received.

Command Parser

The command parser reads program messages from the input queue in the order they were received from the bus. It analyzes the messages to determine what actions the analyzer should take.

One of the parser's most important functions is to determine the position of a program message in the analyzer's command tree (described in Chapter 10). When the command parser is reset, the next command it receives is expected to arise from the base of the analyzer's command tree.

The parser is reset when:

- The analyzer is turned on.
- The Device Clear (DCL) or Selected Device Clear (SDC) command is received.
- A colon immediately follows a semicolon in a program message. (For more information see "Sending Multiple Commands" in Chapter 10.)
- A program message terminator is received. A program message terminator can be an ASCII carriage return ($^{C}{}_{R}$) or newline character or the HP-IB END message (EOI set true).

Query Response Generation

When the analyzer parses a query, the response to that query is placed in the analyzer's output queue. The response should be read immediately after the query is sent. This ensures that the response is not cleared before it is read. The response is cleared when one of the following message exchange conditions occurs:

- Unterminated condition — the query is not properly terminated with an ASCII carriage return character or the HP-IB END message (EOI set true) before the response is read.
- Interrupted condition — a second program message is sent before the response to the first is read.
- Buffer deadlock — a program message is sent that exceeds the length of the input queue or that generates more response data than fits in the output queue.

2

Synchronizing the Analyzer and a Controller

Synchronizing the Analyzer and a Controller

The IEEE 488.2 standard provides tools that can be used to synchronize the analyzer and a controller. Proper use of these tools ensures that the analyzer is in a known state when you send a particular command or query.

Device commands can be divided into two broad classes:

- Sequential commands.
- Overlapped commands.

Most of the analyzer's commands are processed sequentially. A sequential command holds off the processing of subsequent commands until it has been completely processed.

Some commands do not hold off the processing of subsequent commands; they are called overlapped commands.

Overlapped Commands

Typically, overlapped commands take longer to process than sequential commands. For example, the `:INITIATE:IMMEDIATE` command restarts a measurement. The command is not considered to have been completely processed until the measurement is complete. This can take a long time with a narrow system bandwidth or when averaging is enabled.

The analyzer has the following overlapped commands:

ABORt
CALibration:ZERO:AUTO
CONFigure[1|2]
DIAGnostic:CCONstants:LOAD
DIAGnostic:CCONstants:STORe:DISK
DIAGnostic:CCONstants:STORe:EEPRom
DIAGnostic:DITHer
DIAGnostic:SPUR:AVOid
HCOPy[:IMMediate]
INITiate[1|2]:CONTinuous
INITiate[1|2][:IMMediate]
MMEMory:LOAD:STATe
OUTPut[:STATe]
PROGram[:SELected]:EXECute
SENSe[1|2]:AVERage:CLEar
SENSe[1|2]:AVERage:COUNt
SENSe[1|2]:AVERage[:STATe]
SENSe[1|2]:BWIDth[:RESolution]
SENSe[1|2]:CORRection:COLLect[:ACQuire]
SENSe[1|2]:CORRection:COLLect:ISTate[:AUTO]
SENSe[1|2]:CORRection:COLLect:METHod
SENSe[1|2]:CORRection:COLLect:SAVE
SENSe[1|2]:CORRection:CSET[:SELect]
SENSe[1|2]:CORRection[:STATe]
SENSe:COUPle
SENSe[1|2]:DETector[:FUNCtion]
SENSe[1|2]:DISTance:STARt (Option 100 only)
SENSe[1|2]:DISTance:STOP (Option 100 only)
SENSe[1|2]:FREQuency:CENTer
SENSe[1|2]:FREQuency:MODE (Option 100 only)

SENSe[1|2]:FREQuency:SPAN
SENSe[1|2]:FREQuency:SPAN:MAXimum
SENSe[1|2]:FREQuency:STARt
SENSe[1|2]:FREQuency:STOP
SENSe[1|2]:FUNCtion
SENSe[1|2]:FUNCtion:SRL:SCAN[:IMMediate] (Option 100 only)
SENSe:ROSCillator:SOURce
SENSe[1|2]:STATe
SENSe[1|2]:SWEep:POINts
SENSe[1|2]:SWEep:TIME
SENSe[1|2]:SWEep:TIME:AUTO
SENSe:SWEep:TRIGger:SOURce
SOURce[1|2]:POWer[:LEVel][:IMMediate][:AMPLitude]
SYSTem:PRESet
TRACe[:DATA]
TRIGger[:SEQuence]:SOURce

The analyzer uses a No Pending Operation (NPO) flag to keep track of overlapped commands. The NPO flag is reset to 0 when an overlapped command has not completed (still pending). It is set to 1 when no overlapped commands are pending. The NPO flag cannot be read directly but all of the following common commands take some action based on the setting of the flag.

*WAI Holds off the processing of subsequent commands until the NPO flag is set to 1. This ensures that commands in the analyzer's input queue are processed in the order received.

The program continues to run and additional commands are received and parsed by the analyzer (but not executed) while waiting for the NPO flag to be set. Use of the *WAI command is demonstrated in the SETUP example program.

*OPC? Places a 1 in the analyzer's output queue when the NPO flag is set to 1. If the program is designed to read the output queue before it continues, this effectively pauses the controller until all pending overlapped commands are completed. Use of the *OPC? command is demonstrated in the TRANCAL and REFLCAL example programs.

*OPC Sets bit 0 of the Standard Event Status event register to 1 when the NPO flag is set to 1. The analyzer's status registers can then be used to generate a service request when all pending overlapped commands are completed. This synchronizes the controller to the completion of an overlapped command, but also leaves the controller free to perform other tasks while the command is executing.

> **NOTE**
>
> *OPC only informs you when the NPO flag is set to 1. It does not hold off the processing of subsequent commands. No commands should be sent to the analyzer between sending the *OPC command and receiving the service request. Any command sent will be executed and may affect how the instrument responds to the previously sent *OPC.

The *CLS and *RST commands cancel any preceding *OPC command or query. Pending overlapped commands are still completed, but their completion is not reported in either the status register or the output queue. Two HP-IB bus management commands — Device Clear (DCL) and Selected Device Clear (SDC) — also cancel any preceding *OPC command or query.

> **NOTE**
>
> Use *WAI, *OPC? or *OPC whenever overlapped commands are used. A recommended technique is to send *WAI at the end of each group of commands.

CAUTION

ALWAYS trigger an individual sweep (using *OPC? and waiting for the reply) before reading data over the bus or executing a marker function. The analyzer has the ability to process the commands it receives faster than it can make a measurement. If the measurement is not complete when the data is read or a marker search function is executed the results are invalid.

The command to use (in an IBASIC OUTPUT statement) is:

```
OUTPUT @Hp8711;"ABOR;:INIT:CONT OFF;:INIT;*OPC?"
ENTER @Hp8711;Opc_done
```

or another form of the :INITiate[1|2][:IMMediate] command combined with the *OPC? query.

Refer to "Taking Sweeps" in Chapter 6 for more information.

3

Passing Control

Passing Control

When an external controller is connected to the analyzer with an HP-IB cable, passing control may be needed to control devices such as printers and plotters that are also connected on the HP-IB. For some operations the active controller must pass control to the analyzer. When the analyzer completes the operation, it automatically passes control of the bus back to the external controller.

An example program, `PASSCTRL`, demonstrates passing control to the analyzer. In this example program control is passed so the analyzer can control a printer for hardcopy output. See Chapter 8, "Example Programs."

NOTE

Pass Control is not needed to control peripherals connected to the serial or parallel ports.

For smooth passing of control, take steps that ensure the following conditions are met:

- The analyzer must know the controller's address so it can pass control back.
- The controller must be informed when the analyzer passes control back.

The following is a procedure for passing control:

1. Send the controller's HP-IB address to the analyzer with the `*PCB` command.
2. Clear the analyzer's status registers with the `*CLS` command.
3. Enable the analyzer's status registers to generate a service request when the Operation Complete bit is set. (Send `*ESE` with a value of 1 and `*SRE` with a value of 32.)
4. Enable the controller to respond to the service request.
5. Send the command that requires control of the bus followed by the `*OPC` command.
6. Pass control to the analyzer and wait for the service request. The service request indicates that the command has been completed and control has been passed back to the controller.

NOTE

For this procedure to work properly, only the command that requires control of the bus should be pending. Other overlapped commands should not. For more information on overlapped commands, see Chapter 2, "Synchronizing the Analyzer and a Controller."

4

Data Types and Encoding

Data Types and Encoding

Data is transferred between the analyzer and a controller via the HP-IB data lines, DIO1 through DIO8. Such transfers occur in a byte-serial (one byte at a time), bit-parallel (8 bits at a time) manner. This section discusses the following aspects of data transfer:

- The different data types used during data transfers.
- Data encoding used during transfers of numeric block data.

Data Types

The uses a number of different data types during data transfers. Data transfer occurs in response to a query. The data type used is determined by the parameter being queried. The different parameter types are described in the "Parameter Types" section of Chapter 10. Data types described in this section are:

- Numeric Data.
- Character Data
- String Data
- Expression Data
- Block Data

Numeric Data

The analyzer returns three types of numeric data in response to queries:

NR1 data — Integers (such as `+1`, `0`, `-1`, `123`, `-12345`). This is the response type for boolean parameters as well as some numeric parameters.

NR2 data — Floating point numbers with an explicit decimal point (such as `12.3`, `+1.234`, `-0.12345`).

NR3 data — Floating point numbers in scientific notation (such as `+1.23E+5`, `+123.4E-3`, `-456.789E+6`).

Character Data

Character data consists of ASCII characters grouped together in mnemonics that represent specific instrument settings (such as `MAXimum` , `MINimum` or `MLOGarithmic`). The analyzer always returns the short form of the mnemonic in upper-case alpha characters.

String Data

String data consists of ASCII characters. The string must be enclosed by a delimiter, either single quotes (`'This is string data.'`) or double quotes (`"This is also string data."`). To include the delimiter as a character in the string it must be typed twice without any characters in between. The analyzer always uses double quotes when it returns string data.

Expression Data

Expression data consists of mathematical expressions that use character parameters. When expression data is sent to the analyzer it is always enclosed in parentheses (such as `(IMPL/CH1SMEM)` or `(IMPL)`). The analyzer returns expression data enclosed in double quotes.

Block Data

Block data are typically used to transfer large quantities of related data (like a data trace). Blocks can be sent as definite length blocks or indefinite length blocks — the instrument will accept either form. The analyzer always returns definite length block data in response to queries.

Definite Block Length

The general form for a definite block length transfer is:

```
#<num_digits><num_bytes><data_bytes>
```

In the definite length block, two numbers must be specified. The single decimal digit <num_digits> specifies how many digits are contained in <num_bytes>. The decimal number <num_bytes> specifies how many data bytes will follow in <data_bytes>. An example IBASIC (or HP BASIC) statement to send ABC+XYZ as a definite block length parameter is shown, note that the data block contains seven bytes (7) and only one digit is needed to describe the block length 1.

```
OUTPUT 716;"#17ABC+XYZ"
```

NOTE

This analyzer will send an additional <CR> with EOI asserted for definite block length transfers. The definite length block form for your analyzer is:

```
#<num_digits><num_bytes><data_bytes><CR><EOI>
```

<num_bytes> is the number of <data_bytes> without counting <CR><EOI>.

Indefinite Block Length

The general form for an indefinite block length transfer is:

```
#0<data_bytes><CR><EOI>
```

After the last data byte is sent, the indefinite length block must be terminated by sending a carriage return or newline with EOI asserted. This forces the termination of the program message. An example IBASIC (or HP BASIC) statement to send `ABC+XYZ` as an indefinite block length parameter is shown, note that `,END` is used to properly terminate the message.

```
OUTPUT 716;"#0ABC+XYZ",END
```

Data Encoding for Large Data Transfers

The FORMat:DATA command selects the type of data and the type of data encoding that is used to transfer large blocks of numeric data between the analyzer and a controller. There are two specifiers:

REAL — specifies the block data type. Either the definite or indefinite length syntax can be used. The block is transferred as a series of binary-encoded floating-point numbers. Data transfers of the REAL,64 data type are demonstrated in the REALDATA example program.

INTeger — specifies the block data type. Either the definite or indefinite length syntax can be used. The block is transferred as an array of binary-encoded data with each point represented by a set of three 16-bit integers. This is the instrument's internal format — it should only be used for data that will be returned to the instrument for later use. Data transfers of the INTeger,16 data type are demonstrated in the INTDATA and LOADCALS example programs.

ASCii — specifies the numeric data type (NR1, NR2 or NR3 syntax). The data is transferred as a series of ASCII-encoded numbers separated by commas. ASCii formatted data transfers are demonstrated in the ASCDATA example program.

Blocks that contain mixed data — both numbers and ASCII characters — ignore the setting of FORMat:DATA. These blocks always transfer as either definite length or indefinite length block data. The following commands transfer blocks of mixed data:

```
PROGram[:SELected]:DEFine
SYSTem:SET
```

ASCII Encoding

The ANSI X3.4-1977 standard defines the ASCII 7-bit code. When an ASCII-encoded byte is sent over the HP-IB, bits 0 through 6 of the byte (bit 0 being the least significant bit) correspond to the HP-IB data lines DIO1 through DIO7. DIO8 is ignored.

When ASCII encoding is used for large blocks of data, the number of significant digits to be returned for each number in the block can be specified. For example, the following command returns all numbers as NR3 data with 7 significant digits.

```
FORMat:DATA ASCii,7
```

Binary Encoding

When binary encoding is used for large blocks of data, all numbers in the block are transferred as 32-bit or 64-bit binary floating point numbers or as an array of 16-bit integers. The binary floating-point formats are defined in the IEEE 754-1985 standard.

`FORMat:DATA REAL,32`···selects the IEEE 32-bit format (not supported by IBASIC or HP BASIC).

`FORMat:DATA REAL,64`···selects the IEEE 64-bit format.

`FORMat:DATA INTeger,16`···selects the 16-bit integer format.

Byte Swapping

PC compatibles frequently use a modification of the IEEE floating point formats with the byte order reversed. To reverse the byte order for data transfer into a PC, the `FORMat:BORDer` command should be used.

`FORMat:BORDer SWAPped`	*selects the byte-swapped format*
`FORMat:BORDer NORMal`	*selects the standard format*

5

Using Status Registers

Using Status Registers

The analyzer's status registers contain information about the condition of the network analyzer and its measurements. This section describes the registers and their use in HP-IB programming.

Example programs using the status registers are included in Chapter 8, "Example Programs." These programs include `SRQ` and `GRAPHICS` which use service request interrupt routines, `PASSCTRL` which uses the status byte to request control of the HP-IB and `LIMITEST` which uses the Limit Fail condition register.

General Status Register Model

The analyzer's status system is based on the general status register model shown in Figure 5-1. Most of the analyzer's register sets include all of the registers shown in the model, although commands are not always available for reading or writing a particular register. The information flow within a register set starts at the condition register and ends at the register summary bit (see Figure 5-2). This flow is controlled by setting bits in the transition and enable registers.

Two register sets — the Status Byte and the Standard Event Status Register — are 8-bits wide. All others are 16-bits wide, but the most significant bit (bit 15) in the larger registers is always set to 0.

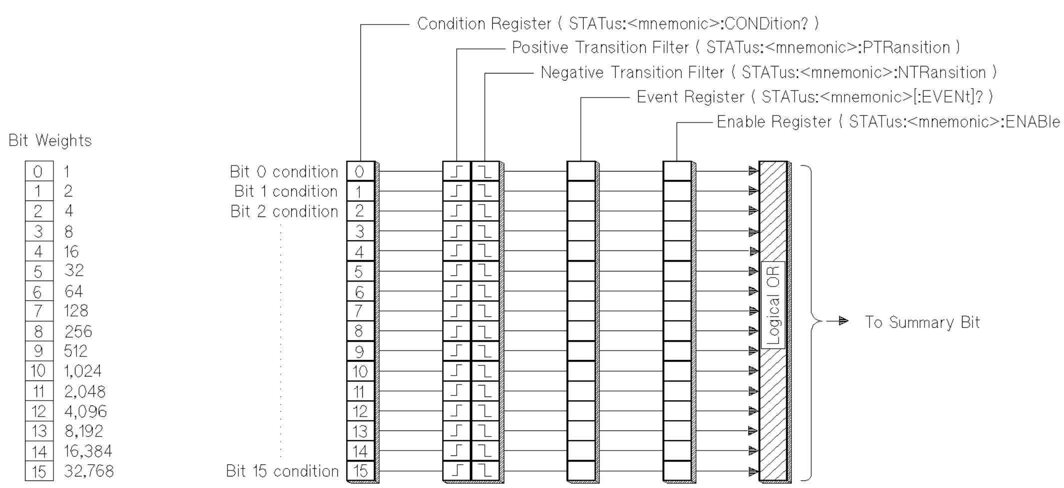


Figure 5-1. General Status Register Model

Condition Register

Condition registers continuously monitor the instrument's hardware and firmware status. Bits in a condition register are not latched or buffered, they are updated in real time. When the condition monitored by a specific bit becomes true, the bit is set to 1. When the condition becomes false the bit is reset to 0. Condition registers are read-only.

Transition Registers

Transition registers control what type of change in a condition register will set the corresponding bit in the event register. Positive state transitions (0 to 1) are only reported to the event register if the corresponding positive transition bit is set to 1. Negative state transitions (1 to 0) are only reported if the corresponding negative transition bit is set to 1. Setting both transition bits to 1 causes both positive and negative changes to be reported. Transition registers are read-write, and are unaffected by `*CLS` (clear status) or queries. They are reset to instrument default conditions at power up and after `*RST` and `SYSTem:PRESet` commands.

Event Register

Event registers latch any reported condition changes. When a transition bit allows a condition change to be reported, the corresponding event bit is set to 1. Once set, an event bit is no longer affected by condition changes. It remains set until the event register is cleared. Event registers are read-only.

An event register is cleared when you read it. All event registers are cleared when you send the `*CLS` (clear status) command.

Enable Register

Enable registers control the reporting of events (latched conditions) to the register summary bit. If an enable bit is set to 1 the corresponding event is included in the logical ORing process that determines the state of the summary bit. (The summary bit is only set to 1 if one or more enabled event bits are set to 1.) Summary bits are recorded in the instrument's status byte. Enable registers are read-write and are cleared by *CLS (clear status).

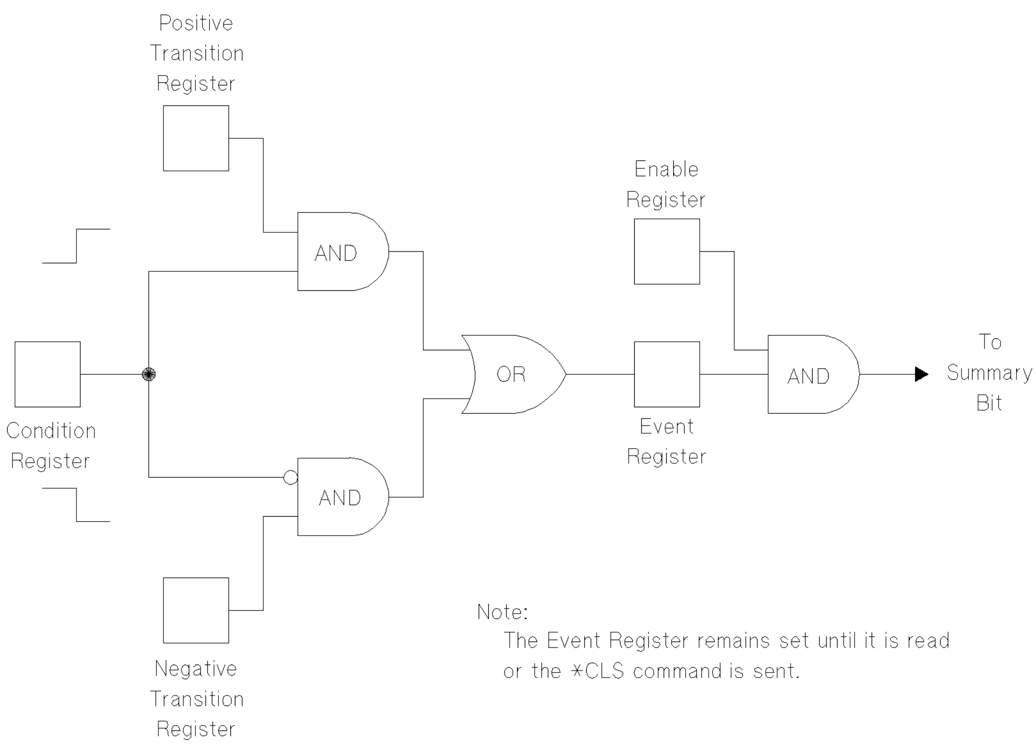


Figure 5-2. Flow of information within a register set

How to Use Registers

There are two methods of accessing the information in status registers:

- The direct-read method.
- The service request (SRQ) method.

In the direct-read method the analyzer is passive. It only tells the controller that conditions have changed when the controller asks the right question. In the SRQ method, the analyzer is more active. It tells the controller when there has been a condition change without the controller asking. Either method allows you to monitor one or more conditions.

The following steps are used to monitor a condition with the direct read method:

1. Determine which register contains the bit that monitors the condition.
2. Send the unique HP-IB query that reads that register.
3. Examine the bit to see if the condition has changed.

The direct-read method works well when it is not necessary to know about changes the moment they occur. It does not work well if immediate knowledge of the condition change is needed. A program that used this method to detect a change in a condition would need to continuously read the registers at very short intervals. The SRQ method is better suited for that type of need.

The Service Request Process

The following steps are used to monitor a condition with the SRQ method:

1. Determine which bit monitors the condition.
2. Determine how that bit reports to the request service (RQS) bit of the Status Byte.
3. Send HP-IB commands to enable the bit that monitors the condition and to enable the summary bits that report the condition to the RQS bit.
4. Enable the controller to respond to service requests.

When the condition changes, the analyzer sets its RQS bit and the HP-IB's SRQ line. The controller is informed of the change as soon as it occurs. The time the controller would otherwise have used to monitor the condition can now be used to perform other tasks. The controller's response to the SRQ is determined by the program being run.

Generating a Service Request

A service request is generated using the Status Byte. As shown in Figure 5-3, the analyzer's other register sets report to the Status Byte. Some of them report directly while others report indirectly through other register sets.

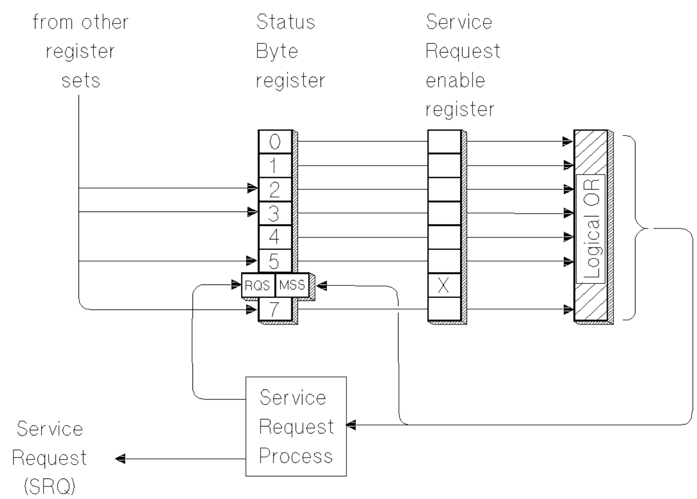


Figure 5-3. Generating a Service Request

The process of preparing the analyzer to generate a service request, and the handling of that interrupt when it is received by a program, are demonstrated in the `SRQ` example program.

When a register set causes its summary bit in the Status Byte to change from 0 to 1, the analyzer can initiate the service request (SRQ) process. If both the following conditions are true the process is initiated:

- The corresponding bit of the Service Request enable register is also set to 1.
- The analyzer does not have a service request pending. (A service request is considered to be pending between the time the analyzer's SRQ process is initiated and the time the controller reads the Status Byte register with a serial poll).

The SRQ process sets the HP-IB's SRQ line true and sets the Status Byte's request service (RQS) bit to 1. Both actions are necessary to inform the controller that the analyzer requires service. Setting the SRQ line informs the controller that some device on the bus requires service. Setting the RQS bit allows the controller to determine that the analyzer was the device that initiated the request.

When a program enables a controller to detect and respond to service requests, it should instruct the controller to perform a serial poll when the HP-IB's SRQ line is set true. Each device on the bus returns the contents of its Status Byte register in response to this poll. The device whose RQS bit is set to 1 is the device that requested service.

NOTE

When the analyzer's Status Byte is read with a serial poll, the RQS bit is reset to 0. Other bits in the register are not affected.

As implied in Figure 5-3, bit 6 of the Status Byte register serves two functions; the request service function (RQS) and the master summary status function (MSS). Two different methods for reading the register allow you to access the two functions. Reading the register with a serial poll allows you to access the bit's RQS function. Reading the register with `*STB` allows you to access the bit's MSS function.

The Analyzer's Status Register Sets

The analyzer uses eight register sets to keep track of instrument status:

Status Byte	`*STB?` and `*SRE`
Device Status	`STATus:DEVice`
Limit Fail	`STATus:QUEStionable:LIMit`
Questionable Status	`STATus:QUEStionable`
Standard Event Status	`*ESR?` and `*ESE`
Measuring Status	`STATus:OPERation:MEASuring`
Averaging Status	`STATus:OPERation:AVERaging`
Operational Status	`STATus:OPERation`

Their reporting structure is summarized in Figure 5-4. They are described in greater detail in the following section.

NOTE

Register bits not explicitly presented in the following sections are not used by the analyzer. A query to one of these bits returns a value of 0.

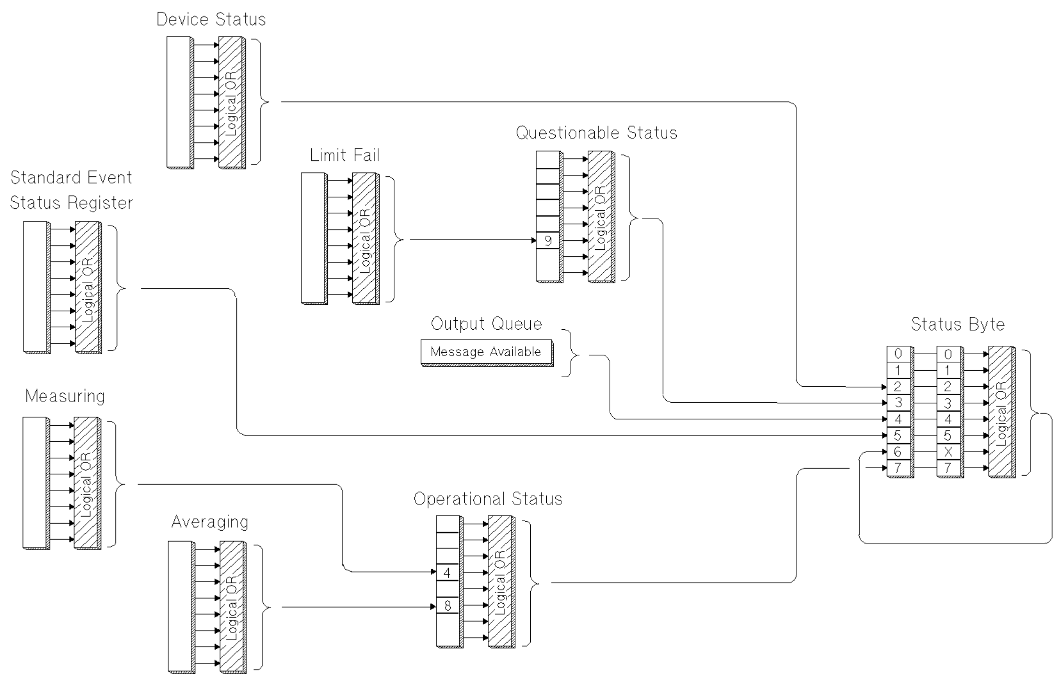


Figure 5-4. Analyzer Register Sets

Status Byte

The Status Byte register set summarizes the states of the other register sets and monitors the analyzer's output queue. It is also responsible for generating service requests (see "Generating a Service Request" earlier in this chapter). See Figure 5-5.

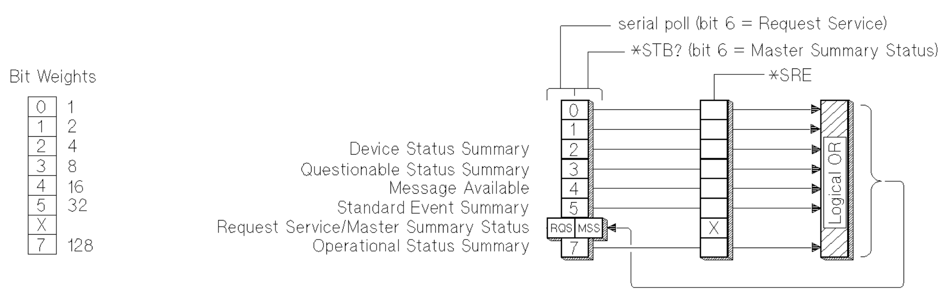


Figure 5-5. The Status Byte Register Set

The Status Byte register set does not conform to the general status register model described at the beginning of this chapter. It contains only two registers: the Status Byte register and the Service Request enable register. The Status Byte register behaves like a condition register for all bits except bit 6. The Service Request enable register behaves like a standard enable register except that bit 6 is always set to 0.

Bits in the Status Byte register are set to 1 under the following conditions:

Device Status Summary	(bit 2) is set to 1 when one or more enabled bits in the Device Status event register are set to 1.
Questionable Status Summary	(bit 3) is set to 1 when one or more enabled bits in the Questionable Status event register are set to 1.
Message Available	(bit 4) is set to 1 when the output queue contains a response message.
Standard Event Status Summary	(bit 5) is set to 1 when one or more enabled bits in the Standard Event Status event register are set to 1.
Master Summary Status	(bit 6, when read by `*STB`) is set to 1 when one or more enabled bits in the Status Byte register are set to 1.
Request Service	(bit 6, when read by serial poll) is set to 1 by the service request process (see "Generating a Service Request" earlier in this chapter).
Operational Status Summary	(bit 7) is set to 1 when one or more enabled bits in the Operational Status event register are set to 1.

The commands used to read and write the Status Byte registers are listed below:

`SPOLL` an IBASIC (or HP BASIC) command used in the service request process to determine which device on the bus is requesting service.

`*STB?` reads the value of the instrument's status byte. This is a non-destructive read, the Status Byte is cleared by the `*CLS` command.

`*SRE <num>` sets bits in the Service Request Enable register. The current setting of the Service Request Enable register is stored in non-volatile memory. If `*PSC` has been set, it will be saved at power on.

`*SRE?` reads the current state of the Service Request Enable register.

Device Status Register Set

The Device Status register set monitors the state of device-specific parameters.

Bits in the Device Status condition register are set to 1 under the following conditions:

Key Pressed	(bit 0) is set to 1 when one of the analyzer's front panel keys has been pressed.

Limit Fail Register Set

The Limit Fail register set monitors limit test results for both measurement channels.

Bits in the Limit Fail condition register are set to 1 under the following conditions:

Channel 1 Limit Failed	(bit 0) is set to 1 when limit testing in enabled and any point on channel 1 fails the limit test.
Channel 2 Limit Failed	(bit 1) is set to 1 when limit testing in enabled and any point on channel 2 fails the limit test.

Questionable Status Register Set

The Questionable Status register set monitors conditions that affect the quality of measurement data.

Bits in the Questionable Status condition register are set to 1 under the following conditions:

Limit Fail	(bit 9) is set to 1 when one or more enabled bits in the Limit Fail event register are set to 1.
Data Questionable	(bit 10) is set to 1 when a change in the analyzer's configuration requires that new measurement data be taken.

Standard Event Status Register Set

The Standard Event Status register set monitors HP-IB errors and synchronization conditions. See Figure 5-6

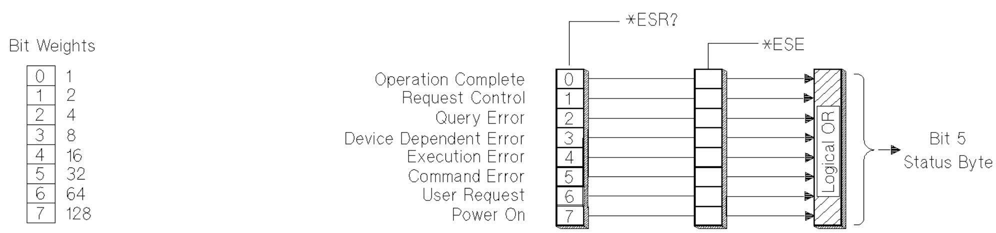


Figure 5-6. The Standard Event Status Register Set

The Standard Event Status register set does not conform to the general status register model described at the beginning of this section. It contains only two registers: the Standard Event Status event register and the Standard Event Status enable register. The Standard Event Status event register is similar to other event registers, but behaves like a register set that has a positive transition register with all bits set to 1. The Standard Event Status enable register is the same as other enable registers.

Operation Complete (bit 0) is set to one when the following two events occur (in the order listed):

- The `*OPC` command is sent to the analyzer.
- The analyzer completes all pending overlapped commands.

Request Control (bit 1) is set to 1 when both of the following conditions are true:

- The analyzer is configured as a talker/listener for HP-IB operation.
- The analyzer is instructed to do something (such as plotting or printing) that requires it to take control of the bus.

Query Error (bit 2) is set when the command parser detects a query error.

Device Dependent Error (bit 3) is set to 1 when the command parser detects a device-dependent error.

Execution Error (bit 4) is set to 1 when the command parser detects an execution error.

Command Error (bit 5) is set to 1 when the command parser detects a command error.

Power On (bit 7) is set to 1 when you turn on the analyzer.

The commands used to read and write the Standard Event Status registers are listed below:

`*ESR?` reads the value of the standard event status register.

`*ESE <num>` sets bits in the standard event status enable register. The current setting of the standard event statue enable register is stored in non-volatile memory. If `*PSC` has been set, it will be saved at power on.

`*ESE?` reads the current state of the standard event status enable register.

Measuring Status Register Set

The Measuring Status register set monitors conditions in the analyzer's measurement process.

Bits in the Measuring Status condition register are set to 1 under the following conditions:

Channel 1 Measuring (bit 0) is set to 1 while the analyzer is collecting measurement data on channel 1.

Channel 2 Measuring (bit 1) is set to 1 while the analyzer is collecting measurement data on channel 2.

Averaging Status Register Set

The Averaging Status register set monitors conditions in the analyzer's measurement process when the trace averaging function is in use.

Bits in the Averaging Status condition register are set to 1 under the following conditions:

Channel 1 Averaging (bit 0) is set to 1 while the analyzer is sweeping on channel 1 and the number of sweeps completed (since "average restart") is less than the averaging factor.

Channel 2 Averaging (bit 1) is set to 1 while the analyzer is sweeping on channel 2 and the number of sweeps completed (since "average restart") is less than the averaging factor.

Operational Status Register Set

The Operation Status register set monitors conditions in the analyzer's measurement process, disk operations, and printing/plotting operations. It also monitors the state of the current HP Instrument BASIC program.

Bits in the Operational Status condition register are set to 1 under the following conditions:

Calibrating	(bit 0) is set to 1 while the instrument is zeroing the broadband diode detectors.
Settling	(bit 1) is set to 1 while the measurement hardware is settling.
Measuring	(bit 4) is set to 1 when one or more enabled bits in the Measuring Status event register are set to 1.
Correcting	(bit 7) is set to 1 while the analyzer is performing a calibration function.
Averaging	(bit 8) is set to 1 when one or more enabled bits in the Averaging Status event register are set to 1.
Hardcopy Running	(bit 9) is set to 1 while the analyzer is performing a hardcopy (print or plot) function.
Test Running	(bit 10) is set to 1 when one of the analyzer's internal service tests is being run.
Program Running	(bit 14) is set to 1 while an HP Instrument BASIC program is running on the analyzer's internal controller.

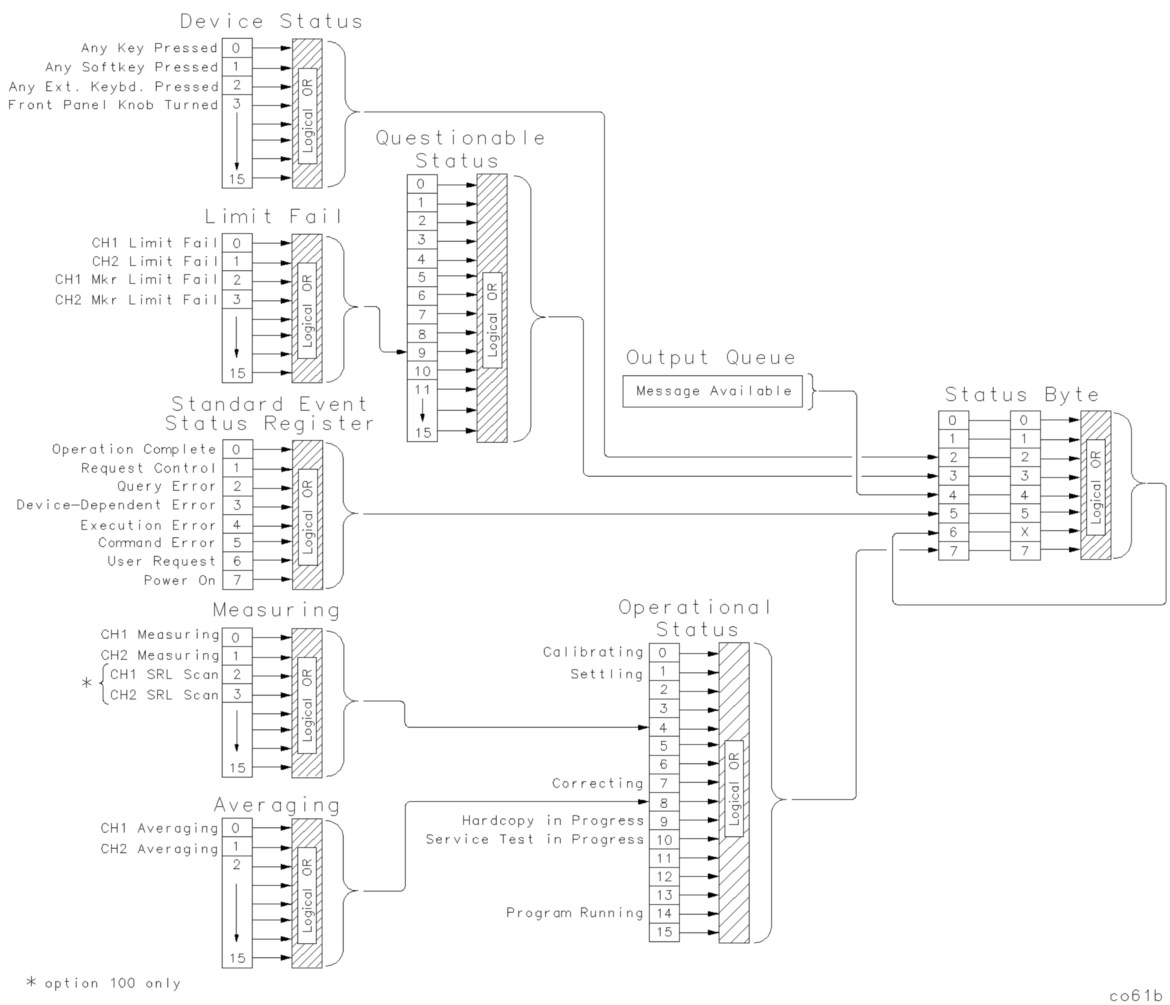
STATus:PRESet Settings

Executing the `STATus:PRESet` command changes the settings in the enable (`ENAB`), positive transition (`PTR`) and negative transition (`NTR`) registers. The table below shows the settings after the command is executed.

Register Set	`ENABle`	`PTRansition`	`NTRansition`
`STATus:DEVice`	all 0s	all 1s	all 0s
`STATus:QUEStionable:LIMit`	all 1s	all 1s	all 0s
`STATus:QUEStionable`	all 0s	all 1s	all 0s
`STATus:OPERation:MEASuring`	all 1s	all 0s	all 1s
`STATus:OPERation:AVERaging`	all 1s	all 0s	all 1s
`STATus:OPERation`	all 0s	all 1s	all 0s

,

Analyzer Register Set Summary



6

Trace Data Transfers

Trace Data Transfers

This chapter explains how to read (query) the measurement data trace from the analyzer into your program. It also describes how to send data from your program to the analyzer's measurement arrays. Accessing the measurement arrays is done using SCPI commands. If you are using IBASIC (Option 1C2), you can also access the measurement arrays using high-speed subroutines. Refer to the *HP Instrument BASIC User's Handbook* for more details.

Figure 6-1 is a data processing flow diagram that represents the flow of numerical data. The data passes through several math operations, denoted in the figure by single-line boxes. Most of these operations can be selected and controlled with the front panel CONFIGURE block menus. **The data is stored in arrays along the way, denoted by double-line boxes. These arrays are places in the flow path where data is accessible via HP-IB.** While only a single flow path is shown, two identical paths are available, corresponding to channel 1 and channel 2.

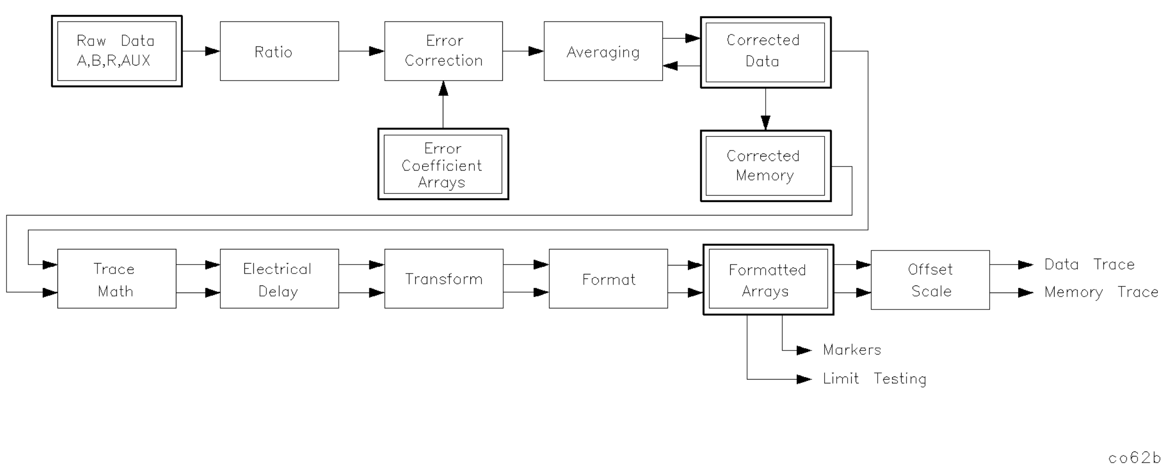


Figure 6-1. Numeric Data Flow Through the Network Analyzer

Querying the Measurement Trace Using BASIC

After making a measurement, you can read the resultant measurement trace out of the analyzer using the SCPI query

```
"TRACE:DATA?  CH1FDATA"
```

The BASIC program segment below shows how to read the trace from the analyzer into an array in your program.

```
10   REAL Trace(1:201)
20   ASSIGN @Hp8711 TO 716
30   ! Take sweep here
40   OUTPUT @Hp8711;"FORM:DATA ASCII,5"
50   OUTPUT @Hp8711;"TRACE:DATA? CH1FDATA"
60   ENTER @Hp8711;Trace(*)
70   DISP Trace(1),Trace(2),Trace(3),". . . ."
```

In this program, the `TRACE:DATA?` query returns all of the measurement points as a single block. The analyzer computes the value for each point using the measurement format selected by the [FORMAT] menu (CALC:FORM SCPI command), and returns a block of data called the formatted data array. The values of each point correspond to the values displayed on the screen, or those shown in the marker readouts. The frequency stimulus value (X-axis) of each point is not returned by the `TRACE:DATA?` query; only the measurement response (Y-axis) values are returned.

When transferring the block of trace data, you may select either binary or ASCII data encoding. This is explained in Chapter 4 in the section titled "Data Encoding for Large Data Transfers." Notice that the terms "encoding format" and "measurement format" are not the same. The encoding format determines how the numbers are represented as bytes, while the measurement format corresponds to the meaning of the value of the numbers.

The easiest way to transfer a measurement data trace is to use ASCII data encoding.

In the example above, the array Trace(1:201) contains 201 real (floating point) numbers. The SCPI command `"FORM:DATA ASCII,5"` specifies ASCII data encoding, with 5 significant digits. The command `"TRACE:DATA? CH1FDATA"` instructs the analyzer to send the measurement trace. The ENTER statement reads the measurement data sent by the analyzer into the Trace(1:201) array.

It is important to make sure that the Trace array declared in your program is the same size as the measurement trace on the analyzer, or an error will occur. The ENTER statement attempts to read data from the analyzer until it completely fills the Trace array, at which point it expects to receive a end-of-data terminator from the analyzer. To be safe, your program should use the `"SENS:SWE:POIN"` SCPI command to set the number of measurement data points to the desired value.

Refer to the example program ASCDATA in Chapter 8 for a complete example.

Smith Chart and Polar Formats

Each measurement point is represented by a single floating point number. This is the case for all of the analyzer's measurement formats except Smith Chart and Polar in the HP 8712B and 8714B. When Smith Chart or Polar format is selected, each point is represented by two numbers, the first one being the real portion and the second being the imaginary portion of the complex measurement value.

Below is a modified example program that will work when using Smith Chart or Polar formats.

```
10   REAL Trace(1:201,1:2)
20   ASSIGN @Hp8711 TO 716
30   ! Take sweep here
40   OUTPUT @Hp8711;"FORM:DATA ASCII,5"
50   OUTPUT @Hp8711;"TRACE:DATA? CH1FDATA"
60   ENTER @Hp8711;Trace(*)
70   DISP Trace(1,1),Trace(1,2),". . . .",Trace(201,1),Trace(201,2)
```

Querying the Measurement Trace Using SICL

This section includes a complete SICL C program that shows how to read the measurement trace from the analyzer.

```
/***********************************************************************
 * This program takes a sweep, reads the trace, and prints it.
 * It uses SICL (Standard Instrument Control Library) to talk
 * to the analyzer over HP-IB.
 *
 * On HP-UX, compile using:    cc -Aa -o query_trace  query_trace.c -lsicl
 ***********************************************************************/


#include <sicl.h>           /* For iopen(), iprintf(), iscanf(), INST, ... */
#include <stdio.h>          /* For printf() */

int main(void)
{
    INST analyzer;              /* Handle used to talk to analyzer */
    float data_buf[1601];       /* measurement trace.  32-bit floats */
    int num_trace_bytes;
    int pt;

    num_trace_bytes = sizeof(data_buf);   /* Set to max allowable bytes */

    /* Open the network analyzer at address 16 */
    analyzer = iopen("hpib,16");

    /* Clear the bus */
    iclear(analyzer);

    /* Abort current sweep and put analyzer sweep in hold */
    iprintf(analyzer, "ABORT\n");
    iprintf(analyzer, "INIT:CONT OFF\n");

    /* Take one sweep, wait until done */
    iprintf(analyzer, "INIT1\n");
    iprintf(analyzer, "*OPC?\n");
```

```
    iscanf(analyzer, "%*s");

    /* Request the trace data in 32-bit floating point format */
    iprintf(analyzer, "FORM:BORD NORM\n");
    iprintf(analyzer, "FORM:DATA REAL,32\n");

    /* Query the trace, read into data_buf[]. */
    iprintf(analyzer, "TRAC? CH1FDATA\n");
    iscanf(analyzer, "%#b%*c", &num_trace_bytes, &data_buf[0]);

    /* Print the trace values. */
    for (pt = 0; pt < num_trace_bytes/sizeof(float); pt++) {
        printf("%4d     %g\n", pt, data_buf[pt]);
    }

    /* Close analyzer and exit. */
    iclose(analyzer);
    return 0;
}
```

Using Binary Data Encoding

The previous section describes how to query the measurement trace, and transfer it into your program using ASCII encoding. Binary encoding can be used for faster data transfers, as shown in the table below:

Table 6-1. Typical Trace Transfer Times (ms)

Number of Points	Binary	ASCII
51	38	60
201	59	199
401	98	390
1601	335	1510

When using binary data transfers, the entire trace is sent from the analyzer to your program in a block called a definite length block. The details of block data are described in detail in Chapter 4. The definite length block contains a header and a data section. The header indicates how many bytes are in the data section.

In order to read the definite length block, your program must first read the header, and then read the data section. Refer to the example program REALDATA in Chapter 8 for an example of how to do this.

In the REALDATA program, you will notice the following lines which read the definite block header:

```
180   ENTER @Hp8711 USING "%,A,D";A$,Digits
190   ENTER @Hp8711 USING "%,"&VAL$(Digits)&"D";Bytes
```

and these lines which read the data section:

```
200   ASSIGN @Hp8711;FORMAT OFF
210   ENTER @Hp8711;Data1(*)
```

Each measurement point in the data section is represented as 4 or 8 bytes (32 or 64 bits), depending on whether single precision or double precision numbers are requested. When using HP BASIC or IBASIC, you must select double precision numbers to match BASIC's "REAL" data type. Do this using the SCPI command `"FORM:DATA REAL,64"`. If you are using another language that supports single precision data types, you can select single precision using the SCPI command `"FORM:DATA REAL,32"`. Languages such as QuickBASIC and C have support for both single and double precision floating point numbers.

When transferring data using binary encoding, you may need to reverse the order of the bytes in each measurement point, since PCs frequently store IEEE floating point numbers with the byte order reversed. To instruct the analyzer to reverse the byte order of the data, send the command `"FORMAT:BORDer SWAPped"` before querying the trace data.

Trace Data Transfer Sizes

The following table shows how many bytes are transmitted during trace data transfers. The left column shows the format of the data, which you can specify using the SCPI command `Format:DATA`. As you can see, the size of the measurement point data and trace data varies as you change format.

Table 6-2. Size of Trace Data Transfers (in Bytes) Using the TRACE:DATA SCPI Command

Format Type (`FORMat:DATA`)	Type of Data	Single Measurement Point		201 Point Trace	
		Real	Complex	Real	Complex
REAL,32	IEEE 32-bit Floating Point	4	8	809	1614
REAL,64	IEEE 64-bit Floating Point	8	16	1614	3222
ASCII,5	ASCII numbers	13	26	2613	5226
ASCII,3	ASCII numbers	11	22	2211	4422
INT,16	Internal Binary	—	6	—	1212

When transmitting data in "REAL" or "INT" format, a header is sent before the data block. The header indicates the size of the data block. The header size varies in length from 3 to 11 bytes. Refer to Chapter 4 for details on the header.

Transmitting ASCII data requires no header. The ASCII values are separated by commas, and a linefeed is sent after the last value. The sizes shown in the table include the size of the comma(s) and terminating linefeed. Typical data in ASCII,5 format:

```
-1.2254E+000,+5.0035E-001,+4.5226E-001,...
```

The analyzer stores its internal data with approximately 5 significant digits of resolution. Using REAL,32 or ASCII,5 format provides sufficient precision for data transfers. However, REAL,64 may be necessary when using a programming language which does not support IEEE 32-bit floating point.

Transferring Data with IBASIC

If you are using IBASIC, your IBASIC program can avoid the overhead of using OUTPUT and ENTER to transfer trace data, and instead use the analyzer's built-in high-speed subprograms. These built-in subroutines let you quickly move data between the analyzer's measurement arrays and your program's data arrays. For example, to read the analyzer's formatted data array, use the following:

```
10 DIM Fmt(1:201)
20 INTEGER Chan
30 LOADSUB Read_fdata FROM "XFER:MEM 0,0"
40 Chan=1
50 Read_fdata(Chan,Fmt(*))
```

Refer to the *HP Instrument BASIC User's Handbook* for more details.

The table below compares the speed of IBASIC using high-speed transfer subroutines with that of a fast external controller using the SCPI `TRACE:DATA? CH1FDATA` query.

Table 6-3. Typical Trace Transfer Times (ms)

Number of Points	Controller Using Binary `TRACE:DATA?`	IBASIC Using `Read_fdata`
51	38	14
201	59	37
401	98	67
1601	335	251

Taking Sweeps

When making measurements and querying traces, your program should perform the following steps:

1. Place the analyzer's sweep in hold
2. Initiate a single sweep
3. Wait for the sweep to complete
4. Query the measurement trace

Use the following program lines perform these steps:

```
10 OUTPUT @Hp8711;"ABORT;:INIT1:CONT OFF"
20 OUTPUT @Hp8711;"INIT1"
30 OUTPUT @Hp8711;"*OPC?"
35 ENTER @Hp8711;Opc
40 OUTPUT @Hp8711;"TRACE:DATA? CH1FDATA"
45 ENTER @Hp8711;Fmt(*)
```

If you query the measurement trace while the analyzer is in continuous sweep, the query will still work, but the data may not be correct. Using INIT and *OPC? ensures that a complete sweep has finished before you query the measurement data. In many cases, you can also use the command "*WAI" in place of the "*OPC?" query, replacing lines 30 and 35 above with:

```
30 OUTPUT @Hp8711;"*WAI"
```

However, there are cases where `"*WAI"` will produce incorrect results. One case is when using IBASIC's high-speed subprograms to query the trace data. `"*WAI"` only ensures that the SCPI commands following the `"*WAI"` are not executed until the commands before the `"*WAI"` are complete. Since IBASIC subprograms don't use SCPI commands to access the trace data, `"*WAI"` is ineffective, and `"*OPC?"` should be used.

When using `"*OPC?"`, the ENTER statement following the `"*OPC?"` will wait until the previous SCPI commands are complete, preventing your program from executing beyond the ENTER statement. When using `"*WAI"`, your program can continue to run and send SCPI commands, and the analyzer will buffer them and act upon them in order.

For more details, refer to Chapter 2, "Synchronizing the Analyzer and a Controller."

CALC:DATA? versus TRACE:DATA?

The SCPI command "`CALC1:DATA?`" is functionally equivalent to the command "`TRACE:DATA? CH1FDATA`". The two can be used interchangeably for trace queries of the formatted measurement data. The "`TRACE:DATA`" command is more flexible, allowing you to query other measurement arrays and to download data to measurement arrays.

Querying Single Data Points Using Markers

If you only need to query a single data point, you can use a marker query instead of a trace query. The program segment below shows how to do this using the SCPI command CALC:MARK.

```
10   ASSIGN @Hp8711 TO 716
20   ! Take sweep here
30   OUTPUT @Hp8711;"CALC1:MARK ON"   ! turn on marker
40   OUTPUT @Hp8711;"CALC1:MARK1:X 177 MHz"   ! set frequency
50   OUTPUT @Hp8711;"CALC1:MARK1:Y?"          ! read marker
60   ENTER @Hp8711;Marker_y
70   DISP Marker_y
```

You can also use the `CALC:MARK:FUNC:RES?` query to return the results of a bandwidth search. For example:

```
10   ! Select -3 dB bandwidth
20   OUTPUT @Hp8711;"CALC:MARK:BWID -3"
30   ! Get result of bandwidth search
40   OUTPUT @Hp8711;"CALC:MARK:FUNC:RES?"
50   ENTER @Hp8711;Bwidth,Center_freq,Q,Loss
```

For more information on using markers, refer to Chapter 8, "Example Programs."

Accessing Other Measurement Arrays

The preceding sections describe how to query the formatted data array using the **TRACE:DATA?** query with the argument CH1FDATA. The formatted array is the last array in the analyzer's data processing chain, and is generally of most interest.

The analyzer also allows you to query other measurement arrays which are earlier in its data processing chain. Figure 6-2, below, shows the data processing chain.

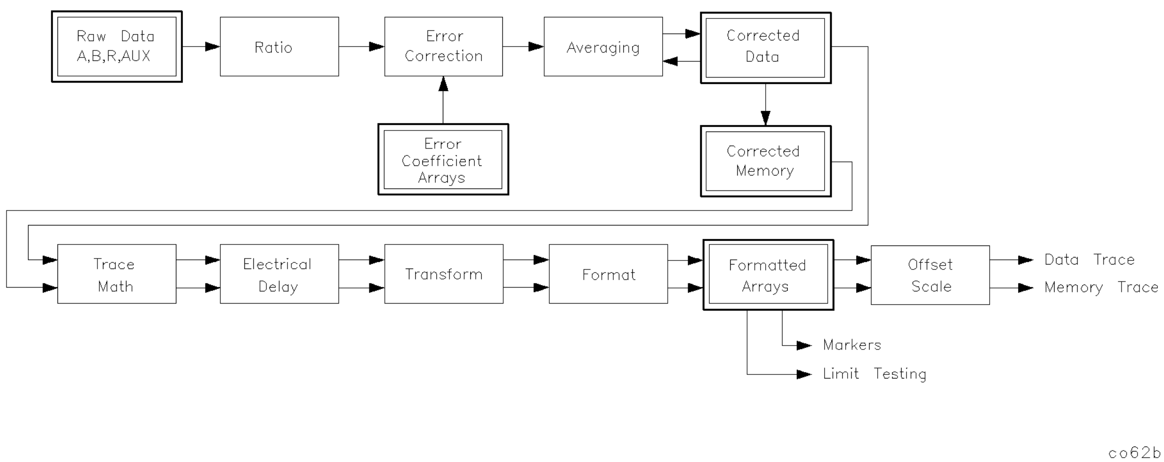


Figure 6-2. Numeric Data Flow Through the Network Analyzer

The first array is the Raw Data Array, which contains each of the separate input components (A, B, R, X, Y) immediately after they are measured. These arrays can be queried and set, but doing so is of limited use, since the data values contained in the arrays are uncorrected, and are not directly correlated to any meaningful reference, such as 0 dBm.

The Error Coefficient Arrays contain default correction values or values created during a measurement calibration. These arrays can be queried and set, but care should be exercised in setting them since incorrect measurements may result. If you wish to apply your own corrections in addition to the analyzer's current correction, the best technique is to use the Corrected Memory array and the Data/Memory feature, explained below.

The Corrected Data array contains the results of the currently selected measurement (Transmission, Reflection, etc.) after error correction and averaging have been applied. The measurement data in these arrays is represented as complex number pairs. When measuring the transmission response of a through cable, the magnitude of the complex numbers will be very close to 1.0. When measuring an open circuit, the magnitude of the complex numbers will be very close to 0.0. When measuring an amplifier, the magnitude of the complex numbers will be greater than 1.0.

The Corrected Memory array is filled with a copy of the Corrected Data array when the Data −> Memory operation is performed. It can be used to apply a gain correction to the measured data. This is described in the following section.

The Formatted Data array contains the measurement data after it has been formatted using the format selected by the [FORMAT] menu. Querying the Formatted Data array is described in detail at the beginning of this chapter. You can also download data to this array, and the analyzer will display the data using the current Scale and Reference values.

Applying Gain Correction Using the Memory Trace

The Corrected Memory array is filled with a copy of the Corrected Data array when the Data –> Memory operation is performed. By setting the analyzer to perform Data/Memory trace math, you can apply your own correction factor to the measurement data trace by filling the Corrected Memory array with the appropriate complex numbers.

In general, you should use the analyzer's calibration feature to correct for errors in your system. However, there may be cases where you wish to simulate the effect of adding a cable in series with your DUT, and observe how this imaginary cable will attenuate the measured response versus frequency. Or you may wish to apply an absolute offset to simulate the effect of adding or removing a pad from the measurement. These simulations are easily accomplished using the Corrected Memory array and the Data/Memory feature.

The Corrected Data and Memory arrays contain complex linear data, as opposed to logged data. When displaying the traces using Lin Mag format, the result of the Data divided by Memory operation (Data/Mem) will be to divide each point of the data trace by each point of the memory trace. When displaying data in Log Mag format, the result of Data/Memory will be equivalent to subtracting the Log Mag value of the Memory trace from that of the Data trace.

The following example BASIC code segment shows how to download a complex array from your program to the analyzer's Memory trace. The program's "Mem" array is initialized with the proper values such that when the analyzer computes Data divided by Memory, the desired increasing gain will be applied.

```
100   REAL Mem(1:201,1:2)
110   ASSIGN @Hp8711 TO 716
120   ! Fill memory array (denominator in Data/Mem)
130   ! with values that will result in an
140   ! upward sloping gain factor vs. frequency.
150   ! Used to compensate for cable loss vs. frequency
160   ! Adds 0 dB of gain at start freq; 3 dB at stop freq
170   FOR Pt=1 TO 201
180      Gain_factor_db=3.0*(Pt - 1)/200  ! 0..3 dB Power
190      Gain_factor_lin=10^(Gain_factor_db/20)
200      Mem(Pt,1)=1.0/Gain_factor_lin  ! real
210      Mem(Pt,2)=0.0                  ! imag
220   NEXT Pt
230   ! Download to the memory trace
240   OUTPUT @Hp8711;"FORM:DATA ASCII"
250   OUTPUT @Hp8711;"TRACE:DATA CH1SMEM";   ! Note the ";"
260   FOR Pt=1 TO 201
270      FOR I=1 TO 2
280         OUTPUT @Hp8711;",";Mem(Pt,I);    ! Note the ";"
290      NEXT I
300   NEXT Pt
310   OUTPUT @Hp8711;""    ! Send linefeed
320   OUTPUT @Hp8711;"CALC1:MATH (IMPL/CH1SMEM)"  ! Data/Mem
```

The example above downloads data to the corrected memory array. The data is sent by the program to the analyzer using ASCII encoding. The data is sent as ASCII characters, separated by commas. The analyzer accepts the comma separated list of numbers until it receives a linefeed to terminate the command. The program uses semicolons at the end of some OUTPUT statements to avoid sending a linefeed until all of the data has been sent. After the last number is sent, the program sends a linefeed, and the analyzer accepts the data.

Remember, for faster transfers, use binary data encoding instead of ASCII.

Performing Your Own Data Processing

After the analyzer has made a measurement, you can read the measurement trace and perform your own post-processing on it, and display the result on the screen. This is done using these steps:

1. Initiate a sweep
2. Wait for the sweep to finish
3. Read the measurement data into an array in your program
4. Perform your post-processing on the measurement data
5. Write (download) the post-processed data to the analyzer's memory trace.

You may want to instruct the analyzer to display only the memory trace and not the data trace, so that only your post-processed data is seen.

The program below demonstrates how to perform data post-processing. It takes the measurement data and reverses it, such that the low frequency data is displayed on the right end of the trace, and the high frequency data is displayed on the left.

```
100   ! Display the measurement data backwards
110   REAL Fmt(1:201)
120   ASSIGN @Hp8711 TO 716
130   !
140   OUTPUT @Hp8711;"FORM:DATA ASCII"
150   OUTPUT @Hp8711;"ABOR;INIT:CONT OFF;*WAI"
160   OUTPUT @Hp8711;"DISP:WIND:TRAC1 OFF;TRAC2 ON"
170   LOOP
180      ! Take sweep
190        OUTPUT @Hp8711;"INIT1;*WAI"
200        ! Read the trace from the formatted data array
210        OUTPUT @Hp8711;"TRACE:DATA? CH1FDATA"
220        ENTER @Hp8711;Fmt(*)
230        ! Download the trace, backwards,
235      ! to the formatted memory array
240        OUTPUT @Hp8711;"TRACE:DATA CH1FMEM";   ! Note the ";"
250        FOR Pt=1 TO 201
260            OUTPUT @Hp8711;",";Fmt(202-Pt);    ! Note the ";"
270        NEXT Pt
280        OUTPUT @Hp8711;""    ! Send linefeed
290   END LOOP
```

This example program uses ASCII trace data transfers. Higher speed can be achieved using binary data transfers. If using IBASIC, high-speed subroutines can be used for even greater performance. Refer to the IBASIC Handbook for details.

Downloading Trace Data Using Binary Encoding

Data traces can be downloaded to the analyzer using binary encoding. Using binary encoding is faster than using ASCII encoding. As mentioned in Chapter 4, the binary encoded trace is transferred as a block; the block containing a header and a data section. There are two different types of blocks that can be used: a definite length block, and an indefinite length block.

To send trace data using a definite length block, your program must calculate the number of bytes in the data segment of the block, and create a block header which tells the analyzer how many bytes are in the data segment.

For example, if you are sending a trace with 201 data points and using 64-bit floating point numbers for each data point ("FORM:DATA REAL,64"), the block's data segment will contain 1608 bytes (201 points * 8 bytes/point). The header characters for a 1608 byte block are: "#41608". The first digit after the "#", "4" tells how many following digits are used to specify the size. In this case, 4 digits follow, and those digits are "1608", meaning that the block contains 1608 bytes.

When you send a definite length block to the analyzer, the analyzer will will read the data segment bytes, stopping when it receives the number specified in the block header.

To send trace data using an indefinite length block, your program simply sends a block header of "#0", followed by the data segment. After sending the data segment, your program must terminate the data block by sending an EOI. The analyzer will read the data segment bytes, stopping when it receives an EOI. To send an EOI using BASIC, you can use the statement:

```
OUTPUT @Hp8711;END
```

Internal Measurement Arrays

The following sections describe the sequence of math operations and the resulting data arrays as the measurement information flows from the raw data arrays to the display. This information explains the measurement arrays accessible via HP-IB.

Figure 6-3 is a data processing flow diagram that represents the flow of numerical data. The data passes through several math operations, denoted in the figure by single-line boxes. Most of these operations can be selected and controlled with the front panel CONFIGURE block menus. **The data is stored in arrays along the way, denoted by double-line boxes. These arrays are places in the flow path where data is accessible via HP-IB.** While only a single flow path is shown, two identical paths are available, corresponding to channel 1 and channel 2.

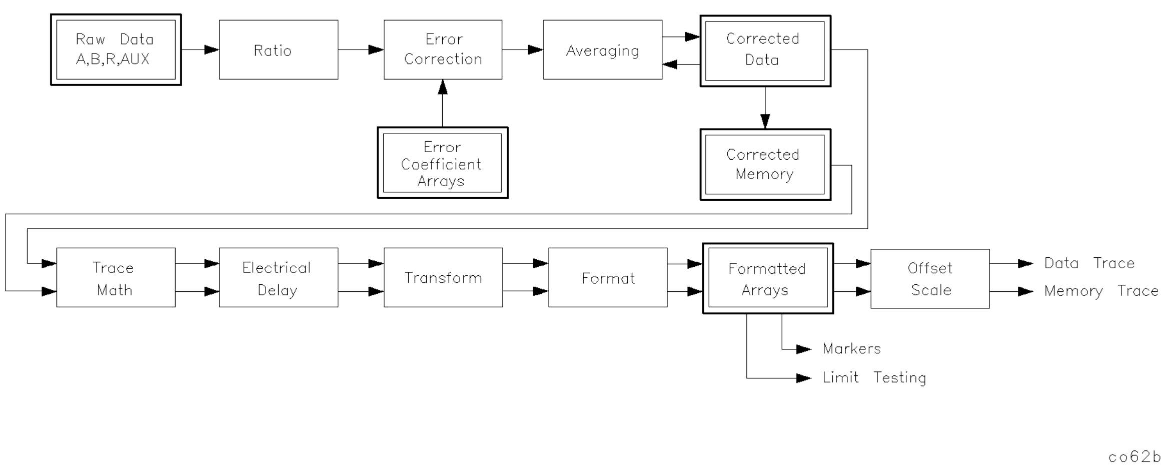


Figure 6-3. Numeric Data Flow Through the Network Analyzer

Raw Data Arrays

These arrays are linear measurements of the inputs used in the selected measurement. Note that these numbers are complex pairs. These arrays are directly accessible via HP-IB and referenced as CH[1|2]AFWD, CH[1|2]BFWD and CH[1|2]RFWD.

Table 6-4. Raw Data Arrays

Selected Measurement	Raw Arrays
Transmission (B/R)	B = CH[1|2]BFWD, R = CH[1|2]RFWD
Reflection (A/R)	A = CH[1|2]AFWD, R = CH[1|2]RFWD
A	A = CH[1|2]AFWD
B	B = CH[1|2]BFWD
R	R = CH[1|2]RFWD
Power (B*)	B* = CH[1|2]BFWD
Conversion Loss (B*/R*)	B* = CH[1|2]BFWD, R* = CH[1|2]RFWD
R*	R* = CH[1|2]RFWD
AM Delay (Y/X)	Y = CH[1|2]BFWD, X = CH[1|2]RFWD
X	X = CH[1|2]RFWD
Y	Y = CH[1|2]BFWD
Y/R*	Y = CH[1|2]BFWD, R* = CH[1|2]RFWD
Y/X, X/Y	Y = CH[1|2]BFWD, X = CH[1|2]RFWD

NOTE

Raw data for AUX INPUT is not available via HP-IB. Use the corrected data array to access AUX INPUT data.

Ratio Calculations

These are performed if the selected measurement is a ratio (e.g. A/R or B/R). This is simply a complex divide operation. If the selected measurement is absolute (e.g. A or B), no operation is performed.

Error Correction

Error correction is performed next if correction is turned on. Error correction removes repeatable systematic errors (stored in the error coefficient arrays) from the raw arrays. The operations performed depend on the selected measurement type.

Error Coefficient Arrays

The error coefficient arrays are either default values or are created during a measurement calibration. These are used whenever correction is on. They contain complex number pairs, and are accessible via HP-IB and are referenced as `CH[1|2]SCORR1`, `CH[1|2]SCORR2` and `CH[1|2]SCORR3`.

Table 6-5. Error Coefficient Arrays

Selected Measurement	Error Coefficient Arrays
Transmission (B/R)	`CH[1|2]SCORR1` = Response `CH[1|2]SCORR2` = Isolation
Reflection (A/R)	`CH[1|2]SCORR1` = Directivity `CH[1|2]SCORR2` = Source Match `CH[1|2]SCORR3` = Tracking
Broadband Internal	`CH[1|2]SCORR1` = R* Response

NOTE

These arrays do not apply to Broadband External measurements.

Averaging

Averaging is a noise reduction technique. This calculation involves taking the complex exponential average of several consecutive sweeps. This averaging calculation is different than the System Bandwidth setting. System Bandwidth uses digital filtering, applying noise reduction to the measured data before it is stored into the Raw Data Arrays.

Corrected Data Arrays

The combined results of the ratio, error correction and averaging operations are stored in the corrected data arrays as complex number pairs. These arrays are accessible via HP-IB and referenced as `CH[1|2]SDATA`.

Corrected Memory Arrays

If the Data−>Mem or Normalize operations are performed, the corrected data arrays are copied into the corrected memory arrays. These arrays are accessible via HP-IB and referenced as `CH[1|2]SMEM`.

Trace Math Operation

This selects either the corrected data array, or the corrected memory array, or both to continue flowing through the data processing path. In addition, the complex ratio of the two (Data/Memory) can also be selected. If memory is displayed, the data from the memory arrays goes through exactly the same data processing flow path as the data from the data arrays.

Electrical Delay

This block adds or subtracts phase, based on the settings of Phase Offset, Electrical Delay, and Port Extension. The Electrical Delay and Port Extension features add or subtract phase in proportion to frequency. This is equivalent to "line stretching" or artificially moving the measurement reference plane. (See the *HP 8712B/14B User's Guide* for more details on these features.)

Transform (Option 100 only)

This block converts frequency domain data into distance domain, or into an SRL impedance value when measuring fault location or SRL. The transform employs an inverse fast Fourier transform (FFT) algorithm to accomplish the conversion.

Formatting

This converts the complex number pairs into a scalar representation for display, according to the selected format (e.g. Log Mag, SWR, etc). These formats are often easier to interpret than the complex number representation. Note that after formatting, it is impossible to recover the complex data.

Formatted Arrays

The results so far are stored in the formatted data and formatted memory arrays. It is important to note that marker values and marker functions are all derived from the formatted arrays. Limit testing is also performed on the formatted arrays. These arrays are accessible via HP-IB and referenced as `CH[1|2]FDATA` and `CH[1|2]FMEM`.

Offset and Scale

These operations prepare the formatted arrays for display. This is where the reference position, reference value, and scale calculations are performed, as appropriate for the format.

7

Using Graphics

Using Graphics

The analyzer has a set of user graphics commands that can be used to create graphics and messages on the display. The `GRAPHICS` example program uses some of these commands to draw a simple setup diagram. These commands, listed below, are of the form:

`DISPlay:WINDow[1|2|10]:GRAPhics:<mnemonic>.`

The number specified in the `WINDow` part of the command selects where the graphics are to be written.

`WINDow1` draws the graphics to the channel 1 measurement screen. (This is the default if no window is specified in the mnemonic.)

`WINDow2` draws the graphics to the channel 2 measurement screen.

`WINDow10` draws the graphics to an IBASIC display partition. (This window is only available on instruments with IBASIC — Option 1C2.)

NOTE

When graphics commands are used to write directly to a measurement screen they write to the static graphics plane (the same plane where the graticule is drawn). There is no sweep-to-sweep speed penalty once the graphics have been drawn.

Unless otherwise specified, the graphics commands listed below start at the current pen location. All sizes are dimensioned in pixels.

DISPlay:WINDow[1|2|10]:GRAPhics:CIRCle <y_radius>

DISPlay:WINDow[1|2|10]:GRAPhics:CLEar

DISPlay:WINDow[1|2|10]:GRAPhics:COLor <pen>

- color choices are: 0 for erase, 1 for bright, 2 for dim

DISPlay:WINDow[1|2|10]:GRAPhics[:DRAW] <new_x>,<new_y>

DISPlay:WINDow[1|2|10]:GRAPhics:LABel <string>

DISPlay:WINDow[1|2|10]:GRAPhics:LABel:FONT <font>

- font choices are: SMALl, HSMall, NORMal, HNORmal, BOLD, HBOLd, SLANt, HSLant
  (H as the first letter of the font name indicates highlighted text – inverse video).

DISPlay:WINDow[1|2|10]:GRAPhics:MOVE <new_x>,<new_y>

DISPlay:WINDow[1|2|10]:GRAPhics:RECTangle <width>,<height>

DISPlay:WINDow[1|2|10]:GRAPhics:STATe?

NOTE

There are more screen pixels in the "X" direction than in the "Y" direction. The CIRCle graphics command compensates for this by drawing an ellipse using a larger X-radius than Y-radius.

Window Geometry

Even though there are only three graphics windows, these windows can have different sizes and locations.

The size and location of the graphics window are determined by the display configuration currently in use — split screen measurements, full screen measurements, and full or partial IBASIC display partitions will affect the dimensions of the graphics window in use.

The sizes of the different graphics windows are listed below.

- Channel 1 or channel 2 full screen measurement: width = 801 pixels, height = 321 pixels.
- Channel 1 or channel 2 split screen measurement: width = 801 pixels, height = 161 pixels.
- IBASIC full screen display: width = 861 pixels, height = 352 pixels.
- IBASIC upper display: width = 861 pixels, height = 160 pixels.
- IBASIC lower display: width = 861 pixels, height = 158 pixels.

There is a set of queries that can be used to determine the size and location of the display window in use.

These queries, listed below, return the width and height of the window or the absolute location of its lower left or upper right corners. All the coordinates and sizes are dimensioned in pixels.

```
DISPlay:WINDow[1|2|10]:GEOMetry:LLEFt?
DISPlay:WINDow[1|2|10]:GEOMetry:SIZE?
DISPlay:WINDow[1|2|10]:GEOMetry:URIGht?
```

NOTE

The origin of EVERY graphics window is its lower left corner. The locations returned in response to the `LLEFt` and `URIGht` are relative to the ABSOLUTE origin of the entire display, NOT to the graphics window.

The Graphics Buffer

The analyzer has a graphics buffer that is used to refresh the graphics display if needed. When the buffer is full, additional graphics can still be drawn — BUT they will not be refreshed. The graphics buffer can be turned on and off using the following command (which is used in the GRAPHICS example program).

```
DISPlay:WINDow:GRAPhics:BUFFer[:STATe] <ON|OFF>
```

The graphics buffer will hold up to:

500 lines
40 circles
40 rectangles
50 strings (60 characters long)

Use the following command to clear the graphics buffer and user-graphics display.

```
DISPlay:WINDow:GRAPhics:CLEar
```

NOTE

Only graphics that can be refreshed will be printed or plotted. If you intend to print or plot your graphics, make sure they will fit within the graphics buffer.

8

Example Programs

Example Programs

The example programs listed in this manual are all written in IBASIC (HP Instrument BASIC). An optional internal controller can be purchased with your analyzer (option 1C2). This controller runs IBASIC directly on the analyzer. It controls the analyzer over an internal interface bus that operates the same way as the external HP-IB interface. For more information about IBASIC refer to the *HP Instrument BASIC User's Handbook*.

IBASIC is a programming language that developed from HP BASIC. Because of this relationship, programs written for IBASIC can be run on external controllers that run HP BASIC.

The example programs are provided on two disks that are included with the network analyzer. Both disks contain the same examples written in IBASIC: only the disk format is different. Because the analyzer's internal 3.5" disk drive is designed to be both DOS and LIF compatible, either disk can be used to supply programs for the analyzer's internal IBASIC controller.

Example Programs Disk – DOS Format HP part number 08712-10001

Example Programs Disk – LIF Format HP part number 08712-10002

Because the examples are designed to run in different environments, the setup at the beginning of each program must determine the operating environment and properly set the analyzer's HP-IB address. In these examples, the internal IBASIC controller uses the address 800 when communicating with the analyzer (the internal HP-IB is at select code 8). The default address of 716 is used when the programs are being run on an external controller.

A version of the following lines is included in all of the example programs. The use of the `Internal` (internal-controller) flag varies due to differences in the programs needs.

`10 IF POS(SYSTEM$("SYSTEM ID"),"HP 871") THEN`	*Identify the operating system.*
`20 ASSIGN @Hp8711 TO 800`	*If internal, set address to 800.*
`30 Internal=1`	*Set internal-control flag to 1.*
`40 ELSE`	*If external, set address to 716.*
`50 ASSIGN @Hp8711 TO 716`	
`60 Internal=0`	*Set internal-control flag to 0.*
`70 ABORT 7`	*Abort all bus transactions and give active control of the bus to the computer.*
`80 CLEAR 716`	*Send a selected device clear (SDC) to the analyzer – this clears all HP-IB errors, resets the HP-IB interface and clears syntax errors. (It does not affect the status reporting system.)*
`90 END IF`	

The following table shows the sections and example programs that are contained in this chapter:

Section Title	Example Program	Program Description
Configuring Measurements	SETUP	Sets up a basic measurement, demonstrates use of `*WAI`
	LIMITEST	Performs automatic pass/fail testing with limit lines
Transfer of Data to/from the Analyzer	MARKERS	Transfers data using markers
	SMITHMKR[1]	Measures reflection of a filter in Smith chart and polar formats
	ASCDATA	Transfers data using ASCII format
	REALDATA	Transfers data using the IEEE 64-bit floating point REAL format
	INTDATA	Transfers data using the 16-bit INTEGER format
Calibration	TRANCAL	Performs a transmission calibration
	REFLCAL	Performs a reflection calibration
	LOADCALS	Uploads and downloads correction arrays
	CALKIT	This is *not* a program, it is an instrument state file for downloading user-defined cal kit definitions
Instrument State and Save/Recall	LEARNSTR	Uses the learn string to upload and download instrument states
	SAVERCL	Saves and recalls instrument states, calibrations and data
Hardcopy Control	PRINTPLT	Uses the serial and parallel ports for hardcopy output
	PASSCTRL	Uses pass control and the HP-IB for hardcopy output
	FAST_PRT	Provides fast graph dumps to PCL5 printers
Service Request	SRQ	Generates a service request interrupt
File Transfer Over HP-IB	GETFILE	Transfers a file from the analyzer to an external controller
	PUTFILE	Transfers a file from an external controller to the analyzer
Customized Display	GRAPHICS	Uses graphics and softkeys to create customized procedures

1 For use with HP 8712B and 8714B only

Configuring Measurements

SETUP	Setting up a basic measurement. The example also demonstrates the use of the *WAI command.
LIMITEST	Performing automatic PASS/FAIL testing with limit lines. The example also demonstrates some methods of combining mnemonics for more efficient programming.

SETUP Example Program

This program demonstrates how to set up the analyzer to make a basic measurement. The `*WAI` command is used extensively throughout this program. This has the effect of making sure that the commands are executed in the order they are received. More information about making measurements with the analyzer is available in the *User's Guide*.

Lines `10-70` are explained in the introduction to the example programs section. They determine which system controller is being used, set flags, and prepare the instrument for remote operation.

```
1     !Filename:  SETUP
2     !
3     ! Description:
4     !   Set Channel 1 to measure filter's transmission.
4     !   Set Channel 2 to measure filter's reflection
5     !   Prompt user for start and stop freq, and set them.
6     !   Take a sweep.
7     !   Set Scale and Reference levels.
8     !
10    IF POS(SYSTEM$("SYSTEM ID"),"HP 871") THEN
20      ASSIGN @Hp8711 TO 800
30    ELSE
40      ASSIGN @Hp8711 TO 716
50      ABORT 7
60      CLEAR 716
70    END IF
71    !
73    ! Preset the instrument.
80    OUTPUT @Hp8711;"SYST:PRES;*WAI"
81    !
83    ! Configure the analyzer to measure transmission
84    ! of a filter on channel 1.  This is the command
85    ! for the BEGIN Filter Transmissn key sequence.
90    OUTPUT @Hp8711;"CONF 'FILT:TRAN';*WAI"
91    !
93    ! Put the instrument in trigger hold mode.
100   OUTPUT @Hp8711;"ABOR;:INIT:CONT OFF;*WAI"
```

```
101   !
103   ! Turn on channel 2.
110   OUTPUT @Hp8711;"SENS2:STAT ON;*WAI"
111   !
113   ! Configure channel 2 to measure reflection.  This
114   ! is the command for the CHAN 2 Reflection key sequence.
120   OUTPUT @Hp8711;"SENS2:FUNC 'XFR:POW:RAT 1,0';DET NBAN"
121   !
122   ! Wait for the previous commands to complete execution
123   ! (respond to the *OPC?).
124   OUTPUT @Hp8711;"*OPC?"
125   ENTER @Hp8711;Opc
126   !
127   ! Input a start frequency.
130   INPUT "Enter Start Frequency (MHz):",Start_f
131   !
132   ! Input a stop frequency.
140   INPUT "Enter Stop Frequency (MHz):",Stop_f
141   !
142   ! Set the start and stop frequencies of the analyzer
143   ! to the values entered.
150   OUTPUT @Hp8711;"SENS2:FREQ:STAR";Start_f;"MHz;STOP"
      ;Stop_f;"MHz;*WAI"
151   !
152   ! Trigger a single sweep.
160   OUTPUT @Hp8711;"INIT;*OPC?"
161   !
162   ! Wait for the sweep to be completed.
170   ENTER @Hp8711;Opc
171   !
172   ! Set up the scale and reference parameters for channel 1.
180   OUTPUT @Hp8711;"DISP:WIND1:TRAC:Y:PDIV 10 DB;RLEV 0 DB
      ;RPOS 8"
181   !
182   ! Now for channel 2.
190   OUTPUT @Hp8711;"DISP:WIND2:TRAC:Y:PDIV 5 DB;RLEV 0 DB;RPOS 8"
191   !
192   ! Make channel 1  active (transmission)
200   OUTPUT @Hp8711;"SENS1:STAT ON"
201   !
202   ! Display the current start and stop frequencies.
```

```
210   DISP "Done measuring.  Start =";Start_f;"MHz
        Stop =";Stop_f;"MHz"
220   END
```

LIMITEST Example Program

This program demonstrates how to set up and use limit lines over the HP-IB. The example device used in this program is the demonstration filter that is shipped with the analyzer. The program sets up the basic measurement, downloads the limit lines and uses the status registers to determine of the device passes its specifications. For more information about limit lines, refer to the *User's Guide*. For information about using the status registers, refer to the previous section "Using the Status Registers."

This example also demonstrates how multiple command mnemonics can be combined together. The easiest commands to combine are ones that are closely related on the command tree (such as the start and stop frequency of a limit segment). For more information of command mnemonics, refer to Chapter 10, "Introduction to SCPI."

Lines 20-80 are explained in the introduction to the example programs section. They determine which system controller is being used, set flags, and prepare the instrument for remote operation.

```
1      !Filename:  LIMITEST
2      !
10     DIM Title$[30]
20     IF POS(SYSTEM$("SYSTEM ID"),"HP 871") THEN
30       ASSIGN @Hp8711 TO 800
40     ELSE
50       ASSIGN @Hp8711 TO 716
60       ABORT 7
70       CLEAR 716
80     END IF
81     !
82     ! Perform a system preset; this clears the limit table.
90     OUTPUT @Hp8711;"SYST:PRES;*WAI"
91     !
92     ! Set up the source frequencies for the measurement.
100    OUTPUT @Hp8711;"SENS1:FREQ:STAR 10 MHZ;STOP 400 MHZ;*WAI"
101    !
102    ! Set up the receiver for the measurement parameters
103    ! (Transmission in this case).
110    OUTPUT @Hp8711;"SENS1:FUNC 'XFR:POW:RAT 2,0';DET NBAN;*WAI"
```

```
111  !
112  ! Configure the display so measurement
113  ! results are easy to see.
120  OUTPUT @Hp8711;"DISP:WIND1:TRAC:Y:PDIV 10 DB;
     RLEV 0 DB;RPOS 9"
121  !
122  ! Reduce the distractions on the display by
123  ! getting rid of notation that will not be
124  ! needed in this example.
130  OUTPUT @Hp8711;"DISP:ANN:YAX OFF"
131  !
132  ! Erase the graticule grid for the same reason.
140  OUTPUT @Hp8711;"DISP:WIND1:TRAC:GRAT:GRID OFF"
141  !
142  ! Create and turn on the first segment for
143  ! the new limit lines; this one is a maximum
144  ! limit.
150  OUTPUT @Hp8711;"CALC1:LIM:SEGM1:TYPE LMAX;STAT ON"
151  !
152  ! Set the amplitude limits for the first limit
153  ! segment.
160  OUTPUT @Hp8711;"CALC1:LIM:SEGM1:AMPL:STAR -70;STOP -70"
161  !
162  ! Set the frequency of the first limit segment.
170  OUTPUT @Hp8711;"CALC1:LIM:SEGM1:FREQ:STAR 10 MHZ;STOP 75 MHZ"
171  !
172  ! Create and turn on a second maximum limit
173  ! segment.
180  OUTPUT @Hp8711;"CALC1:LIM:SEGM2:TYPE LMAX;STAT ON"
181  !
182  ! Set the amplitude limits for segment 2.
190  OUTPUT @Hp8711;"CALC1:LIM:SEGM2:AMPL:STAR 0;STOP 0"
191  !
192  ! Set the frequency range for segment 2.
200  OUTPUT @Hp8711;"CALC1:LIM:SEGM2:FREQ:STAR 145 MHZ
     ;STOP 200 MHZ"
201  !
202  ! Create and turn on a third limit segment;
203  ! this one is a minimum limit.
210  OUTPUT @Hp8711;"CALC1:LIM:SEGM3:TYPE LMIN;STAT ON"
211  !
```

```
212   ! Set the amplitude limits for segment 3.
220   OUTPUT @Hp8711;"CALC1:LIM:SEGM3:AMPL:STAR -6;STOP -6"
221   !
222   ! Set the frequency range for segment 3.
230   OUTPUT @Hp8711;"CALC1:LIM:SEGM3:FREQ:STAR 150 MHZ
      ;STOP 195 MHZ"
231   !
232   ! Create and set parameters for segment 4.
240   OUTPUT @Hp8711;"CALC1:LIM:SEGM4:TYPE LMAX;STAT ON"
250   OUTPUT @Hp8711;"CALC1:LIM:SEGM4:AMPL:STAR -60;STOP -60"
260   OUTPUT @Hp8711;"CALC1:LIM:SEGM4:FREQ:STAR 290 MHZ
      ;STOP 400 MHZ"
261   !
262   ! Send an operation complete query to ensure that
263   ! all overlapped commands have been executed.
270   OUTPUT @Hp8711;"*OPC?"
271   !
272   ! Wait for the reply.
280   ENTER @Hp8711;Opc
281   !
282   ! Turn on the display of the limit lines.
290   OUTPUT @Hp8711;"CALC1:LIM:DISP ON"
291   !
292   ! Turn on the pass/fail testing; watch the
293   ! analyzer's display for the pass/fail indicator.
300   OUTPUT @Hp8711;"CALC1:LIM:STAT ON"
301   !
302   ! Take a controlled sweep to ensure that
303   ! there is real data present for the limit test.
310   OUTPUT @Hp8711;"ABOR;:INIT1:CONT OFF;:INIT1;*WAI"
311   !
312   ! Query the limit fail condition register to see
313   ! if there is a failure.
320   OUTPUT @Hp8711;"STAT:QUES:LIM:COND?"
321   !
322   ! Read the register's contents.
330   ENTER @Hp8711;Fail_flag
331   !
332   ! Bit 0 is the test result for channel 1 while
333   ! bit 1 is the results for channel 2 limit testing.
340   IF BIT(Fail_flag,0)=1 THEN
```

```
341     !
342     ! In case of failure, give additional direction
343     ! to the operator using the title strings.
350     Title$="Limit Test FAIL - Tune device"
351     !
352     ! Turn on the title string.
360     OUTPUT @Hp8711;"DISP:ANN:TITL1:DATA '"&Title$&"';STAT ON"
361     !
362     ! Turn on continuous sweep mode for tuning.
370     OUTPUT @Hp8711;"INIT1:CONT ON;*WAI"
371     !
372     ! Loop while the tuning is taking place.
380     LOOP
381       !
383       ! Monitor the status of the limit fail
384       ! condition register.
390       OUTPUT @Hp8711;"STAT:QUES:LIM:COND?"
400       ENTER @Hp8711;Fail_flag
401     !
402     ! Check the limit fail bit.  Exit if the
403     ! device has been tuned to pass the test.
410     EXIT IF BIT(Fail_flag,0)=0
420     END LOOP
430   END IF
431   !
432   ! Turn off the prompt to the operator and
433   ! return the analyzer to the continuously
434   ! sweeping mode.
440   OUTPUT @Hp8711;"DISP:ANN:TITL1 OFF;:INIT:CONT ON;*WAI"
450   END
```

Transfer of Data to/from the Analyzer

MARKERS	Transferring data using markers. The example also demonstrates the use of the query form of command mnemonics.
⊛ SMITHMKR	Measures reflection of a filter in Smith chart and polar formats.
ASCDATA	Transferring data using the ASCII format.
REALDATA	Transferring data using the IEEE 64-bit floating point REAL format. The example also demonstrates block data transfers of both indefinite length and definite length syntax. Also demonstrated is access to the swapped-byte data format designed for PCs.
INTDATA	Transferring data using the 16-bit INTEGER format.

MARKERS Example Program

This program demonstrates how to transfer measurement data by using the markers. Before any data is read over the HP-IB a controlled sweep should be taken. The analyzer has the ability to process and execute commands very quickly when they are received over the HP-IB. This speed can lead to commands (such as marker searches) being executed before any data has been taken. To ensure that the sweep has completed and the data is present before it is read, the command for a single sweep is used before data is requested. Note that `*WAI` is sent with that command. More information about making measurements with the analyzer is available in the *User's Guide*.

Lines `10-70` are explained in the introduction to the example programs section. They determine which system controller is being used, set flags, and prepare the instrument for remote operation.

```
1     !Filename:  MARKERS
2     !
3     ! Description:
4     !   1. Take sweep
5     !   2. Set marker to 175 MHz, and query Y value
6     !   3. Execute Marker -> Max, and query X and Y
7     !   4. Turn on marker tracking
8     !   5. Execute a 3 dB bandwidth search
9     !   6. Query the result
10    IF POS(SYSTEM$("SYSTEM ID"),"HP 871") THEN
20      ASSIGN @Hp8711 TO 800
30    ELSE
40      ASSIGN @Hp8711 TO 716
50      ABORT 7
60      CLEAR 716
70    END IF
71    !
72    ! Turn on channel 1 and set up start and stop
73    ! frequencies for the example.  These frequencies
74    ! were chosen for the demonstration filter that is
75    ! shipped with the analyzer.
80    OUTPUT @Hp8711;"SENS1:STAT ON;FREQ:STAR 10 MHZ;STOP 400
```

```
MHZ;*WAI"
81    !
82    ! Configure a transmission measurement on channel 1
83    ! using the narrowband detection mode.
90    OUTPUT @Hp8711;"SENS1:FUNC 'XFR:POW:RAT 2,0';DET NBAN;*WAI"
91    !
92    ! Take a single controlled sweep and have the
93    ! analyzer wait until it has completed before
94    ! executing the next command.
100   OUTPUT @Hp8711;"ABOR;:INIT:CONT OFF;:INIT;*WAI"
101   !
102   ! Turn on the first marker.
110   OUTPUT @Hp8711;"CALC1:MARK1 ON"
111   !
112   ! Set marker 1 to a frequency of 175 MHz.
120   OUTPUT @Hp8711;"CALC1:MARK1:X 175 MHZ"
121   !
122   ! Query the amplitude of the signal at 175 MHz.
130   OUTPUT @Hp8711;"CALC1:MARK1:Y?"
131   !
132   ! Read the data; the data is in the NR3 format.
140   ENTER @Hp8711;Data_1
150   DISP "Marker 1 (175 MHz) = ";Data_1
160   WAIT 5
161   !
162   ! Turn on the second marker and use a marker
163   ! search function to find the maximum point
164   ! on the data trace.
170   OUTPUT @Hp8711;"CALC1:MARK2 ON;MARK2:MAX"
171   !
172   ! Query the frequency and amplitude of the
173   ! maximum point.  Note that the two queries can
174   ! be combined into one command.
180   OUTPUT @Hp8711;"CALC1:MARK2:X?;Y?"
181   !
182   ! Read the data.
190   ENTER @Hp8711;Freq2,Data2
191   !
192   ! Display the results of the marker search.
200   DISP "Max = ";Data2;"dB at";Freq2/1.E+6;"MHz"
201   !
```

```
202   ! Put the analyzer into its continuously
203   ! sweeping mode.  This mode works well for
204   ! tuning applications.
210   OUTPUT @Hp8711;"INIT:CONT ON;*WAI"
211   !
212   ! Turn on the marker search tracking function.
213   ! This function causes the marker 2 to track
214   ! the maximum value each time the analyzer takes
215   ! a sweep.
220   OUTPUT @Hp8711;"CALC1:MARK2:FUNC:TRAC ON"
230   WAIT 5
231   !
232   ! Turn off marker 2.
240   OUTPUT @Hp8711;"CALC1:MARK2 OFF"
241   !
242   ! Take a single controlled sweep.
250   OUTPUT @Hp8711;"ABOR;:INIT:CONT OFF;:INIT;*WAI"
251   !
252   ! Perform a search for the -3 dB bandwidth of
253   ! the filter.  This function uses several
254   ! markers to find four key values.
260   OUTPUT @Hp8711;"CALC1:MARK:BWID -3;FUNC:RES?"
261   !
262   ! Read the four values:  the bandwidth, center
263   ! frequency, Q and the insertion loss.
270   ENTER @Hp8711;Bwid,Center_f,Q,Loss
271   !
272   ! Display the results.
280   DISP "BW: ";Bwid
290   WAIT 5
300   DISP "Center Freq: ";Center_f
310   WAIT 5
320   DISP "Q: ";Q
330   WAIT 5
340   DISP "Loss: ";Loss
341   !
342   ! Turn off all the markers.
350   OUTPUT @Hp8711;"CALC1:MARK:AOFF"
360   END
```

SMITHMKR Example Program

```
10    !Filename:  SMITHCHART
20    !
70    !
80    ! Description:  Measures a 175 MHz BPF using the
90    !   Smith and Polar plot formats.  User must connect
100   !   the 175 MHz filter between the reflection and
          transmission
110   !   ports.  The program will do the following:
120   !     1) Set analyzer to sweep over the filter's
                passband (50 MHz)
130   !     2) Set analyzer to Smith Chart format; make a marker
140   !         reading (Frequency, Real Impedance in ohms,
                 Imaginary Impedance
150   !         in ohms, Impedance Capacitance or Inductance);
                 dump the
160   !         trace and print S11 Real and Imaginary values
                 for the
170   !         first data point.
180   !     3) Set analyzer to Polar Chart format; make
                a marker
190   !        reading (Frequency, Linear Magnitude in "units",
200   !        Phase in degrees); dump the
210   !        trace and print S11 Real and Imaginary values
                for the
220   !        first data point.
230   !
240   !**************************************************
250   ! DEFINITIONS
260   !
270   REAL Opc,Freq_center,Freq_span,Freq_start,Bpf_q,Bpf_loss
280   REAL Mrkr_freq,Mrkr_res,Mrkr_reac,Mrkr_ind
290   REAL Trace_s11(1:201,1:2),Mrkr_mag,Mrkr_phas
300   !
310   !***************************************************
320   ! Determine computer type
```

```
330   !
340   CLEAR SCREEN
350   !
360   IF POS(SYSTEM$("SYSTEM ID"),"HP 871") THEN
      !if this is an 871x
370     ASSIGN @Hp871x TO 800
        !use 871x internal address
380   ELSE
      !program running on ext computer
390     ASSIGN @Hp871x TO 716
        !use 871x external address
400     ABORT 7
        !abort operations on HP-IB
410     CLEAR 716
420   END IF
430   !
440   !-----------------------------------------------
450   ! Preset analyzer, set Center and Span frequencies
460   !
470   OUTPUT @Hp871x;"SYST:PRES;*OPC?"
      !preset instrument
480   ENTER @Hp871x;Opc
      !waits for PRESET to finish before proceeding
490   !
500   ! Center the filter's frequency response (to get an
        accurate Bandwidth measurement)
510   !
520   DISP "Setting analyzer frequencies..."
      !message to user
530   OUTPUT @Hp871x;"ABOR;:INIT:CONT OFF;:INIT;*OPC?"
      !take a single sweep
540   ENTER @Hp871x;Opc
      !wait for sweep to finish
550   OUTPUT @Hp871x;"CALC1:MARK:FUNC MAX;*WAI"
      !set Marker 1 to max
560   OUTPUT @Hp871x;"CALC1:MARK:X?;*WAI"
      !get Marker frequency setting
570   ENTER @Hp871x;Mrkr_freq
      !read frequency of max marker
580   OUTPUT @Hp871x;"SENS1:FREQ:CENT "&VAL$(Mrkr_freq)
```

```
     &" HZ;*WAI"  !set Center Freq
590  OUTPUT @Hp871x;"SENS1:FREQ:SPAN 200 MHZ;*WAI"
     !set Span Freq = 200 MHz
600  !
610  ! Measure Bandwidth, set Center to band center,
       Span to 50 MHz
620  !
630  OUTPUT @Hp871x;"ABOR;:INIT:CONT OFF;:INIT;*OPC?"
     !take a single sweep
640  ENTER @Hp871x;Opc
     !wait for sweep to finish
650  OUTPUT @Hp871x;"CALC1:MARK:FUNC BWID;*OPC?"
     !search filter for -3dB bandwidth
660  ENTER @Hp871x;Opc
     !wait for bandwidth to be found
670  OUTPUT @Hp871x;"CALC1:MARK:FUNC:RES?"
     !read the bandwidth data
680  ENTER @Hp871x;Freq_span,Freq_center,Bpf_q,Bpf_loss

     !read in data
690  OUTPUT @Hp871x;"SENS1:FREQ:CENT "&VAL$(Freq_center)&"
     HZ;*WAI"  !set Center Freq
700  OUTPUT @Hp871x;"SENS1:FREQ:SPAN 50 MHZ;*WAI"
     !set Span Freq to 50 MHz (passband)
710  !
720  !--------------------------------------------------
730  ! Set marker 1 to beginning of trace
740  !
750  OUTPUT @Hp871x;"CALC1:MARK:AOFF;*WAI"
     !clear all markers
760  OUTPUT @Hp871x;"CALC1:MARK1 ON"
     !turn on marker 1
770  OUTPUT @Hp871x;"SENS1:FREQ:STAR?"
     !get start frequency
780  ENTER @Hp871x;Freq_start
     !enter start freq
790  OUTPUT @Hp871x;"CALC1:MARK1:X "&VAL$(Freq_start)
     &";*OPC?"      !set marker to start freq
800  ENTER @Hp871x;Opc
     !wait for all previous commands to finish
810  !
```

```
820   !-----------------------------------------------
830   ! Set to Reflection mode & Smith Chart format.
840   !
850   DISP "Setting to Smith Chart format..."
860   OUTPUT @Hp871x;"ABOR;:INIT1:CONT ON;*WAI"
      !set to Cont Sweep mode so can select reflection
870   OUTPUT @Hp871x;"SENS1:FUNC 'XFR:POW:RAT 1,0'
      ;DET NBAN;*WAI"       !CHAN1=reflection
880   OUTPUT @Hp871x;"CALC1:FORM SMIT;*WAI"
      !set smith chart format
890   !
900   !------------------------------------------------
910   ! Read marker information from Smith Chart
920   !
930   OUTPUT @Hp871x;"ABOR;:INIT:CONT OFF;:INIT;*OPC?"
      !force one sweep before read markers
940   ENTER @Hp871x;Opc
      !wait for sweep to finish
950   OUTPUT @Hp871x;"ABOR;:INIT1:CONT ON;*WAI"
      !set to Continuous Sweep mode
960   OUTPUT @Hp871x;"CALC1:MARK:X?"
      !read marker frequency
970   ENTER @Hp871x;Mrkr_freq
      !units are in Hz
980   OUTPUT @Hp871x;"CALC1:MARK:Y:RES?"
      !read real part of marker impedance
990   ENTER @Hp871x;Mrkr_res
      !units are in ohms
1000  OUTPUT @Hp871x;"CALC1:MARK:Y:REAC?"
      !read imaginary part of marker impedance
1010  ENTER @Hp871x;Mrkr_reac
      !units are in ohms
1020  OUTPUT @Hp871x;"CALC1:MARK:Y:IND?"
      !read inductance (or capacitance)
1030  ENTER @Hp871x;Mrkr_ind
      !units are Henries if positive value, Farads if negative
1040  !
1050  !----------------------------------------------------------
1060  ! Display Smith Marker data
1070  !
1080  Mrkr_freq=DROUND(Mrkr_freq,3)
```

```
      !round frequency to 3 digits
1090  DISP "Smith Marker Frequency = "&VAL$(Mrkr_freq)
      &"Hz"      !display frequency
1100  WAIT 3
1110  !
1120  Mrkr_res=DROUND(Mrkr_res,3)
      !round resistance to 3 digits
1130  DISP "Smith Marker Resistance = "&VAL$(Mrkr_res)
      &" ohms"
1140  WAIT 3
1150  !
1160  Mrkr_reac=DROUND(Mrkr_reac,3)
      !round reactance to 3 digits
1170  DISP "Smith Marker Reactance = "&VAL$(Mrkr_reac)
      &" ohms"
1180  WAIT 3
1190  !
1200  Mrkr_ind=DROUND(Mrkr_ind,3)
      !round inductance to 3 digits
1210  IF Mrkr_ind<0 THEN
      !label as capacitance if negative
1220    DISP "Smith Marker Capacitance = "&VAL$(-Mrkr_ind)
        &"F"        !label capacitance
1230  ELSE
      !label as inductance if positive
1240    DISP "Smith Marker Inductance = "&VAL$(Mrkr_ind)
        &"H"       !label inductance
1250  END IF
1260  WAIT 3
1270  !
1280  !----------------------------------------------------
1290  ! Read Smith Chart formatted trace data, display
        first data point.
1300  !   Data is transferred in ASCII format with 3
          significant digits.
1310  !   S11 trace data is read out as:  Real data for
          point #1, Imaginary data
1320  !   for point #1, Real data for point #2, Imaginary
          data for point #2...
1330  !   Since instrument was preset, number of trace data
          points
```

```
1340  !   defaults to 201.
1350  !
1360  OUTPUT @Hp871x;"FORM:DATA ASC,3;:TRAC? CH1FDATA"
      !set up to read ASCII data, 3 digits
1370  ENTER @Hp871x;Trace_s11(*)
      !read trace data, real & imaginary pairs
1380  !
1390  ! Display data
1400  !
1410  DISP "Smith Trace Point #1:  S11(REAL) =
      "&VAL$(Trace_s11(1,1))&" Units"  !display Real data
1420  WAIT 3
1430  DISP "Smith Trace Point #1:  S11(IMAGINARY) =
      "&VAL$(Trace_s11(1,2))&" Units"
      !display Imaginary data
1440  WAIT 3
1450  !
1460  !++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
1470  ! Set to Polar Chart Format, read Polar Markers
1480  !
1490  DISP "Setting to Polar Format..."
1500  OUTPUT @Hp871x;"CALC1:FORM POL;*WAI"
      !set polar chart format
1510  OUTPUT @Hp871x;"CALC1:MARK:X?"
      !read marker frequency
1520  ENTER @Hp871x;Mrkr_freq
      !units are in Hz
1530  OUTPUT @Hp871x;"CALC1:MARK:Y:MAGN?"
      !read magnitude marker reflection coefficient
1540  ENTER @Hp871x;Mrkr_mag
      !magnitude in "units"
1550  OUTPUT @Hp871x;"CALC1:MARK:Y:PHAS?"
      !read phase of marker reflection coefficient
1560  ENTER @Hp871x;Mrkr_phas
      !units are in degrees
1570  !
1580  !------------------------------------------------------------
1590  ! Display Polar Marker data
1600  !
1610  Mrkr_freq=DROUND(Mrkr_freq,3)
      !round frequency to 3 digits
```

```
1620  DISP "Polar Marker Frequency = "&VAL$(Mrkr_freq)&"Hz"
      !display frequency
1630  WAIT 3
1640  !
1650  Mrkr_mag=DROUND(Mrkr_mag,3)
      !round magnitude to 3 digits
1660  DISP "Polar Marker Magnitude = "&VAL$(Mrkr_mag)
      &" Units"    !display magnitude
1670  WAIT 3
1680  !
1690  Mrkr_phas=DROUND(Mrkr_phas,3)
      !round phase to 3 digits
1700  DISP "Polar Marker Phase = "&VAL$(Mrkr_phas)
      &" Degrees"        !display phase
1710  WAIT 3
1720  !
1730  !-----------------------------------------------------
1740  ! Read Polar Chart trace data, display first data point.
1750  !   S11 trace data is read out as:  Real data for
          point #1, Imaginary data
1760  !   for point #1, Real data for point #2, Imaginary data
          for point #2...
1770  !
1780  OUTPUT @Hp871x;"FORM:DATA ASC,3;:TRAC? CH1FDATA"
      !set up to read ASCII data, 3 digits
1790  ENTER @Hp871x;Trace_s11(*)
      !read trace data, real & imaginary pairs
1800  !
1810  ! Display data
1820  !
1830  DISP "Polar Trace Point #1:  S11(REAL) =
      "&VAL$(Trace_s11(1,1))&" Units"  !display Real data
1840  WAIT 3
1850  DISP "Polar Trace Point #1:  S11(IMAGINARY) =
      "&VAL$(Trace_s11(1,2))&" Units"  !display Imaginary data
1860  WAIT 3
1870  DISP ""          !clear display line
1880  !
1890  STOP
1900  END
```

ASCDATA Example Program

This program demonstrates how to read data arrays from the analyzer and write them back again. The `ASCii` data format is being used with a resolution of 5 digits. More information about data transfer is available in Chapter 4, "Data Types and Encoding," and Chapter 6, "Trace Data Transfers."

In addition to the channel 1 formatted data array used in this example, there are a number of arrays that can be accessed inside the instrument. These arrays and their corresponding mnemonics are listed in Chapter 6 in Table 6-4 and Table 6-5.

```
1     !Filename:  ASCDATA
2     !
3     ! Description:
4     !  1. Takes a sweep, and reads the formatted
5     !     data trace into an array.  The trace
6     !     is read as a definite length block.
7     !  2. Instructs you to remove DUT
8     !  3. Downloads the trace back to the analyzer
9     !     as an indefinite length block.
10    REAL Data1(1:51)
20    IF POS(SYSTEM$("SYSTEM ID"),"HP 871") THEN
30      ASSIGN @Hp8711 TO 800
40    ELSE
50      ASSIGN @Hp8711 TO 716
60      ABORT 7
70      CLEAR 716
80    END IF
81    !
82    ! Set the analyzer to measure 51 data points.
90    OUTPUT @Hp8711;"SENS1:SWE:POIN 51;*WAI"
91    !
92    ! Take a single sweep, leaving the analyzer
93    ! in trigger hold mode.
100   OUTPUT @Hp8711;"ABOR;:INIT1:CONT OFF;:INIT1;*WAI"
101   !
102   ! Set up the ASCII data format with 5
```

```
103   ! significant digits
110   OUTPUT @Hp8711;"FORM:DATA ASC,5"
111   !
112   ! request the channel 1 formatted data array
113   ! from the instrument.
115   OUTPUT @Hp8711;"TRAC? CH1FDATA"
116   !
117   ! Get the data and put into data array Data1.
120   ENTER @Hp8711;Data1(*)
121   !
122   ! Display the first 3 numbers in the array.
130   DISP "Trace: ";Data1(1);Data1(2);Data1(3);"..."
131   !
132   ! Use the wait time to visually compare the
133   ! numbers to the visible data trace.
140   WAIT 5
141   !
142   ! Prompt the operator to disconnect the test
143   ! device and then how to continue the program.
150   DISP "Disconnect the test device -- Press Continue"
160   PAUSE
161   !
162   ! Update the display line.
165   DISP "Taking a new sweep...";
166   !
167   ! Take a sweep so the display shows new data.
170   OUTPUT @Hp8711;":INIT1;*WAI"
175   DISP " Done."
180   WAIT 5
181   !
182   ! Prepare the analyzer to receive the data.
183   ! Suppress the "end" character by using a
184   ! semicolon at end of output statement.
185   DISP "Downloading saved trace...";
190   OUTPUT @Hp8711;"TRAC CH1FDATA";
191   !
192   ! Send the data array one point at a time,
193   ! using the semicolon at the end of the
194   ! output statement to suppress the
195   ! end character.
200   FOR I=1 TO 51
```

```
210     OUTPUT @Hp8711;", ";Data1(I);
220   NEXT I
221   !
222   ! Now send the end character.
230   OUTPUT @Hp8711;""
240   DISP " Done!"
250   END
```

REALDATA Example Program

This program demonstrates how to read data arrays from the analyzer and write them back again. The `REAL,64` data format is being used. Note that the analyzer outputs the data using the definite length block syntax. This example uses the indefinite length block syntax when data is being written back to the analyzer. More information about data transfer is available in Chapter 4, "Data Types and Encoding." All of the arrays listed in the `ASCDATA` example section can also be accessed using this data format.

Lines `30-70` are explained in the introduction to the example programs section. They determine which system controller is being used, set flags, and prepare the instrument for remote operation.

Lines `20-80` are explained in the introduction to the example programs section. They determine which system controller is being used, set flags, and prepare the instrument for remote operation.

```
1     !Filename:  REALDATA
2     !
3     ! Description:
4     !  1. Takes a sweep, and reads the formatted
5     !     data trace into an array.  The trace
6     !     is read as a definite length block.
7     !  2. Instructs you to remove DUT
8     !  3. Downloads the trace back to the analyzer
9     !     as an indefinite length block.
10    DIM A$[10],Data1(1:51)
20    INTEGER Digits,Bytes
30    IF POS(SYSTEM$("SYSTEM ID"),"HP 871") THEN
40      ASSIGN @Hp8711 TO 800
50    ELSE
60      ASSIGN @Hp8711 TO 716
70      ABORT 7
80      CLEAR 716
90    END IF
91    !
92    ! Set up the analyzer to measure 51 data points.
```

```
100   OUTPUT @Hp8711;"SENS1:SWE:POIN 51;*WAI"
101   !
102   ! Take a single sweep, leaving the analyzer
103   ! in trigger hold mode.
110   OUTPUT @Hp8711;"ABOR;:INIT1:CONT OFF;:INIT1;*WAI"
111   !
112   ! Select binary block transfer
120   OUTPUT @Hp8711;"FORM:DATA REAL,64"
121   !
130   ! Request the channel 1 formatted data array
140   ! from the analyzer.
160   OUTPUT @Hp8711;"TRAC? CH1FDATA"
161   !
162   ! Turn on ASCII formatting on the I/O path.
163   ! It is needed for reading the header
164   ! information.
170   ASSIGN @Hp8711;FORMAT ON
171   !
173   ! Get the data header.  "A$" will contain the
174   ! "#" character indicating a block data transfer.
175   ! "Digits" will contain the number of characters
176   ! for the number of bytes value which follows.
180   ENTER @Hp8711 USING "%,A,D";A$,Digits
181   !
182   ! Get the rest of the header.  The number of
183   ! bytes to capture in the data array will be
184   ! placed in "Bytes".  Note the use of "Digits"
185   ! in the IMAGE string.
190   ENTER @Hp8711 USING "%,"&VAL$(Digits)&"D";Bytes
191   !
192   ! Turn off ASCII formatting on the I/O path;
193   ! it is not needed for transferring binary
194   ! formatted data.
200   ASSIGN @Hp8711;FORMAT OFF
201   !
202   ! Get the data.
210   ENTER @Hp8711;Data1(*)
211   !
212   ! Turn on ASCII formatting again.
220   ASSIGN @Hp8711;FORMAT ON
221   !
```

```
222   ! Get the "end of data" character.
230   ENTER @Hp8711;A$
231   !
232   ! Display the first three numbers in the array.
240   DISP "Trace: ";Data1(1);Data1(2);Data1(3);"..."
241   !
242   ! Use this time to visually compare the
243   ! numbers to the visible data trace.
250   WAIT 5
251   !
252   ! Prompt the operator to disconnect the test
253   ! device and how to continue the program.
260   DISP "Disconnect the test device -- Press Continue"
270   PAUSE
271   !
272   ! Update the display line.
275   DISP "Taking a new sweep...";
276   !
277   ! Take a sweep so the display shows new data.
280   OUTPUT @Hp8711;":INIT1;*WAI"
285   DISP " Done."
290   WAIT 5
291   !
292   ! Send the header for an indefinite block length
293   ! data transfer.
295   DISP "Downloading saved trace...";
300   OUTPUT @Hp8711;"TRAC CH1FDATA, #0";
301   !
302   ! Turn off ASCII formatting.
310   ASSIGN @Hp8711;FORMAT OFF
311   !
312   ! Send the data array back to the analyzer.
320   OUTPUT @Hp8711;Data1(*),END
321   !
322   ! Turn on ASCII formatting again.
330   ASSIGN @Hp8711;FORMAT ON
340   DISP " Done!"
350   END
```

INTDATA Example Program

This program demonstrates how to read data arrays from the analyzer and write them back again. The `INTeger,16` data format is being used. This data format is the instrument's internal format. It should only be used to read data that will later be returned to the instrument.

The data array dimensioned in line `20` is different from the arrays in either `REAL,64` or `ASCii` examples. This is because each data point is represented by a set of three 16-bit integers. Another difference in using this data format is that all arrays cannot be accessed with it. The formatted data arrays `CH1FDATA` and `CH2FDATA` cannot be read using the `INTEGER` format.

Note that the analyzer outputs the data using the definite length block syntax. This example uses the indefinite length block syntax when data is being written back to the analyzer. More information about data transfer is available in Chapter 4, "Data Types and Encoding."

Lines `30-70` are explained in the introduction to the example programs section. They determine which system controller is being used, set flags, and prepare the instrument for remote operation.

```
1     !Filename:  INTDATA
2     !
3     ! Description:
4     !  1. Takes a sweep, and reads the formatted
5     !     data trace into an array.  The trace
6     !     is read as a definite length block.
7     !  2. Instructs you to remove DUT
8     !  3. Downloads the trace back to the analyzer
9     !     as an indefinite length block.
10    DIM A$[10]
20    INTEGER Digits,Bytes,Data1(1:51,1:3)
30    IF POS(SYSTEM$("SYSTEM ID"),"HP 871") THEN
40      ASSIGN @Hp8711 TO 800
50    ELSE
60      ASSIGN @Hp8711 TO 716
70      ABORT 7
80      CLEAR 716
90    END IF
```

```
91    !
93    ! Set up the analyzer to measure 51 data points.
100   OUTPUT @Hp8711;"SENS1:SWE:POIN 51;*WAI"
101   !
102   ! Take a single sweep, leaving the analyzer
103   ! in trigger hold mode.
110   OUTPUT @Hp8711;"ABOR;:INIT1:CONT OFF;:INIT1;*WAI"
111   !
112   ! Select binary block transfer
120   OUTPUT @Hp8711;"FORM:DATA INT,16"
121   !
130   ! Request the channel 1 unformatted data array
140   ! from the analyzer.
160   OUTPUT @Hp8711;"TRAC? CH1SDATA"
161   !
162   ! Turn on ASCII formatting on the I/O path;
163   ! it is needed for reading the header information.
170   ASSIGN @Hp8711;FORMAT ON
171   !
172   ! Get the data header.  "A$" will contain the
173   ! "#" character indicating a block data transfer.
174   ! "Digits" will contain the number of characters
175   ! for the number of bytes value which follows.
180   ENTER @Hp8711 USING "%,A,D";A$,Digits
181   !
182   ! Get the rest of the header.  The number of
183   ! bytes to capture in the data array will be
184   ! placed in "Bytes".  Note the use of "Digits"
185   ! in the IMAGE string.
190   ENTER @Hp8711 USING "%,"&VAL$(Digits)&"D";Bytes
191   !
192   ! Turn off ASCII formatting on the I/O path;
193   ! it is not needed for transferring binary
194   ! formatted data.
200   ASSIGN @Hp8711;FORMAT OFF
201   !
202   ! Get the data.
210   ENTER @Hp8711;Data1(*)
211   !
212   ! Turn on ASCII formatting again.
220   ASSIGN @Hp8711;FORMAT ON
```

```
221   !
222   ! Get the "end of data" character.
230   ENTER @Hp8711;A$
231   !
232   ! Display the first 3 numbers; there will
233   ! be no visible similarity between these
234   ! numbers and the data displayed on the
235   ! analyzer.
240   DISP "Trace: ";Data1(1,1);Data1(1,2);Data1(1,3);"..."
250   WAIT 5
251   !
252   ! Prompt the operator to disconnect the test
253   ! device and how to continue the program.
260   DISP "Disconnect the test device -- Press Continue"
270   PAUSE
271   !
272   ! Update the display line.
275   DISP "Taking a new sweep...";
276   !
277   ! Take a sweep so the display shows new data.
280   OUTPUT @Hp8711;":INIT1;*WAI"
285   DISP " Done."
290   WAIT 5
291   !
292   ! Send the header for an indefinite block length
293   ! data transfer.
295   DISP "Downloading saved trace...";
300   OUTPUT @Hp8711;"TRAC CH1SDATA, #0";
301   !
302   ! Turn off ASCII formatting.
310   ASSIGN @Hp8711;FORMAT OFF
311   !
312   ! Send the data back to the analyzer.
320   OUTPUT @Hp8711;Data1(*),END
321   !
322   ! Turn on ASCII formatting.
330   ASSIGN @Hp8711;FORMAT ON
340   DISP "Done!"
350   END
```

TRANCAL	Performing a transmission calibration. The calibration is **User Defined** (performed over the instruments current source settings). This example also demonstrates the use of the `*OPC?` command.
REFLCAL	Performing a reflection calibration. The calibration is **Full Band** (performed over the instrument's preset source settings). This example also demonstrates the detection of front panel key presses, the use of softkeys, and the use of the `*OPC?` command.
LOADCALS	Uploading and downloading correction arrays. The data transfer is performed in the 16-bit integer format. The arrays must be dimensioned properly for both the number of data points and the format of the data being transferred.
CALKIT	Instrument state file for downloading User Defined cal kit definitions. This example is NOT a program. It is an instrument state file example. This type of file enables the user to calibrate the analyzer for use with connector types that are not in the firmware. See "Writing and Editing Your Own Cal Kit File" in Chapter 6 of the *User's Guide*.

TRANCAL Example Program

This program demonstrates a transmission calibration performed over user-defined source settings (frequency range, power and number of points). The operation complete query is used at each step in the process to make sure the steps are taken in the correct order. More information on calibration is available in the *User's Guide*.

Lines 10-70 are explained in the introduction to the example programs section. They determine which system controller is being used, set flags, and prepare the instrument for remote operation.

```
1      ! Filename:  TRANCAL
2      !
3      ! Guide user through a transmission cal.
4      !
10     IF POS(SYSTEM$("SYSTEM ID"),"HP 871") THEN
20       ASSIGN @Hp8711 TO 800
30     ELSE
40       ASSIGN @Hp8711 TO 716
50       ABORT 7
60       CLEAR 716
70     END IF
71     !
72     ! Configure the analyzer to measure transmission
73     ! on channel 1.
80     OUTPUT @Hp8711;"SENS1:FUNC 'XFR:POW:RAT 2,0'
       ;DET NBAN;*WAI"
81     !
82     ! Select a calibration kit type.
90     OUTPUT @Hp8711;"SENS:CORR:COLL:CKIT 'COAX,7MM,
       TYPE-N,50,FEMALE'"
91     !
92     ! Select a transmission calibration for the current
93     ! analyzer settings.  The "IST:OFF" ensures that
94     ! the current settings will be used.
100    OUTPUT @Hp8711;"SENS1:CORR:COLL:IST OFF;METH TRAN1"
101    !
102    ! Prompt the operator to make a through
```

```
103   ! connection.
110   DISP "Connect THRU - Press Continue"
120   PAUSE
130   DISP "Measuring THRU"
131   !
132   ! Analyzer measures the through.
140   OUTPUT @Hp8711;"SENS1:CORR:COLL STAN1;*OPC?"
141   !
142   ! Wait until the measurement is complete.
150   ENTER @Hp8711;Opc
160   DISP "Calculating Error Coefficients"
161   !
162   ! Tell the analyzer to calculate the
163   ! error coefficients after the measurement
164   ! is made, and then save for use during
165   ! subsequent transmission measurements.
166   ! Note that this is not the same as using
167   ! the SAVE RECALL key functionality.
170   OUTPUT @Hp8711;"SENS1:CORR:COLL:SAVE;*OPC?"
171   !
172   ! Wait for the calculations and save to be
173   ! completed.
180   ENTER @Hp8711;Opc
190   DISP "User Defined TRANSMISSION CAL COMPLETED!"
200   END
```

REFLCAL Example Program

This program demonstrates a reflection calibration performed over the preset source settings (frequency range, power and number of points). The operation complete query is used at each step in the process to make sure the steps are taken in the correct order. More information on calibration is available in the *User's Guide*.

Lines 20-100 are explained in the introduction to the example programs section. They determine which system controller is being used, set flags, and prepare the instrument for remote operation.

```
1     !Filename:  REFLCAL
2     !
3     ! Guide user through a reflection cal.
4     !
10    DIM Msg$[50]
20    IF POS(SYSTEM$("SYSTEM ID"),"HP 871") THEN
30      ASSIGN @Hp8711 TO 800
40      Internal=1
50    ELSE
60      ASSIGN @Hp8711 TO 716
70      Internal=0
80      ABORT 7
90      CLEAR 716
100   END IF
101   !
102   ! Configure the analyzer to measure
103   ! reflection on channel 1.
110   OUTPUT @Hp8711;"SENS1:FUNC 'XFR:POW:RAT 1,0'
      ;DET NBAN;*WAI"
111   !
113   ! Select Calibration Kit for 50 ohm instruments.
120   OUTPUT @Hp8711;"SENS:CORR:COLL:CKIT 'COAX,7MM,
      TYPE-N,50,FEMALE'"
121   !
123   ! Select Calibration Kit for 75 ohm instruments.
124   ! (Comment out the 50 ohm line above and uncomment the line
125   ! below.)
```

```
127   ! OUTPUT @Hp8711;"SENS:CORR:COLL:CKIT 'COAX,7MM,
        TYPE-N,75,FEMALE'"
128   !
129   ! Select a reflection calibration for the current
130   ! analyzer settings.  The "IST:OFF" ensures that
131   ! current settings will be used.
133   OUTPUT @Hp8711;"SENS1:CORR:COLL:IST OFF;METH REFL3"
134   !
135   ! Prompt the operator to connect an open.
140   Msg$="Connect OPEN"
150   GOSUB Get_continue
160   DISP "Measuring OPEN"
161   !
162   ! Measure the open.
170   OUTPUT @Hp8711;"SENS1:CORR:COLL STAN1;*OPC?"
171   !
172   ! Wait until the measurement of the open
173   ! is complete.
180   ENTER @Hp8711;Opc
181   !
182   ! Prompt the operator to connect a short.
190   Msg$="Connect SHORT"
200   GOSUB Get_continue
210   DISP "Measuring SHORT"
211   !
212   ! Measure the short.
220   OUTPUT @Hp8711;"SENS1:CORR:COLL STAN2;*OPC?"
221   !
222   ! Wait until measurement of the short
223   ! is complete.
230   ENTER @Hp8711;Opc
231   !
232   ! Prompt operator to connect a load.
240   Msg$="Connect LOAD"
250   GOSUB Get_continue
260   DISP "Measuring LOAD"
261   !
262   ! Measure the load.
270   OUTPUT @Hp8711;"SENS1:CORR:COLL STAN3;*OPC?"
271   ! Wait until measurement of the load
272   ! is complete.
```

```
280   ENTER @Hp8711;Opc
290   DISP "Calculating Error Coefficients"
291   !
292   ! Tell the analyzer to calculate the
293   ! error coefficients, and then save
294   ! for use during subsequent reflection
295   ! measurements.  Note that this is not
296   ! the same as using the SAVE RECALL key
297   ! functionality.
300   OUTPUT @Hp8711;"SENS1:CORR:COLL:SAVE;*OPC?"
301   !
302   ! Wait for the calculations to be completed
303   ! and the calibration saved.
310   ENTER @Hp8711;Opc
320   DISP "Full Band REFLECTION CAL COMPLETED!"
330   STOP
331   !
340 Get_continue: ! Subroutine to handle operator prompts.
341   !
342   ! "Internal" is determined above based on the
343   ! controller.
350   IF Internal=1 THEN
351     !
352     ! If internal control, then use the display
353     ! line for the prompt.
360     DISP Msg$&" - Press Measure Standard"
361     !
362     ! Use the softkey 2 for the response; loop
363     ! while waiting for it to be pressed.
370     ON KEY 2 LABEL "Measure    Standard" RECOVER Go_on
380     LOOP
390     END LOOP
400   ELSE
401     !
402     ! If external control, clear the key queue
403     ! so previous key presses will not interfere.
410     OUTPUT @Hp8711;"SYST:KEY:QUE:CLE"
411     !
412     ! Use the BEGIN key for the response.
420     DISP Msg$&" - Press BEGIN to continue"
421     !
```

```
422     ! Turn on the key queue to trap all key
423     ! presses.
430     OUTPUT @Hp8711;"SYST:KEY:QUE ON"
431     !
432     ! Loop while waiting for a key to be
433     ! pressed.
440     LOOP
441       ! Query the device status condition
442       ! register.
450       OUTPUT @Hp8711;"STAT:DEV:COND?"
460       ENTER @Hp8711;Dev_cond
461       !
462       ! Check the bit that indicates a key press.
470       IF BIT(Dev_cond,0)=1 THEN
480         OUTPUT @Hp8711;"SYST:KEY?"
490         ENTER @Hp8711;Key_code
500       END IF
501     !
502     ! Stop looping if the BEGIN key was pressed.
510     EXIT IF Key_code=40
520     END LOOP
530     Key_code=0
540   END IF
541   !
550 Go_on: ! Subroutine to turn off the softkeys
551        ! on the analyzer and the computer,
553        ! and return to main body of the
554        ! program.
560   OFF KEY
570   RETURN
580   END
```

LOADCALS Example Program

This program demonstrates how to read the correction arrays from the analyzer and write them back again. The `INTeger,16` data format is being used because the data does not need to be interpreted, only stored and retrieved. More information about calibration is available in the *User's Guide*.

The size of the arrays into which the data is read is critical. If they are not dimensioned correctly the program will not work. Most correction arrays, including the factory default (`DEF`) and the full band (`FULL`, preset source settings) arrays have 801 points. For user defined calibrations (`USER`) the number of points must be determined. If the number of points is other than 801, lines `30` and `280` will need to be changed to allocate arrays for the correct number of points. The number of points can be found by reading the correction array's header and determining the size as shown in the example below.

Lines `40-70` are explained in the introduction to the example programs section. They determine which system controller is being used, set flags, and prepare the instrument for remote operation.

```
1      !Filename:  LOADCALS
2      !
3      ! Description:
4      !  1. Query the calibration arrays, based on
5      !     the current measurement (trans/refl).
6      !  2. Change number of points to 801
7      !  3. Download the calibration arrays back
8      !     into the analyzer.
9      !
10     DIM Func$[20],A$[10]
20     INTEGER Swap,Arrays,Digits,Bytes,Points
30     INTEGER Corr1(1:801,1:3),Corr2(1:801,1:3)
       ,Corr3(1:801,1:3)
40     IF POS(SYSTEM$("SYSTEM ID"),"HP 871") THEN
50       ASSIGN @Hp8711 TO 800
60     ELSE
70       ASSIGN @Hp8711 TO 716
80     END IF
81     !
```

```
90    ! Query the measurement parameter.
120   OUTPUT @Hp8711;"SENS1:FUNC?"
121   !
122   ! Read the analyzer's response.
130   ENTER @Hp8711;Func$
131   !
132   ! Set up a SELECT/CASE depending on the
133   ! response.
140   SELECT Func$
141     !
143     ! This is the transmission case, a ratio of
144     ! the powers measured by detector 2 (B) and
145     ! detector 0 (R).
150     CASE """XFR:POW:RAT 2, 0"""
151       !
152       ! The transmission calibration has only one
153       ! correction array.
160       Arrays=1
161     !
162     ! This is the reflection case, a ratio of
163     ! the powers measured by detector 1 (A) and
164     ! detector 0 (R).
170     CASE """XFR:POW:RAT 1, 0"""
171       !
172       ! The reflection calibration has 3 correction
173       ! arrays.
180       Arrays=3
190   END SELECT
191   !
192   ! Select the 16 bit integer binary data format.
200   OUTPUT @Hp8711;"FORM:DATA INT,16"
201   !
202   ! Select normal byte order.
210   OUTPUT @Hp8711;"FORM:BORD NORM"
220   !
221   ! Request the first correction array from the a
222   ! analyzer.
230   OUTPUT @Hp8711;"TRAC? CH1SCORR1"
231   !
232   ! Turn on ASCII formatting on the I/O path
233   ! to read the header information.
```

```
240   ASSIGN @Hp8711;FORMAT ON
241   !
242   ! Get the header, including the number of
243   ! of characters that will hold the number
244   ! of bytes value which follows.
250   ENTER @Hp8711 USING "%,A,D";A$,Digits
251   !
252   ! Get the rest of the header.  The number
253   ! of bytes to capture in the correction
254   ! array will be placed in "Bytes".  Note
255   ! the use of "Digits" in the IMAGE string.
260   ENTER @Hp8711 USING "%,"&VAL$(Digits)&"D";Bytes
261   !
262   ! Determine the number of points from the
263   ! number of bytes (6 bytes per point).
270   Points=Bytes/6
271   !
272   ! This example was set up in line 30 above
273   ! for 801 points.  Edit this line and line 30
274   ! to allow other dimensions.
280   IF Points<>801 THEN
290     DISP "Arrays are not dimensioned for this calibration"
300     STOP
310   END IF
320   DISP "Uploading (querying) calibration arrays . . . ."
321   !
322   ! Turn off ASCII formatting on the I/O path.
330   ASSIGN @Hp8711;FORMAT OFF
331   !
332   ! Get the first error correction array.
340   ENTER @Hp8711;Corr1(*)
341   !
342   ! Turn on ASCII formatting.
350   ASSIGN @Hp8711;FORMAT ON
351   !
352   ! Get the "end of data" character.
360   ENTER @Hp8711;A$
361   !
362   ! For the reflection there are two more
363   ! arrays to read.
370   IF Arrays=3 THEN
```

```
371     !
372     ! Request and read in the second
373     ! correction array.
380     OUTPUT @Hp8711;"TRAC? CH1SCORR2"
390     Read_array(@Hp8711,Corr2(*))
391     !
392     ! Request and read in the third
393     ! correction array.
400     OUTPUT @Hp8711;"TRAC? CH1SCORR3"
410     Read_array(@Hp8711,Corr3(*))
420   END IF
430   DISP "Calibration arrays have been uploaded."
440   WAIT 5
450   DISP "Downloading (setting) calibration arrays . . . ."
451   !
452   ! Turn off correction before writing a
453   ! calibration back into the analyzer.
460   OUTPUT @Hp8711;"SENS1:CORR:STAT OFF"
461   !
462   ! Set the number of points for the correction
463   ! arrays.  (Not necessary in this example,
464   ! but shown for emphasis.)
470   OUTPUT @Hp8711;"SENS1:SWE:POIN";Points
471   !
472   ! Prepare the analyzer to receive the first
473   ! correction array in the indefinite block
474   ! length format.
480   OUTPUT @Hp8711;"TRAC CH1SCORR1, #0";
481   !
482   ! Turn off ASCII formatting.
490   ASSIGN @Hp8711;FORMAT OFF
491   !
492   ! Send the first correction array to the
493   ! analyzer.  The array transfer is
494   ! terminated with the "END" signal.
500   OUTPUT @Hp8711;Corr1(*),END
501   !
502   ! Turn on ASCII formatting.
510   ASSIGN @Hp8711;FORMAT ON
511   !
512   ! For a reflection array download, there
```

```
513   ! are two more arrays.
520   IF Arrays=3 THEN
521     !
522     ! Prepare the analyzer to receive the
523     ! 2nd array, then output it.
530     OUTPUT @Hp8711;"TRAC CH1SCORR2, ";
540     Write_array(@Hp8711,Corr2(*))
541     !
542     ! Prepare the analyzer to receive the
543     ! 3rd array, then output it.
550     OUTPUT @Hp8711;"TRAC CH1SCORR3, ";
560     Write_array(@Hp8711,Corr3(*))
570   END IF
571   !
572   ! Turn on the calibration just downloaded.
580   OUTPUT @Hp8711;"SENS1:CORR:STAT ON;*WAI"
590   DISP "Calibration arrays have been downloaded."
600   END
601   !
602   ! Subprogram for reading binary data array from
603   ! the analyzer.  The command requesting a specific
604   ! data array has already been sent prior to
605   ! calling this subprogram.
606   !
610   SUB Read_array(@Hp8711,INTEGER Array(*))
620     DIM A$[10]
630     INTEGER Digits,Bytes
640     ASSIGN @Hp8711;FORMAT ON
650     ENTER @Hp8711 USING "%,A,D";A$,Digits
660     ENTER @Hp8711 USING "%,"&VAL$(Digits)&"D";Bytes
670     ASSIGN @Hp8711;FORMAT OFF
680     ENTER @Hp8711;Array(*)
690     ASSIGN @Hp8711;FORMAT ON
700     ENTER @Hp8711;A$
710   SUBEND
711   !
713   ! Subprogram for writing binary data array to
714   ! the analyzer.  The command requesting a specific
715   ! data array has already been sent prior to
716   ! calling this subprogram.
717   !
```

```
720   SUB Write_array(@Hp8711,INTEGER Array(*))
730     OUTPUT @Hp8711;"#0";
740     ASSIGN @Hp8711;FORMAT OFF
750     OUTPUT @Hp8711;Array(*),END
760     ASSIGN @Hp8711;FORMAT ON
770   SUBEND
```

CALKIT Example Program

This instrument state file demonstrates the type of file required to download user-defined calibration kits. To see an example of using this feature, refer to "Writing or Editing Your Own Cal Kit File", in Chapter 6 of the *User's Guide*.

```
10  !$  Standard Definitions for HP 85054B Precision
        Type-N Cal Kit.
11  !
12  !$  This is a Cal Kit definition file, which
13  !$  uses the same format as a BASIC program.
14  !$  Lines that contain "!$" are comments.
15  !$
16  !$  Put your Cal Kit file on a disk, and use the
17  !$  analyzer's [SAVE/RECALL] [Recall State] keys
18  !$  to load your custom Cal Kit into the analyzer.
20  !
30  !$  Definitions for 50 Ohm jack (FEMALE center
        contact) test
40  !$  ports, plug (MALE center contact) standards.
50  !
60  ! OPEN:  $ HP 85054-60027 Open Circuit Plug
70  !    Z0  50.0 $ Ohms
80  !    DELAY  57.993E-12 $ Sec
90  !    LOSS  0.8E+9 $ Ohms/Sec
100 !    C0  88.308E-15 $ Farads
110 !    C1  1667.2E-27 $ Farads/Hz
120 !    C2 -146.61E-36 $ Farads/Hz^2
130 !    C3  9.7531E-45 $ Farads/Hz^3
140 !
150 ! SHORT:  $ HP 85054-60025 Short Circuit Plug
160 !    Z0  50.0 $ Ohms
170 !    DELAY  63.078E-12 $ Sec
180 !    LOSS  8.E+8 $ Ohms/Sec
190 !
200 ! LOAD:  $ HP 00909-60011 Broadband Load Plug
210 !    Z0  50.0 $ Ohms
220 !    DELAY  0.0 $ Sec
230 !    LOSS   0.0 $ Ohms/Sec
```

```
240 !
250 ! THRU:  $ HP 85054-60038 Plug to Plug Adapter
260 !    ZO  50.0 $ Ohms
270 !    DELAY  196.0E-12 $ Sec
280 !    LOSS   2.2E+9 $ Ohms/Sec
290 !
300 END
```

Instrument State and Save/Recall

LEARNSTR	Using the learn string to upload and download instrument states.
SAVERCL	Saving and recalling instrument states, calibrations and data. The example also demonstrates saving data in an ASCII file that includes both magnitude and frequency information.

LEARNSTR Example Program

This program demonstrates how to upload and download instrument states using the learn string. The learn string is a fast and easy way to read an instrument state. It is read out using the *LRN? query (an IEEE 488.2 common commands). To restore the learn string simply output the string to the analyzer.

The learn string contains a mnemonic at the beginning that tells the analyzer to restore the instrument state.

The learn string is transferred as a block. The header is ASCII formatted and the data is in the instrument's internal binary format. The number of bytes in the block of data is determined by the instrument state (no more than 20000 bytes).

```
"SYST:SET #<digits><bytes><learn string data>"
```

The "long" learnstring, in addition to the instrument state like the normal learnstring, will include data and calibration arrays IF they are selected using the Define Save function under SAVE RECALL. The SCPI equivalent command for saving the calibration arrays is added before the "long" learnstring query.

Lines 20-80 are explained in the introduction to the example programs section. They determine which system controller is being used and prepare the instrument for remote operation.

```
1      !Filename:  LEARNSTR
2      !
3      ! Description:
4      !   1. Query the learn string
5      !   2. Preset the analyzer
6      !   3. Send the learn string,
7      !      restoring the previous state.
8      !
10     DIM Learnstr$[20000]
20     IF POS(SYSTEM$("SYSTEM ID"),"HP 871") THEN
30       ASSIGN @Hp8711 TO 800
40     ELSE
50       ASSIGN @Hp8711 TO 716
60       ABORT 7
70       CLEAR 716
80     END IF
81     !
82     ! Request the learnstring.  If the "long"
83     ! learnstring is desired, comment the line
84     ! below, and uncomment the line after it.
85     ! The "long" learnstring, in addition to
86     ! the instrument state like the normal
87     ! learnstring, will include data and
88     ! calibration arrays IF they are selected
89     ! using the Define Save function under
90     ! SAVE RECALL.  The SCPI equivalent command
91     ! for saving the calibration arrays is
92     ! added before the "long" learnstring query.
94     OUTPUT @Hp8711;"*LRN?"
95     ! OUTPUT @Hp8711;"MMEM:STOR:STAT:CORR ON;
         :SYST:SET:LRNL?"
96     !
97     ! Read the learnstring from the analyzer.
98     ! The USING "-K" format allows the data
99     ! being transmitted to include characters
100    ! (such as the line feed character) that
101    ! would otherwise terminate the learnstring
```

```
102   ! request prematurely.
103   ENTER @Hp8711 USING "-K";Learnstr$
110   DISP "Learn string has been read"
120   WAIT 5
121   !
122   ! Preset the analyzer.
130   OUTPUT @Hp8711;"SYST:PRES;*OPC?"
131   !
132   ! Wait for the preset operation to complete.
140   ENTER @Hp8711;Opc
150   DISP "Instrument has been PRESET"
160   WAIT 5
161   !
162   ! Output the learnstring to the analyzer.
163   ! The mnemonic is included in the string,
164   ! so no command preceding "Learnstr$" is
165   ! necessary.
170   OUTPUT @Hp8711;Learnstr$
180   DISP "Instrument state has been restored"
190   END
```

SAVERCL Example Program

This program demonstrates how to save instrument states, calibrations and data to a mass storage device. The device used in this example is the analyzer's internal 3.5" disk drive. The only change needed to use this program with the internal non-volatile memory is to change the mass storage unit specifier.

The four choices are the internal floppy disk drive (`INT:`), the internal non-volatile memory, (`MEM:`), the internal volatile memory, (`RAM:`), and an external HP-IB floppy disk drive (`EXT:`). To perform a save/recall to an external disk drive requires passing control of the HP-IB from the controller to the analyzer. For more information on passing control of the bus refer to Chapter 3, "Passing Control," or the `PASSCTRL` example program.

Lines `10-70` are explained in the introduction to the example programs section. They determine which system controller is being used, set flags, and prepare the instrument for remote operation.

Lines `80-130` are an example of saving an instrument state and calibration on the internal floppy disk drive.

Lines `190-200` are an example of recalling that instrument state and calibration.

Lines `210-230` are an example of saving a data trace (magnitude and frequency values) to an ASCII formatted file on the internal floppy disk drive. This file cannot be recalled into the instrument. It can, however, be imported directly into spreadsheets and word processors.

```
1      !Filename:  SAVERCL
2      !
10     IF POS(SYSTEM$("SYSTEM ID"),"HP 871") THEN
20       ASSIGN @Hp8711 TO 800
30     ELSE
40       ASSIGN @Hp8711 TO 716
50       ABORT 7
60       CLEAR 716
70     END IF
71     !
72     ! Select the internal floppy disk drive
```

```
73    ! as the mass storage device.
80    OUTPUT @Hp8711;"MMEM:MSIS 'INT:'"
81    !
82    ! Turn on the saving of the instrument state
83    ! as part of the "Define Save" function under
84    ! SAVE RECALL.
90    OUTPUT @Hp8711;"MMEM:STOR:STAT:IST ON"
91    !
92    ! Turn on the saving of the calibration
93    ! as part of the "Define Save" function under
94    ! SAVE RECALL.
100   OUTPUT @Hp8711;"MMEM:STOR:STAT:CORR ON"
101   !
102   ! Turn off the saving of the data
103   ! as part of the "Define Save" function under
104   ! SAVE RECALL.
110   OUTPUT @Hp8711;"MMEM:STOR:STAT:TRAC OFF"
111   !
112   ! Save the current defined state (STAT 1) into
113   ! a file named "FILTER".  Use *OPC? to make
114   ! sure the operation is completed before any
115   ! other operation begins.
120   OUTPUT @Hp8711;"MMEM:STOR:STAT 1,'FILTER';*OPC?"
130   ENTER @Hp8711;Opc
140   DISP "Instrument state and calibration have been saved"
141   !
142   ! Preset the instrument so that the change in state
143   ! is easy to see when it is recalled.
150   OUTPUT @Hp8711;"SYST:PRES;*OPC?"
160   ENTER @Hp8711;Opc
170   DISP "Instrument has been PRESET"
180   WAIT 5
181   !
182   ! Recall the file "FILTER" from the internal
183   ! floppy disk drive.  This becomes the new instrument
184   ! state.  Use of the *OPC query allows hold off of
185   ! further commands until the analyzer is reconfigured.
190   OUTPUT @Hp8711;"MMEM:LOAD:STAT 1,'INT:FILTER';*OPC?"
200   ENTER @Hp8711;Opc
201   !
202   ! Take a single sweep to ensure that valid measurement
```

```
203   ! data is acquired.
210   OUTPUT @Hp8711;"ABOR;:INIT:CONT OFF;:INIT;*WAI"
220   DISP "Instrument state and calibration have been recalled"
221   !
222   ! Save that measurement data into an ASCII file
223   ! called "DATA0001" on the internal floppy disk drive.
230   OUTPUT @Hp8711;"MMEM:STOR:TRAC CH1FDATA,'INT:DATA0001'"
240   DISP "Data has been saved (ASCII format)"
250   END
```

Hardcopy Control

PRINTPLT	Using the serial and parallel ports for hardcopy output. The example also demonstrates plotting test results to an HPGL file.
PASSCTRL	Using pass control and the HP-IB for hardcopy output. The example uses an HP-IB printer.
FAST_PRT	Provides fast graph dumps to PCL5 printers.

PRINTPLT Example Program

This program demonstrates how to send a hardcopy to a printer on the serial interface. This is done by selecting the appropriate device, setting up the baud rate and hardware handshaking, and sending the command to print or plot. The *OPC? query is used in this example to indicate when the printout is complete. Another method of obtaining the same results is to monitor the Hardcopy in Progress bit (bit 9 in the Operational Status Register). More information on printing or plotting is available in the *User's Guide*.

Lines 10-70 are explained in the introduction to the example programs section. They determine which system controller is being used, set flags, and prepare the instrument for remote operation.

Lines 80-150 demonstrate sending a hardcopy output to a printer connected to the serial port. The same program could be used to send hardcopy output to a device on the parallel port. The only changes would be deleting lines 100-110 and changing line 90 to read HCOP:DEV:PORT PAR.

Lines 160-260 demonstrate how to create an HPGL file (plotter language) and send it to the disk in the internal 3.5" disk drive. HPGL files are supported by many applications including the leading word processors and desktop publishing products.

```
1     !Filename:  PRINTPLT
2     !
3     ! Description:
4     !   1. Select serial port.  Configure it.
5     !   2. Dump table of trace values
6     !   3. Re-configure hardcopy items to dump
7     !   4. Dump HP-GL file to internal floppy
8     !
10    IF POS(SYSTEM$("SYSTEM ID"),"HP 871") THEN
20      ASSIGN @Hp8711 TO 800
30    ELSE
40      ASSIGN @Hp8711 TO 716
50      ABORT 7
60      CLEAR 716
70    END IF
71    !
72    ! Select the output language (PCL-Printer
```

```
73    ! Control Language) and the hardcopy port
74    ! to serial.
80    OUTPUT @Hp8711;"HCOP:DEV:LANG PCL;PORT SER"
81    !
82    ! Select baud rate to 19200.
90    OUTPUT @Hp8711;"SYST:COMM:SER:TRAN:BAUD 19200"
91    !
92    ! Select the handshaking protocol to Xon/Xoff.
100   OUTPUT @Hp8711;"SYST:COMM:SER:TRAN:HAND XON"
101   !
102   ! Select the type of output to table, which
103   ! is the same as the softkey List Trace
104   ! Values under the Define Hardcopy menu.
110   OUTPUT @Hp8711;"HCOP:DEV:MODE TABL"
111   !
112   ! Send the command to start a hardcopy, and
113   ! use *OPC query to make sure the hardcopy is
114   ! complete before continuing.
120   OUTPUT @Hp8711;"HCOP;*OPC?"
130   ENTER @Hp8711;Opc
140   DISP "Hardcopy to serial printer - COMPLETE!"
141   !
142   ! Select the HPGL language and the hardcopy
143   ! port to be the currently selected mass memory
144   ! device.
150   OUTPUT @Hp8711;"HCOP:DEV:LANG HPGL;PORT MMEM"
151   !
152   ! Include trace data in the plot.
160   OUTPUT @Hp8711;"HCOP:ITEM:TRAC:STAT ON"
161   !
162   ! Turn graticule off in the hardcopy dump.
170   OUTPUT @Hp8711;"HCOP:ITEM:GRAT:STAT OFF"
171   !
172   ! Include frequency and measurement
173   ! annotation.
180   OUTPUT @Hp8711;"HCOP:ITEM:ANN:STAT ON"
181   !
182   ! Include marker symbols.
190   OUTPUT @Hp8711;"HCOP:ITEM:MARK:STAT ON"
191   !
192   ! Include title (and/or time/date if
```

```
193   ! already selected).
200   OUTPUT @Hp8711;"HCOP:ITEM:TITL:STAT ON"
201   !
202   ! Define the hardcopy to be both the graph
203   ! and a marker table.
210   OUTPUT @Hp8711;"HCOP:DEV:MODE GMAR"
211   !
212   ! Send the command to plot and use *OPC
213   ! query to wait for finish.
220   OUTPUT @Hp8711;"HCOP;*OPC?"
230   ENTER @Hp8711;Opc
240   DISP "Plot to floppy disk - COMPLETE!"
250   END
```

PASSCTRL Example Program

This program demonstrates how to send a hardcopy to an HP-IB printer. This is done by passing active control of the bus to the analyzer so it can control the printer. More information about passing control to the analyzer is available in Chapter 3, "Passing Control."

Lines 10-90 are explained in the introduction to the example programs section. They determine which system controller is being used, set flags, and prepare the instrument for remote operation.

```
1     !Filename:  PASSCTRL
2     !
3     ! Description:
4     !   External controller runs this program, which
5     !   instructs the analyzer to perform a hardcopy
6     !   and then passes control to the analyzer.
7     !   Analyzer performs hardcopy over HP-IB
8     !   to printer at 701, then passes control back.
9     !
10    !   This program only works on controllers which
11    !   implement pass control properly.  HP s700
12    !   computers running BASIC-UX 7.0x will need
13    !   to upgrade to a newer BASIC-UX version.
14    !
15    IF POS(SYSTEM$("SYSTEM ID"),"HP 871") THEN
20      ASSIGN @Hp8711 TO 800
30      Internal=1
40    ELSE
50      ASSIGN @Hp8711 TO 716
60      Internal=0
70      ABORT 7
80      CLEAR 716
90    END IF
91    !
92    ! Select the language to PCL (Printer
93    ! Control Language) and the output port
94    ! to HP-IB.
100   OUTPUT @Hp8711;"HCOP:DEV:LANG PCL;PORT GPIB"
101   !
```

```
102   ! Select the HP-IB address for the hardcopy
103   ! device on the HP-IB.
110   OUTPUT @Hp8711;"SYST:COMM:GPIB:HCOP:ADDR 1"
111   !
112   ! Set the output to graph only.
120   OUTPUT @Hp8711;"HCOP:DEV:MODE GRAP"
121   !
122   ! If the internal controller is being used...
130   IF Internal=1 THEN
131     !
132     ! then make it System Controller of HP-IB
140     OUTPUT @Hp8711;"SYST:COMM:GPIB:CONT ON"
150   END IF
151   !
152   ! Clear Status Registers
160   OUTPUT @Hp8711;"*CLS"
161   !
162   ! Enable the Request Control bit in the Event
163   ! Status Register.
170   OUTPUT @Hp8711;"*ESE 2"
171   !
172   ! Clear the Service Request enable register;
173   ! SRQ is not being used.
180   OUTPUT @Hp8711;"*SRE 0"
181   !
182   ! Send the hardcopy command to start the
183   ! print.
190   OUTPUT @Hp8711;"HCOP"
200   LOOP
201     !
202     ! Read the status byte using Serial Poll.
210     Stat=SPOLL(@Hp8711)
211   !
212   ! Exit when the analyzer requests active control
213   ! of HP-IB from the system controller.
220   EXIT IF BIT(Stat,5)=1
230   END LOOP
231   !
233   ! Now system controller passes control to
234   ! the analyzer.
240   PASS CONTROL @Hp8711
```

```
250   DISP "Hardcopy in Progress...";
260   IF Internal=1 THEN
261     ! If using the internal IBASIC controller,
262     ! then use the *OPC query method to wait
263     ! for hardcopy completion.
270     OUTPUT @Hp8711;"*OPC?"
280     ENTER @Hp8711;Opc
290   ELSE
291     ! If external computer control, then...
300     LOOP
301       !
303       ! Monitor the HP-IB status in the
304       ! external computer's HP-IB status
305       ! register.  Here, the HP-IB interface
306       ! code 7 register 6 status is requested
307       ! and put into "Hpib".
308       DISP ".";
309       WAIT 1   ! No need to poll rapidly
310       STATUS 7,6;Hpib
311     !
312     ! When active control is returned to the
313     ! system controller (bit 6 set), then exit.
314     ! (This fails on s700s running BASIC 7.0x)
320     EXIT IF BIT(Hpib,6)=1
330     END LOOP
340   END IF
350   DISP "HARDCOPY COMPLETE!"
360   END
```

FAST_PRT Example Program

This program configures a PCL5 printer to accept HP-GL graphics commands from the analyzer. The program executes a hardcopy which causes the analyzer to send HP-GL commands to the parallel port PCL5 printer. Provides up to 10× speed improvement of some hardcopies.

```
10   ! FAST_PRT
20   !
30   ! This program is designed to set up a PCL5 printer
40   ! connected to the parallel port of the analyzer to
50   ! accept HP-GL syntax.  HP-GL gives fast graph dumps.
60   !
70   ! Connect your PCL5 printer to the parallel
       printer of the
80   ! analyzer, then run the program.
90   !
100  ! Once the parallel printer has been configured
       to accept
110  ! HPGL commands, a hardcopy is done, the printer is
120  ! reset to normal mode, and the page is ejected.
130  !
140   DIM A$[50]
150  !
160   IF POS(SYSTEM$("SYSTEM ID"),"HP 871") THEN
170     ASSIGN @Rfna TO 800
180     Internal=1
190     Isc=8
200   ELSE
210     ASSIGN @Rfna TO 716
220     Internal=0
230     Isc=7
240     ABORT 7
250     CLEAR 716
260   END IF
270  !
280  ! Define the hardcopy device
290   OUTPUT @Rfna;"HCOP:DEV:LANG HPGL;PORT CENT"
300  !
```

```
310  ! Define PCL5 escape codes needed to set up
       HPGL commands:
320   DATA @E                   ! Reset, Eject page
330   DATA @&l2A                ! Page size 8.5 x 11
340   DATA @&a0L@&a4000M@&l0E   ! No margins
350   DATA @*c7400x5650y        ! 10.28 x 7.85 size 720/in
360  !DATA @*c5500x5650y        ! if Marker table included
370  !DATA @*c4255x3283y        ! portrait,remove
                                  Landscape Mode
380   DATA @&l1O                ! Landscape Mode
390   DATA @*p50x50y            ! Cursor to anchor point
400   DATA @*c0T                ! Set picture anchor point
410   DATA @*r-3U               ! CMY Palette
420  !DATA @*r1U                ! Monochrome optional
430   DATA @%1B                 ! HPGL Mode
440   DATA $                    ! dump plot
450   DATA @%0A                 ! Exit HPGL Mode
460   DATA @E                   ! Eject page
470   DATA DONE                 ! End of defined escape codes
480  !
490  ! Send the defined escape codes to the printer
500   LOOP
510     READ A$
520   EXIT IF A$="DONE"
530     FOR I=1 TO LEN(A$)
540       SELECT A$[I;1]
550       CASE "@" ! Escape Character
560         OUTPUT @Rfna;"DIAG:PORT:WRITE 15,0,27"
570       CASE "$" ! Dump the plot
580         OUTPUT @Rfna;"HCOP;*WAI"
590       CASE ELSE! Send Character
600         OUTPUT @Rfna;"DIAG:PORT:WRITE 15,0,
            ";NUM(A$[I;1])
610       END SELECT
620     NEXT I
630   END LOOP
640  !
650   END
```

Service Request

SRQ Generating a service request interrupt. The example uses the status reporting structure to generate an interrupt as soon as averaging is complete.

SRQ Example Program

This program demonstrates generating a service request interrupt. The SRQ is used to indicate when averaging is complete. More information on service requests and the status registers is available in Chapter 5, "Using the Status Registers."

In this program, the `STATus:PRESet` executed in line `130` has the effect of setting all bits in the averaging status transition registers (positive transitions to `0`, negative transitions to `1`). It also sets up the operational status transition registers (positive transitions to `1`, negative transitions to `0`). These are the states needed to generate an interrupt when averaging is complete.

Lines `10-90` are explained in the introduction to the example programs section. They determine which system controller is being used, set flags, and prepare the instrument for remote operation.

```
1     !Filename:  SRQ
2     !
3     ! Description:
4     !   Set an SRQ to occur when averaging is complete.
5     !   Turn on averaging, and set to 8 averages.
6     !   Initiate sweeps.  SRQ will occur after 8 sweeps.
7     !   Wait in a do-nothing loop, checking SRQ flag.
8     !   Display message after SRQ flag is set.
9     !
10    IF POS(SYSTEM$("SYSTEM ID"),"HP 871") THEN
20      ASSIGN @Hp8711 TO 800
30      Isc=8
40    ELSE
50      ASSIGN @Hp8711 TO 716
60      Isc=7
70      ABORT 7
80      CLEAR 716
90    END IF
91    !
92    ! Clear status registers.
100   OUTPUT @Hp8711;"*CLS"
101   !
102   ! Clear the Service Request Enable register.
```

```
110   OUTPUT @Hp8711;"*SRE 0"
111   !
112   ! Clear the Standard Event Status Enable register.
120   OUTPUT @Hp8711;"*ESE 0"
121   !
122   ! Preset the remaining status registers.
130   OUTPUT @Hp8711;"STAT:PRES"
131   !
132   ! Set operation status register to report
133   ! to the status byte on POSITIVE transition of
134   ! the averaging bit.
140   OUTPUT @Hp8711;"STAT:OPER:ENAB 256"
141   !
142   ! Set averaging status register to report to
143   ! operational status register on NEGATIVE transition
144   ! of the averaging done bits.  The NEGATIVE
145   ! transition needs to be detected because the
146   ! averaging bit 0 is set to 1 while the analyzer
147   ! is sweeping on channel 1 and the number of
148   ! sweeps completed since averaging restart is
149   ! less than the averaging factor.  When the bit
150   ! goes back to 0, the averaging is done.
152   OUTPUT @Hp8711;"STAT:OPER:AVER:ENAB 1"
153   !
154   ! Enable the operational status bit in the status
155   ! byte to generate an SRQ.
160   OUTPUT @Hp8711;"*SRE 128"
161   !
162   ! On an interrupt from HP-IB "Isc" (Interface
163   ! Select Code) SRQ bit (2), branch to the interrupt
164   ! service routine "Srq_handler".
170   ON INTR Isc,2 GOSUB Srq_handler
171   !
172   ! Now enable the interrupt on SRQ (Service Request).
180   ENABLE INTR Isc;2
181   !
182   ! Set averaging factor to 8.
190   OUTPUT @Hp8711;"SENS1:AVER:COUN 8;*WAI"
191   !
192   ! Turn on averaging and restart.
200   OUTPUT @Hp8711;"SENS1:AVER ON;AVER:CLE;*WAI"
```

```
201   !
202   ! Turn on continuous sweep trigger mode.
210   OUTPUT @Hp8711;"ABOR;:INIT1:CONT ON;*WAI"
211   !
212   ! Initialize flag indicating when averaging done
213   ! to 0.  Then loop continuously until the
214   ! interrupt is detected, and the interrupt
216   ! service routine acknowledges the
217   ! interrupt and sets the flag to 1.
220   Avg_done=0
225   DISP "Waiting for SRQ on averaging complete.";
230   LOOP
240     DISP ".";
245     WAIT 0.1   ! Slow down dots
250   EXIT IF Avg_done=1
260   END LOOP
261   !
262   ! Display desired completion message.
265   DISP
270   DISP "Got SRQ.  Averaging Complete!"
280   STOP
290   !
300 Srq_handler: ! Interrupt Service Routine
301   !
302   ! Determine that the analyzer was actually
303   ! the instrument that generated the
304   ! interrupt.
310   Stb=SPOLL(@Hp8711)
311   !
312   ! Determine if the operation status register
313   ! caused the interrupt by looking at bit 7
314   ! of the result of the serial poll.
320   IF BINAND(Stb,128)<>0 THEN
321     !
322     ! Read the operational status event register.
330     OUTPUT @Hp8711;"STAT:OPER:EVEN?"
340     ENTER @Hp8711;Op_event
341     !
342     ! Determine if the averaging status register
343     ! bit 8 is set.
350     IF BINAND(Op_event,256)<>0 THEN
```

```
351        !
352        ! If so, then set flag indicating
353        ! averaging done.
360        Avg_done=1
370      END IF
380    END IF
390    RETURN
400    END
```

File Transfer Over HP-IB

Two example programs demonstrate how to transfer files from the analyzer's mass memory to and from mass memory of an external controller via HP-IB. Instrument states and program files may be transferred to or from the analyzers internal non-volatile memory, (`MEM:`), internal-volatile memory, (`RAM:`), and the internal 3.5" floppy disk, (`INT:`).

This can be a convenient method to archive data and programs to a central large mass storage hard drive.

To run these programs, connect an external controller to the analyzer with an HP-IB cable.

GETFILE	Transfers a file from the analyzer to an external controller.
PUTFILE	Transfers a file from an external controller to the analyzer.

GETFILE Example Program

Files are transferred from the analyzer to an external RMB controller. Run this program on your external RMB controller. The program will prompt you to specify which analyzer program to transfer, the mass storage unit (MEM:), internal non-volatile memory, (RAM:), internal volatile memory, or (INT:), internal 3.5” floppy disk drive and the name of the file to be created on your external controller mass storage. Transfers instrument state files or program files.

```
10   !GETFILE
20   !
30   !  This program will get files from 871X specified
        mass storage to a host
40   !  mass storage.  The user specifies the mass storage unit,
        the filename
50   !  of the 871X and the file on the host controller
        to be created.
60   !
110  !
120   DIM Blk$(1:4)[32000]
      ! Max file size = 4 * 32000 = 128000 bytes
130  !
140   DIM Filename$[15],Mass$[15],Dest$[15]
150   INTEGER Word1
160   ASSIGN @Hp8712 TO 716
170   CLEAR @Hp8712
180   BEEP
190   Mass$="INT"
200   Dest$="File871X"
210   INPUT "Enter the name of the 871X file to get.",Filename$
220   INPUT "Enter 871X Mass Storage (mem,INT,ram)",Mass$
260   INPUT "Enter host filename (default='File871X')",Dest$
270   DISP "READING FILE "&Mass$&":"&Filename$&" ..."
280   OUTPUT @Hp8712;"MMEM:TRANSFER? '"&Mass$&":"&Filename$&"'"
290   ENTER @Hp8712 USING "#,W";Word1
300   ENTER @Hp8712 USING "%,-K";Blk$(*)
310   FOR I=1 TO 4
320     Filelength=LEN(Blk$(I))+Filelength
```

```
330   NEXT I
340   BEEP
350   PRINT "Length",Filelength
360   DISP "Creating new file..."
370   ON ERROR GOTO Save_file
380   PURGE Dest$
390 Save_file:     !
400   OFF ERROR
410   CREATE Dest$,Filelength
420   ASSIGN @File TO Dest$;FORMAT ON
430   OUTPUT @File;Blk$(*);
440   ASSIGN @File TO *
450   DISP "File "&Dest$&" created."
460   BEEP
470   END
```

PUTFILE Example Program

PUTFILE - Files are transferred from the RMB mass storage to the analyzer. Run this program on your external RMB controller. The program will prompt you to specify the file to transfer and where to transfer the file. BDATA or ASCII files may be transfered to the analyzer's internal non-volatile memory, (`MEM:`), the internal volatile memory, (`RAM:`), or the internal built in 3.5" floppy disk, (`INT:`).

```
10   !  PUTFILE
20   !
30   !  This program will transfer files from RMB mass mem to the
specified
40   !  871X mass storage.  The user specifies the 871X mass
storage unit,
50   !  the 871X file to be created, file type, and file to be
transferred.
60   !
110  !
120   DIM A$(1:4)[32000]
130   DIM Filename$[15],Mass$[15],Source$[15]
140   INTEGER Word1
150   Bdat$="n"
160   ASSIGN @Hp8712 TO 716
170   CLEAR @Hp8712
180   BEEP
190   Mass$="INT"
200   INPUT "Enter the name of the 871X file to create",Filename$
210   INPUT "File type BDAT? (y,n) [n]",Bdat$
220   INPUT "Enter the 871X Mass Storage (mem,INT,ram)",Mass$
260   INPUT "Enter source filename",Source$
270   DISP "READING FILE "&Source$&" ..."
280   ASSIGN @File TO Source$;FORMAT OFF
290   ENTER @File USING "%,-K";A$(*)
300   ASSIGN @File TO *
310  !PRINT A$
320   BEEP
330   Length=0
340   FOR I=1 TO 4
```

```
350     Length=LEN(A$(I))+Length
360   NEXT I
370   DISP "TRANSFERRING FILE = ",Length
380   IF Bdat$="y" OR Bdat$="Y" THEN
390     IF Length<10 THEN
400       Blk$="1"&VAL$(Length)
410     ELSE
420       IF Length<100 THEN
430         Blk$="2"&VAL$(Length)
440       ELSE
450         IF Length<1000 THEN
460           Blk$="3"&VAL$(Length)
470         ELSE
480           IF Length<10000 THEN
490             Blk$="4"&VAL$(Length)
500           ELSE
510             IF Length<100000 THEN
520               Blk$="5"&VAL$(Length)
530             ELSE
540               Blk$="6"&VAL$(Length)
550             END IF
560           END IF
570         END IF
580       END IF
590     END IF
600     OUTPUT @Hp8712;"MMEM:TRANSFER:BDAT '"&Mass$&"
        :"&Filename$&"',#"&Blk$;
610   ELSE
620     OUTPUT @Hp8712;"MMEM:TRANSFER
'"&Mass$&":"&Filename$&"',#0";
630   END IF
640   OUTPUT @Hp8712;A$(*);END
650   DISP "871X file "&Mass$&":"&Filename$&" created."
660   BEEP
670   END
```

Customized Display

GRAPHICS	Using graphics and softkeys to create customized procedures. The example demonstrates the use of some of the user graphics commands including the one to erase a previously drawn line. It also demonstrates use of the softkeys and detecting a front panel keypress with the service request interrupt process.

GRAPHICS Example Program

This program demonstrates how to use the analyzer's user graphics commands to draw setup diagrams. It also demonstrates generating a service request in response to a keyboard interrupt. More information on user graphics commands is available in Chapter 7, "Using Graphics," and in Chapter 12, "SCPI Command Summary". Information on generating a service request and using the status reporting structure is in Chapter 5, "Using the Status Registers."

Note that this program uses the analyzer's user graphics commands. If the IBASIC option is installed, graphics may sometimes be more easily implemented using BASIC commands such as POLYGON and RECTANGLE. For further information, see the "BARCODE" program description in the *HP Instrument BASIC Users Handbook*.

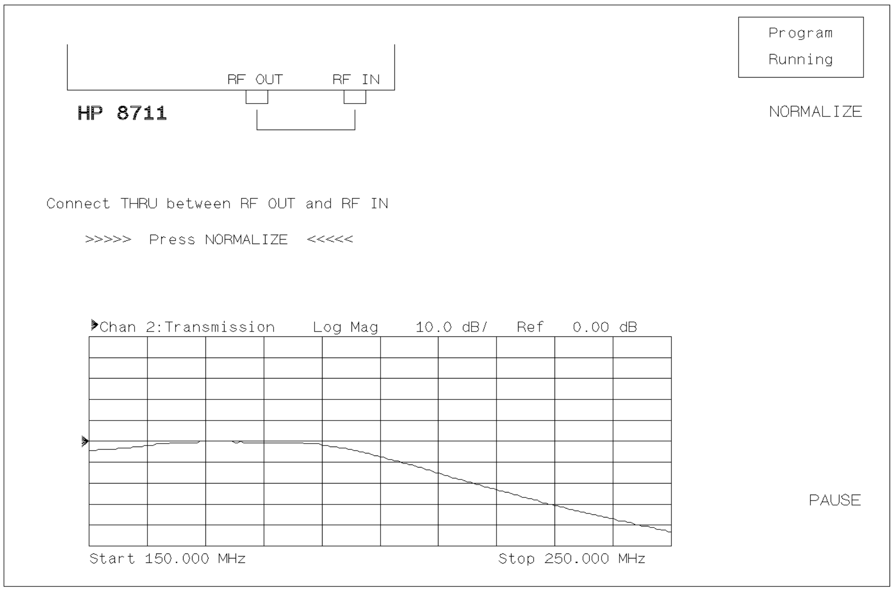
Lines `10-110` are explained in the introduction to the example programs section. They determine which system controller is being used, set flags, and prepare the instrument for remote operation.

Lines `170-240` draw and label a representation of an HP 8711 for a connection diagram. This example is a simple front view from the top.

Lines `250-450` draw the connection needed for a normalization. The operator is prompted to make this connection and to press a softkey on the instrument. A flashing message is used to attract attention.

NOTE

This program works properly *only* when option 1C2, IBASIC, has been installed. Refer to program GRAPH2 if your analyzer does *not* have the IBASIC option installed.



GRAPHICS example connection diagram

Lines 460-580 perform the normalization, erase the prompts (without erasing the whole screen) and prepare for the test.

Lines 590-730 are a branching routine that handles the service request generated interrupts used by the external controller.

```
1      ! Filename:  GRAPHICS
2      !
3      ! Description: Draws a simple connection diagram
4      ! in the IBASIC window, and displays a softkey.
5      !
6      ! NOTE:  This program works properly ONLY
7      ! when option 1C2, IBASIC, has been installed.
8      ! Refer to program GRAPH2 if no IBASIC option.
9      !
```

```
10     IF POS(SYSTEM$("SYSTEM ID"),"HP 871") THEN
20       ASSIGN @Hp8711 TO 800
30       Internal=1
40       Isc=8
50     ELSE
60       ASSIGN @Hp8711 TO 716
70       Internal=0
80       Isc=7
90       ABORT 7
100      CLEAR 716
110    END IF
111    !
112    ! Allocate an IBASIC display partition
113    ! to show the graphics.
120    OUTPUT @Hp8711;"DISP:PROG UPP"
121    !
122    ! Clear the IBASIC display partition.
130    OUTPUT @Hp8711;"DISP:WIND10:GRAP:CLE"
131    !
132    ! Turn on channel 2 for measurements.  The
133    ! lower part of the display is
134    ! devoted to display of measurements.
140    OUTPUT @Hp8711;"SENS2:STAT ON;*WAI"
141    !
142    ! Take a single controlled sweep to ensure
143    ! a valid measurement using *OPC query.
150    OUTPUT @Hp8711;"ABOR;:INIT2:CONT OFF;:INIT2;*OPC?"
160    ENTER @Hp8711;Opc
161    !
162    ! Select the bright "pen" and bold font.
170    OUTPUT @Hp8711;"DISP:WIND10:GRAP:COL 1;LAB:FONT BOLD"
171    !
172    ! Draw a label reading "HP 8711B" at 45 pixels
173    ! to the right and 120 pixels above the origin.
174    ! The origin is the lower left corner of the
175    ! current graphics window (upper half).
180    OUTPUT @Hp8711;"DISP:WIND10:GRAP:MOVE 45,120
       ;LAB 'HP 8711B'"
181    !
182    ! Draw a box to represent the analyzer.
190    OUTPUT @Hp8711;"DISP:WIND10:GRAP:MOVE 30,175
```

```
     ;DRAW 30,140;DRAW 480,140;DRAW 480,175"
191  !
192  ! Draw a box to represent the REFLECTION RF OUT port.
200  OUTPUT @Hp8711;"DISP:WIND10:GRAP:MOVE 275,140
     ;DRAW 275,130;DRAW 305,130;DRAW 305,140"
201  ! Draw a box to represent the TRANSMISSION RF IN port.
210  OUTPUT @Hp8711;"DISP:WIND10:GRAP:MOVE 410,140
     ;DRAW 410,130;DRAW 440,130;DRAW 440,140"
211  ! Change the text font to small, which is the
212  ! same as that used for PRINT or DISP statements.
220  OUTPUT @Hp8711;"DISP:WIND10:GRAP:LAB:FONT SMAL"
221  !
222  ! Label the RF OUT port.
230  OUTPUT @Hp8711;"DISP:WIND10:GRAP:MOVE 250,145
     ;LAB 'RF OUT'"
231  !
232  ! Label the RF IN port.
240  OUTPUT @Hp8711;"DISP:WIND10:GRAP:MOVE 395,145
     ;LAB 'RF IN'"
241  !
250 Normalize: !
251  !
252  ! Draw a through connection between the RF OUT
253  ! and RF IN ports.
260  OUTPUT @Hp8711;"DISP:WIND10:GRAP:MOVE 290,125
     ;DRAW 290,110;DRAW 425,110;DRAW 425,125"
261  ! Prompt the operator to connect the through.
270  OUTPUT @Hp8711;"DISP:WIND10:GRAP:MOVE 1,50
     ;LAB 'Connect THRU between RF OUT and RF IN'"
280  IF Internal=1 THEN
281    ! If using the IBASIC (internal) controller,
282    ! then use the "ON KEY" method to handle
283    ! user interface.
290    ON KEY 1 LABEL "NORMALIZE" RECOVER Norm
300  ELSE
301    ! If using an external controller...
302    !
303    ! Initialize flag for checking on keyboard
304    ! interrupts.
310    Keycode=-1
311    !
```

```
312      ! Label softkey 1.
320      OUTPUT @Hp8711;"DISP:MENU:KEY1 'NORMALIZE'"
321      !
322      ! Clear the status register and event status
323      ! register.
330      OUTPUT @Hp8711;"*CLS;*ESE 0"
331      !
332      ! Preset the other status registers.
333      ! Enable the Device Status register to report
334      ! to the Status Byte on positive transition
335      ! of bit 0 (key press).  Enable the Status
336      ! Byte to generate an interrupt when the
337      ! Device Status register's summary bit
338      ! changes.
340      OUTPUT @Hp8711;"STAT:PRES;DEV:ENAB 1;*SRE 4"
341      !
342      ! Clear the key queue to ensure that previous
343      ! key presses  do not generate an interrupt.
350      OUTPUT @Hp8711;"SYST:KEY:QUE:CLE"
351      !
352      ! Set up and enable the interrupt on the HP-IB
353      ! when a service request is received.
360      ON INTR Isc,5 RECOVER Srq
370      ENABLE INTR Isc;2
380    END IF
381    !
382    ! Turn off the graphics buffer.
390    OUTPUT @Hp8711;"DISP:WIND10:GRAP:BUFF OFF"
391    !
392    ! Loop for waiting for press of the NORMALIZE key.
393    ! The two different output statements along with
394    ! the wait statements create a blinking effect.
395    ! There is not exit from this loop other than
396    ! a keyboard interrupt.
400    LOOP
410      OUTPUT @Hp8711;"DISP:WIND10:GRAP:MOVE 55,18
         ;LAB '>>>>>  Press NORMALIZE  <<<<<'"
420      WAIT .2
430      OUTPUT @Hp8711;"DISP:WIND10:GRAP:MOVE 55,18
         ;LAB '       Press NORMALIZE       '"
440      WAIT .2
```

```
450   END LOOP
451   !
460 Norm: ! Entry point to wait for a key press.
461   !
462   ! If wrong key pressed, return to Normalize.
470   IF Keycode<>0 THEN GOTO Normalize
480   OFF KEY
481   !
482   ! The through should now be connected and
483   ! ready to measure.
484   !
485   ! Turn the graphics buffer back on.
490   OUTPUT @Hp8711;"DISP:WIND10:GRAP:BUFF ON"
491   !
492   ! Select the "erase" pen (pen color 0) and
493   ! erase the prompts.
500   OUTPUT @Hp8711;"DISP:WIND10:GRAP:COL 0;MOVE 55,18
      ;LAB '>>>>>  Press NORMALIZE  <<<<<'"
510   OUTPUT @Hp8711;"DISP:WIND10:GRAP:MOVE 1,50
      ;LAB 'Connect THRU between RF OUT and RF IN'"
520   OUTPUT @Hp8711;"DISP:MENU:KEY1 '        '"
521   !
522   ! Display the active data trace only.  Turn off
523   ! any previous normalization.
530   OUTPUT @Hp8711;"CALC2:MATH (IMPL)"
531   !
532   ! Take a single sweep on channel 2.
540   OUTPUT @Hp8711;"INIT2;*WAI"
541   !
542   ! Copy the new data trace into the memory array.
550   OUTPUT @Hp8711;"TRAC CH2SMEM,CH2SDATA"
551   !
552   ! Normalize; that is, display the active data
553   ! relative to the memory trace.
560   OUTPUT @Hp8711;"CALC2:MATH (IMPL/CH2SMEM)"
561   !
562   ! Display only one of the traces (the normalized
563   ! trace).
570   OUTPUT @Hp8711;"DISP:WIND2:TRAC1 ON;TRAC2 OFF"
571   !
572   ! Erase the through connect and select pen color 1 again.
```

```
580   OUTPUT @Hp8711;"DISP:WIND10:GRAP:MOVE 290,110
      ;DRAW 425,110;DRAW 425,125;COL 1"
590   STOP
600   !
610 Srq: ! This is the branching routine that handles
           service request
611        ! generated interrupts.
612   !
613   ! Do a serial poll to find out if analyzer generated the
614   ! interrupt.
620   Stb=SPOLL(@Hp8711)
621   !
622   ! Determine if the Device Status register's summary
623   ! bit (bit 2 of the Status Byte) has been set.
630   IF BINAND(Stb,4)<>0 THEN
631     !
632     ! If so, then get the Device Status Register contents.
640     OUTPUT @Hp8711;"STAT:DEV:EVEN?"
650     ENTER @Hp8711;Dev_event
651     !
652     ! Check for key press...
660     IF BINAND(Dev_event,1)<>0 THEN
661       ! If so, then determine which key.
670       OUTPUT @Hp8711;"SYST:KEY?"
680       ENTER @Hp8711;Keycode
690     END IF
700   END IF
701   !
702   ! Reenable the interrupt in case wrong key
703   ! was pressed.
710   ENABLE INTR Isc
720   GOTO Norm
730   END
```

9

Front Panel Keycodes

Front Panel Keycodes

Your program can monitor the analyzer's front panel and determine when a key has been pressed or when the knob (RPG — rotary pulse generator) has been turned. Key presses from an attached PC DIN keyboard can also be captured.

When keys are pressed or when the knob is turned, the analyzer detects this event, sets bit 0 of the Device Status Register (see Chapter 5, "Using Status Registers") and stores the associated information in a key queue. Your program can use the SCPI `SYSTem:KEY` commands to read the contents of the key queue.

The SCPI query `SYSTem:KEY:TYPE?` returns a string indicating the type of key press event:

Return Value	Meaning
NONE	No key has been pressed
KEY	A front panel key has been pressed
RPG	The analyzer's knob has been turned
ASC	A key on the ASCII PC DIN keyboard has been pressed

The SCPI query `SYSTem:KEY[:VALue]?` returns a number describing the type of key press. The meaning of the number depends on the key type returned by the `SYSTem:KEY:TYPE?` query:

`SYST:KEY:TYPE`	`SYST:KEY:VALUE` **Meaning**
NONE	No meaning. Returns -1.
KEY	A number from 0 to 56 representing the "key code" of the front panel key. See following table for list.
RPG	The number of knob "ticks". Positive values indicate a clock-wise turn; negative numbers indicate counter-clockwise. Larger numbers indicate the knob has been turned faster or further.
ASC	The ASCII value of the pressed key.

The `SYSTem:KEY[:VALue]?` query removes the key from the key queue, so that you can read the next key. For this reason, you must perform the `SYSTem:KEY:TYPE?` query before performing the `SYSTem:KEY[:VALue]?`.

The queue that stores the key press events has a finite length. In firmware revision B.03.00, this length is 32. This means that after 32 key presses occur without being read (using `SYSTem:KEY[:VALue]?`), subsequent key presses or knob ticks will be ignored.

Your program can query the queue length using the SCPI command:

```
SYSTem:KEY:QUEue:MAXimum?
```

You can clear the queue using:

```
SYSTem:KEY:QUEue:CLEar
```

You can check how many key presses or knob tick events have occurred using

```
SYSTem:KEY:QUEUE:COUNt?
```

Finally, you can turn the key queue on or off using

```
SYSTem:KEY:QUEUE[:STATe] <ON|OFF>
```

When the queue is turned off, your program must read each key before a following key is pressed, or information will be lost. It is generally best to leave the queue enabled.

For a complete example of how to read the front panel keys and knob, refer to the KEYCODE example program.

Front Panel Keycodes

Key Group	Key Label	Key Code
Softkeys	Softkey 1 (top key)	0
	Softkey 2	1
	Softkey 3	2
	Softkey 4	3
	Softkey 5	4
	Softkey 6	5
	Softkey 7	6
	Softkey 8 (bottom key)	7
Numeric Keys	(0) (zero)	10
	(1) (one)	11
	(2) (two)	12
	(3) (three)	13
	(4) (four)	14
	(5) (five)	15
	(6) (six)	16
	(7) (seven)	17
	(8) (eight)	18
	(9) (nine)	19
	(ENTER)	20
	(.) (decimal)	21

Key Group	Key Label	Key Code
	(— / ←) (minus/backspace)	22
	(⇑) (step up)	23
	(⇓) (step down)	24
Feature Keys	(BEGIN)	40
	(CHAN 1)	41
	(CHAN 2)	42
	(POWER)	43
	(MENU)	44
	(FREQ)	45
	(SWEEP)	46
	(CAL)	47
	(DISPLAY)	48
	(SCALE)	49
	(AVG)	50
	(FORMAT)	51
	(MARKER)	52
	(SAVE/RECALL)	53
	(SYSTEM/OPTIONS)	54
	(HARD/COPY)	55
	(PRESET)	56

10

Introduction to SCPI

Introduction to SCPI

This chapter is a guide to HP-IB control of the analyzer. Its purpose is to provide concise information about the operation of the analyzer under HP-IB control. The reader should already be familiar with making measurements with the analyzer and with the general operation of HP-IB.

Standard Commands for Programmable Instruments (SCPI) is a programming language designed specifically for controlling instruments by Hewlett-Packard and other industry leaders. SCPI provides commands that are common from one instrument to another. This elimination of "device specific" commands for common functions allows programs to be used on different instruments with very little modification.

SCPI was developed to conform to the IEEE 488.2 standard (replacing IEEE 728-1982). The IEEE 488.2 standard defines the syntax and data formats used to send data between devices, the structure of status registers, and the commands used for common tasks. For more information, refer to the IEEE standard itself. SCPI defines the commands used to control device-specific functions, the parameters accepted by these functions, and the values they return.

The Command Tree

The SCPI standard organizes related instrument functions by grouping them together on a common branch of a command tree. Each branch is assigned a mnemonic to indicate the nature of the related functions. The analyzer has 16 major SCPI branches or **subsystems**. See Figure 10-1 for a model of how these subsystems are organized to manage the measurement and data flow for the analyzer.

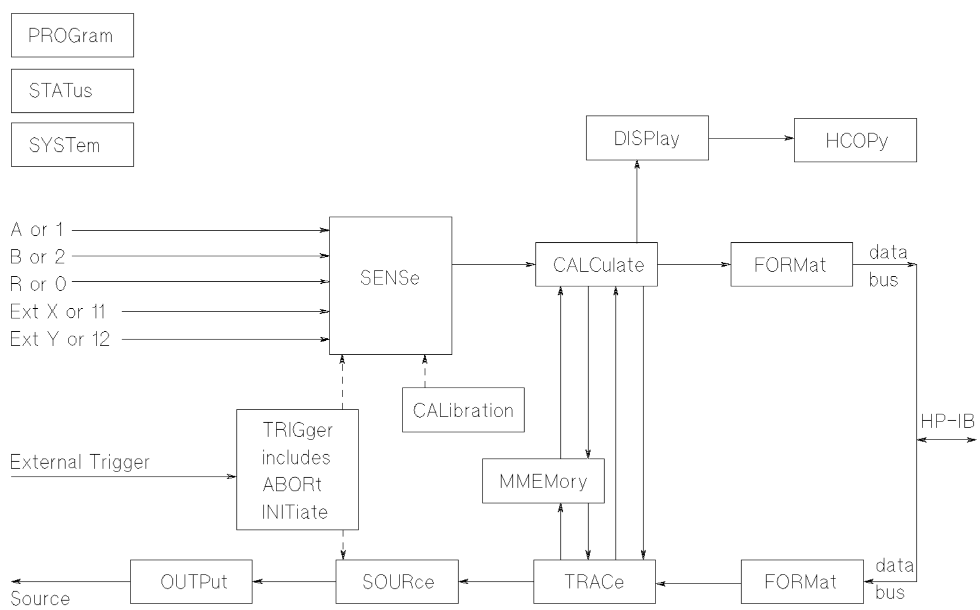


Figure 10-1. Measurement and Data Flow of the Analyzer

The analyzer's major SCPI subsystems and their functions are described below.

`ABORt`	Aborts any sweep in progress.
`CALCulate`	Configures post-measurement processing of the measured data (such as marker and limit testing functions).
`CALibration`	Controls zeroing the broadband diode detectors.
`DISPlay`	Controls the display of measurement data, annotation and user graphics.
`FORMat`	Controls the format of data transfers over the HP-IB. (For more information about HP-IB data transfer refer to Chapter 4, "Data Types and Encoding.")
`HCOPy`	Controls hardcopy (printer and plotter) output.
`INITiate`	Controls the triggering of sweeps.
`MMEMory`	Controls mass storage of instrument states and data (disk and internal memory interface functions).
`OUTPut`	Turns on/off the source output power (power to the device under test).
`PROGram`	Interfaces IBASIC programs and commands with an external controller. (For more information on IBASIC programming refer to *HP Instrument BASIC User's Handbook*.)
`SENSe`	Configures parameters (such as the frequency and measurement parameters) related to the sweep and the measured signal (from the device under test). This subsystem also controls the narrowband calibration routines.
`SOURce`	Controls the RF output power level of the source (power to the device under test).

`STATus`	Contains the commands for using the SCPI status registers. (For more information about using the status registers refer to Chapter 5, "Using Status Registers.")
`SYSTem`	Contains miscellaneous system configuration commands (such as I/O port, clock and softkey control).
`TRACe`	Interfaces with the internal data arrays (functions such as data transfer and trace memory).
`TRIGger`	Controls the source of the sweep triggering.

When many functions are grouped together on a particular branch, additional branching is used to organize these functions into groups that are even more closely related. The branching process continues until each analyzer function is assigned to its own branch. For example, the function that turns on and off the marker tracking feature is assigned to the `TRACKING` branch of the `FUNCTION` branch of the `MARKER` branch of the `CALCULATE` subsystem. The command looks like this:

```
CALCULATE:MARKER:FUNCTION:TRACKING ON
```

NOTE

Colons are used to indicate branching points on the command tree. A parameter is separated from the rest of the command by a space.

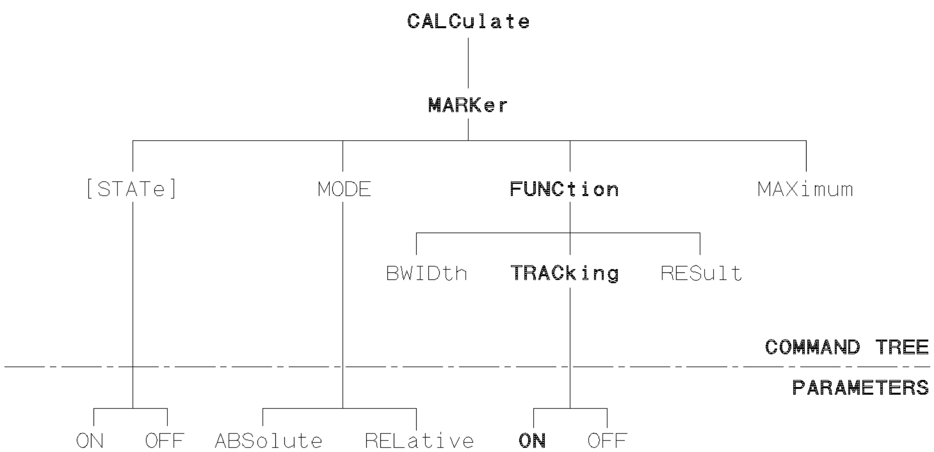


Figure 10-2. Partial Diagram of the CALCulate Subsystem Command Tree

Sending Multiple Commands

Multiple commands can be sent within a single program message by separating the commands with semicolons. For example, the following program message — sent within an HP BASIC `OUTPUT` statement — turns on the marker reference and moves the main marker to the highest peak on the trace:

```
OUTPUT 716;"CALCULATE:MARKER:MODE
RELATIVE;:CALCULATE:MARKER:MAXIMUM"
```

One of the analyzer's command parser main functions is to keep track of a program message's position in the command tree. This allows the previous program message to be simplified. Taking advantage of this parser function, the simpler equivalent program message is:

```
OUTPUT 716;"CALCULATE:MARKER:MODE RELATIVE;MAXIMUM"
```

In the first version of the program message, the semicolon that separates the two commands is followed by a colon. Whenever this occurs, the command parser is reset to the base of the command tree. As a result, the next command is only valid if it includes the entire mnemonic path from the base of the tree.

In the second version of the program message, the semicolon that separates the two commands is not followed by a colon. Whenever this occurs, the command parser assumes that the mnemonics of the second command arise from the same branch of the tree as the final mnemonic of the preceding command. `MODE`, the final mnemonic of the first command, arises from the `MARKER` branch. So `MAXIMUM`, the first mnemonic of the second command is also assumed to arise from the `MARKER` branch.

The following is a longer series of commands — again sent within HP BASIC `OUTPUT` statements — that can be combined into a single program message:

```
OUTPUT 716;"CALCULATE:MARKER:STATE ON"
OUTPUT 716;"CALCULATE:MARKER:MODE RELATIVE"
OUTPUT 716;"CALCULATE:MARKER:MAXIMUM"
OUTPUT 716;"CALCULATE:MARKER:FUNCTION:TRACKING ON"
```

The single program message is:

```
OUTPUT 716;"CALCULATE:MARKER:STATE ON;MODE
            RELATIVE;MAXIMUM;FUNCTION:TRACKING ON"
```

Command Abbreviation

Each command mnemonic has a long form and a short form. The short forms of the mnemonics allow you to send abbreviated commands. Only the exact short form or the exact long form is accepted.

The short form mnemonics are created according to the following rules:

- If the long form mnemonic has four characters or less, the short form is the same as the long form. For example, `DATA` remains `DATA`.
- If the long form mnemonic has more than four characters and the fourth character is a consonant, the short form consists of the first four characters of the long form. For example, `CALCULATE` becomes `CALC`.
- If the long form mnemonic has more than four characters and the fourth character is a vowel, the short form consists of the first three characters of the long form. For example, `LIMIT` becomes `LIM`.

NOTE

The short form of a particular mnemonic is indicated by the use of UPPER-CASE characters in this manual.

SCPI is not case sensitive so any mix of upper- and lower-case lettering can be used when sending commands to the analyzer.

If the rules listed in this section are applied to the last program message in the preceding section, the statement:

```
OUTPUT 716;"CALCULATE:MARKER:STATE ON;MODE
            RELATIVE;MAXIMUM;FUNCTION:TRACKING ON"
```

becomes:

```
OUTPUT 716;"CALC:MARK:STAT ON;MODE REL;MAX;FUNC:TRAC ON"
```

Implied Mnemonics

Some mnemonics can be omitted from HP-IB commands without changing the effect of the command. These special mnemonics are called implied mnemonics, and they are used in many subsystems. In addition to entire mnemonics, variable parts of some mnemonics may also be implied. These are usually a number indicating a particular measurement channel, marker, or similar choice.

> **NOTE**
>
> When a number is not supplied for an implied variable, a default choice is assumed; this choice is always `1`.

The `INITIATE` subsystem contains both the implied mnemonic `IMMEDIATE` at its first branching point and an implied variable for the measurement channel. The command to trigger a new sweep is shown in the "SCPI Command Summary" as:

```
OUTPUT 716;"INITiate[1|2][:IMMediate]
```

Any of the following forms of the command can be sent to the analyzer (using HP BASIC) to trigger a new sweep on channel 1:

```
OUTPUT 716;"INITIATE1:IMMEDIATE"
OUTPUT 716;"INITIATE:IMMEDIATE"
OUTPUT 716;"INITIATE1"
OUTPUT 716;"INITIATE"
```

If the sweep is to be triggered for measurement channel 2, the channel number *must* be specified:

```
OUTPUT 716;"INITIATE2:IMMEDIATE"
OUTPUT 716;"INITIATE2"
```

Parameter Types

Parameters are used in many commands. The analyzer uses several types of parameters with different types of commands and queries. When a parameter is sent with a SCPI command it must be separated from the command by a space. If more than one parameter is sent they are separated from each other by commas.

Numeric Parameters

Most subsystems use numeric parameters to specify physical quantities. Simple numeric parameters accept all commonly used decimal representations of numbers, including optional signs, decimal points, and scientific notation. If an instrument setting programmed with a numeric parameter can only assume a finite number of values, the instrument automatically rounds the parameter. In addition to numeric values, all numeric parameters accept `MAXimum` and `MINimum` as values (note that `MAXimum` and `MINimum` can be used to set or query values).

`<num>` is used in this document to denote a numeric parameter.

An example is the command to set the stop frequency for a measurement. The first command below sets the stop frequency to a specific value. The second command below sets the stop frequency to its maximum possible value (1300 MHz for HP 8711B/12B or 3000 MHz for HP 8713B/14B).

```
OUTPUT 716;"SENSE1:FREQUENCY:STOP 1300 MHZ"

OUTPUT 716;"SENSE1:FREQUENCY:STOP MAX"
```

Query Response

When a numeric parameter is queried the number is returned in one of the three numeric formats.

NR1 Integers (such as +1, 0, -1, 123, -12345)

NR2 Floating point number with an explicit decimal point (such as 12.3, +1.234, -0.12345)

NR3 Floating point number in scientific notation (such as +1.23E+5, +123.4E-3, -456.789E+6)

An example is the response to a query of the stop frequency after executing the above commands (this response is of the NR3 type).

```
OUTPUT 716;"SENSE1:FREQUENCY:STOP?"
```

returns the value `1.3E+9`.

Character Parameters

Character parameters (sometimes referred to as discrete parameters) consist of ASCII characters. They are typically used for program settings that have a finite number of values.

These parameters use mnemonics to represent each valid setting. They have a long and a short form which follow the same rules as command mnemonics.

`<char>` is used in this document to denote a character parameter.

An example of a command using a character parameter is the command that selects the format in which the measurement data is displayed:

```
OUTPUT 716;"CALCULATE1:FORMAT MLOGARITHMIC"
```

Query Response

When a character parameter is queried the response is always the short form of the mnemonic that represents the current setting. An example is the response to a query of the data format after executing the above command.

```
OUTPUT 716;"CALCULATE1:FORMAT?"
```

returns the value `MLOG`.

Boolean Parameters

Boolean parameters are used for program settings that can be represented by a single binary condition. Commands that use this type of parameter accept the values `ON` (or `1`) and `OFF` (or `0`).

`<ON|OFF>` is used in this document to denote a boolean parameter.

An example of a command that uses a boolean parameter is the command that makes the analyzer continuously trigger (or stop triggering) measurements.

```
OUTPUT 716;"INITIATE:CONTINUOUS ON"
```

A special group of commands uses boolean parameters to control automatic functions of the instrument, such as automatically selecting the fastest possible sweep speed. With these automatic functions an additional value is available for the parameter. This value `ONCE` causes the function to execute once before turning off.

Query Response

The response when a boolean parameter is queried is a single NR1 number indicating the state `1` for on or `0` for off. An example is the response to a query on the sweep trigger status after executing the above command.

```
OUTPUT 716;"INITIATE:CONTINUOUS?"
```

returns the value `1`.

String Parameters

String parameters can contain virtually any set of ASCII characters. The string must begin with a single quote (') or a double quote (") and end with the same character (called the delimiter). The delimiter can be included as a character (embedded) inside the string by typing it twice without any characters in between. For example:

```
OUTPUT 716;"DISP:ANN:TITL:DATA 'DUT''S PHASE'"
```

`<string>` is used in this document to denote a string parameter.

A example of a command that uses a string parameter is the `CONFIGURE` command:

```
OUTPUT 716;"CONFIGURE 'FILTER:TRANSMISSION'"
```

Some of the string parameters used by the analyzer, like `'FILTER:TRANSMISSION'` in the example above, follow the same rules that apply to mnemonics. They may have branching (`'FILTER:REFLECTION'` is a related command) and abbreviated versions.

Query Response

The response when a string parameter is queried is a string. The only difference is that the response string will only use double quotes as delimiters. Embedded double quotes may be present in string response data. When the string follows the "SCPI" mnemonic rules, the string returned in response to a query is in the abbreviated form. An example is the response to the configuration status of the analyzer (after executing the last command).

```
OUTPUT 716;"CONFIGURE?"
```

returns the value `"FILT:TRAN"`.

Block Parameters

Block parameters are typically used to transfer large quantities of related data (like a data trace). Blocks can be sent as definite length blocks or indefinite length blocks — the instrument will accept either form. For more information on block data transfers refer to Chapter 4, “Data Types and Encoding.”

`<block>` is used in this document to denote a block parameter.

Syntax Summary

The following conventions are used throughout this manual whenever SCPI mnemonics are being described.

angle brackets (< >) are used to enclose required parameters within a command or query. The definition of the variable is usually explained in the accompanying text.

square brackets ([ ]) are used to enclose implied or optional parameters within a command or query.

UPPERlower case are used to indicate the short form (upper-case) of a given mnemonic. The remaining (lower-case) letters are the rest of the long form mnemonic.

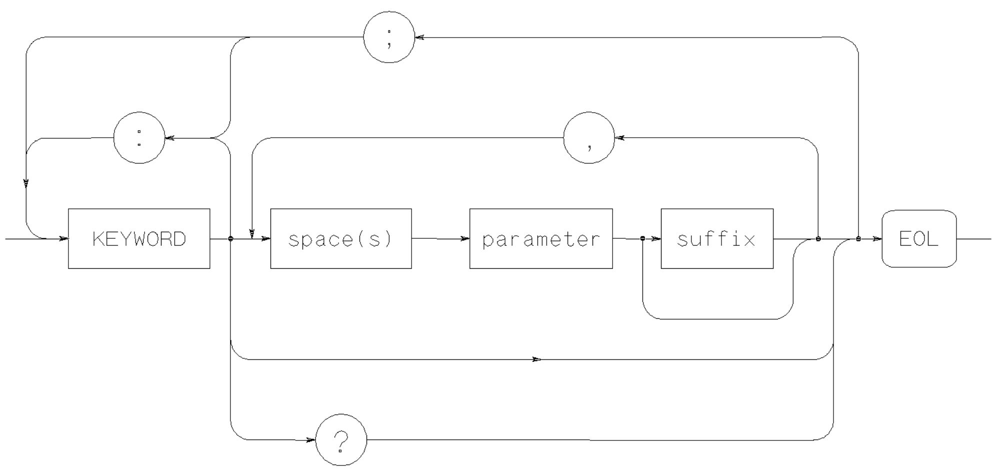


Figure 10-3. SCPI Command Syntax

The following elements have special meanings within a SCPI program message (or combination or mnemonics).

colon (:)	When a command or query contains a series of mnemonics, they are separated by colons. A colon immediately following a mnemonic tells the command parser that the program message is proceeding to the next level of the command tree. A colon immediately following a semicolon tells the command parser that the program message is returning to the base of the command tree.
semicolon (;)	When a program message contains more than one command or query, a semicolon is used to separate them from each other.
comma (,)	A comma separates the data sent with a command or returned with a response.
space ( )	One space is required to separate a command or query from its data (or parameters). Spaces are not allowed inside a command or query.

IEEE 488.2 Common Commands

IEEE 488.2 defines a set of common commands. All instruments are required to implement a subset of these commands, specifically those commands related to status reporting, synchronization and internal operations. The rest of the common commands are optional. The following list details which of these IEEE 488.2 common commands are implemented in the analyzer and the response of the analyzer when the command is received.

*CLS — Clears the instrument Status Byte by emptying the error queue and clearing all event registers, also cancels any preceding *OPC command or query (does not change the enable registers or transition filters).

*ESE <num> — Sets bits in the Standard Event Status Enable Register — current setting is saved in non-volatile memory.

*ESE? — Reads the current state of the Standard Event Status Enable Register.

*ESR? — Reads and clears the current state of the Standard Event Status Register.

*IDN? — Returns a string that uniquely identifies the analyzer. The string is of the form

```
"HEWLETT-PACKARD,8711B,<serial number>,<software revision>"
```

*LRN? — This returns a string of device specific characters that, when sent back to the analyzer will restore the instrument state active when *LRN? was sent. Data formatting (ENTER USING "-K" in HP BASIC) or a similar technique should be used to ensure that the transfer does not terminate on a carriage return or line feed (both $^{C}{}_{R}$ and $^{L}{}_{F}$ are present in the learn string as part of the data).

*OPC — Operation complete command. The analyzer will generate the OPC message in the Standard Event Status Register when all pending overlapped operations have been completed (e.g. a sweep, or a preset). For more information about overlapped operations refer to "Overlapped Commands" in Chapter 2.

***OPC?** Operation complete query. The analyzer will return an ASCII "1" when all pending overlapped operations have been completed.

***OPT?** Returns a string identifying the analyzer's option configuration. The string is of the form `"1E1,1C2"`. The options are identified by the following:

1EC	75 ohm
1E1	60 dB step attenuator
1C2	IBASIC
1DA	AM delay (50 Ω)
1DB	AM delay (75 Ω)

***PCB <num>** Sets the pass-control-back address (the address of the controller before a pass control is executed).

***PSC <num>** Sets the state of the Power-on Status Clear flag — flag is saved in non-volatile memory. This flag determines whether or not the Service Request enable register and the Event Status enable register are cleared at power-up.

***RST** Executes a device reset and cancels any pending `*OPC` command or query. The contents of the instrument's nonvolatile memory are *not* affected by this command.

This command is different from the front panel (PRESET) function in the state of the commands (and their reset states) listed below.

The preset instrument state is described in the *User's Guide*.

```
INITiate:CONTinuous          = OFF
OUTPut[:STATe]               = OFF
CALibration:ZERO:AUTO        = OFF
SENSe:CORRection[:STATe]     = OFF
SENSe:SWEep:POINts           = MAX
SOURce:POWer                 = MIN
```

***SRE <num>** Sets bits in the Service Request Enable Register. Current setting is saved in non-volatile memory.

***SRE?** Reads the current state of the Service Request Enable Register.

`*STB?` Reads the value of the instrument Status Byte. This is a non-destructive read, the Status Byte is cleared by the `*CLS` command.

`*TST?` Returns the result of a complete self-test. An ASCII `0` indicates no failures found. Any other character indicates a specific self-test failure. Does not perform any self-tests. See the *Service Guide* for further information.

`*WAI` Prohibits the instrument from executing any new commands until all pending overlapped commands have been completed.

`*TRG` Triggers a sweep on the active channel when in Trigger Hold mode. Ignored if in continuous sweep.

11

Menu Map with SCPI Commands

Menu Map with SCPI Commands

This chapter contains a map of all the softkey menu choices in the analyzer. There is a table for each major hardkey on the analyzer's front panel. The softkeys are shown with corresponding SCPI commands (if one exists). Hardkeys are indicated with the (Hardkey) notation, softkeys are shown as Softkeys. SCPI commands are all shown in their short form.

Some commands (such as source settings) have mnemonics that specify the channel in use. These mnemonics are shown as SENS[1|2]: . . . indicating that either channel could be used. The actual mnemonic entered would be SENS1: . . . for setting channel 1 or SENS2: . . . for channel 2. Mnemonics for keys that toggle between two states are shown as . . . ON|OFF.

<num> and <string> refer to parameter types described in the "Parameter Types" section. <string> parameters are typically enclosed in single quotes ('the string data').

(PRESET) SCPI Command

KEYSTROKES	SCPI COMMAND
(PRESET)	SYST:PRES;*WAI

(BEGIN) SCPI Commands

KEYSTROKES	SCPI COMMAND
(BEGIN)	
Amplifier	
Transmissn	CONF 'AMPL:TRAN';*WAI
Reflection	CONF 'AMPL:REFL';*WAI
Power	CONF 'AMPL:POW';*WAI
Filter	
Transmissn	CONF 'FILT:TRAN';*WAI
Reflection	CONF 'FILT:REFL';*WAI
Broadband Passive	
Transmissn	CONF 'BBAN:TRAN';*WAI
Reflection	CONF 'BBAN:REFL';*WAI
Mixer	
Conversion Loss	CONF 'MIX:CLOS';*WAI
Reflection	CONF 'MIX:REFL';*WAI
AM Delay [1]	CONF 'MIX:GDEL';*WAI

1 Options 1DA and 1DB only

BEGIN SCPI Commands (continued)

KEYSTROKES	SCPI COMMAND
Cable [1]	
Transmissn	CONF[1|2] 'CABL:TRAN';*WAI
Reflection	CONF[1|2] 'CABL:REFL';*WAI
Fault Location	CONF[1|2] 'CABL:FAULT';*WAI
Start Distance	SENS[1|2]:DIST:STAR <num> [FEET|MET];*WAI
Stop Distance	SENS[1|2]:DIST:STOP <num> [FEET|MET];*WAI
Feet	SENS:DIST:UNIT FEET
Meters	SENS:DIST:UNIT MET
Low Pass	SENS:FREQ:MODE LOWP;*WAI
Band Pass	SENS:FREQ:MODE CENT;*WAI
Center Frequency	DISP:ANN:FREQ[1|2]:MODE CSPAN
(Number) Units	SENS[1|2]:FREQ:CENT <num> [MHZ|KHZ|HZ]; *WAI
SRL	CONF[1|2] 'CABL:SRL';*WAI
Start Freq	DISP:ANN:FREQ[1|2]:MODE SSTOP
(Number) Units	SENS[1|2]:FREQ:STAR <num> [MHZ|KHZ|HZ]; *WAI
Stop Freq	DISP:ANN:FREQ[1|2]:MODE SSTOP
(Number) Units	SENS[1|2]:FREQ:STOP <num> [MHZ|KHZ|HZ]; *WAI
Connector Model	
Measure Connector	SENS[1|2]:CORR:MODEL:CONN
Connector Length	
(Number) (ENTER)	SENS[1|2]:CORR:LENG:CONN <num>
Connector C (Number) (ENTER)	SENS[1|2]:CORR:CAP:CONN <num>

1 Option 100 only

(BEGIN) SCPI Commands (continued)

KEYSTROKES	SCPI COMMAND
Z Cutoff Frequency	
(Number) (ENTER)	SENS:FREQ:ZST <num> [MHZ|KHZ|HZ]
Auto Z ON off	SENS[1|2]:FUNC:SRL:MODE <AUTO|MANUAL>
Manual Z (Number) (ENTER)	SENS[1|2]:FUNC:SRL:IMP <num>
SRL Cable Scan	SENS[1|2]:FUNC:SRL:SCAN;*WAI
Number of Points (Number) (ENTER)	SENS[1|2]:SWE:POIN <num>;*WAI
User BEGIN	No SCPI command

CHAN 1 | CHAN 2 **SCPI Commands**

KEYSTROKES	SCPI COMMAND
CHAN 1 | CHAN 2	SENS[1|2]:STAT ON;*WAI
Transmissn	SENS[1|2]:FUNC 'XFR:POW:RAT 2,0'; DET NBAN;*WAI
Reflection	SENS[1|2]:FUNC 'XFR:POW:RAT 1,0'; DET NBAN;*WAI
Fault Location [1]	SENS[1|2]:FUNC 'FLOC 1,0'; DET NBAN;*WAI
SRL [1]	SENS[1|2]:FUNC 'SRL 1,0'; DET NBAN;*WAI
More	
Power	SENS[1|2]:FUNC 'XFR:POW 2';DET BBAN;*WAI
Conversion Loss	SENS[1|2]:FUNC 'XFR:POW:RAT 2,0'; DET BBAN;*WAI
AM Delay [2]	SENS[1|2]:FUNC 'XFR:GDEL:RAT 12,11'; DET BBAN;*WAI
Detection Options	
Narrowband Internal	
A	SENS[1|2]:FUNC 'XFR:POW 1';DET NBAN;*WAI
B	SENS[1|2]:FUNC 'XFR:POW 2';DET NBAN;*WAI
R	SENS[1|2]:FUNC 'XFR:POW 0';DET NBAN;*WAI
A/R	SENS[1|2]:FUNC 'XFR:POW:RAT 1,0'; DET NBAN;*WAI
B/R	SENS[1|2]:FUNC 'XFR:POW:RAT 2,0'; DET NBAN;*WAI

1 Option 100 only

2 Options 1DA and 1DB only

CHAN 1 | CHAN 2 SCPI Commands

KEYSTROKES	SCPI COMMAND
Broadband Internal	
B*	SENS[1|2]:FUNC 'XFR:POW 2';DET BBAN;*WAI
R*	SENS[1|2]:FUNC 'XFR:POW 0';DET BBAN;*WAI
B*/R*	SENS[1|2]:FUNC 'XFR:POW:RAT 2,0'; DET BBAN;*WAI
Broadband External	
X	SENS[1|2]:FUNC 'XFR:POW 11';DET BBAN;*WAI
Y	SENS[1|2]:FUNC 'XFR:POW 12';DET BBAN;*WAI
X/Y	SENS[1|2]:FUNC 'XFR:POW:RAT 11,12'; DET BBAN;*WAI
Y/X	SENS[1|2]:FUNC 'XFR:POW:RAT 12,11'; DET BBAN;*WAI
Y/R*	SENS[1|2]:FUNC 'XFR:POW:RAT 12,0'; DET BBAN;*WAI
AUX Input	SENS[1|2]:FUNC 'XFR:VOLT';*WAI
Chan OFF	SENS[1|2]:STAT OFF;*WAI

(FREQ) SCPI Commands

KEYSTROKES	SCPI COMMAND
(FREQ)	
Start	DISP:ANN:FREQ[1|2]:MODE SSTOP
(Number) Units	SENS[1|2]:FREQ:STAR <num> [MHZ|KHZ|HZ];*WAI
Stop	DISP:ANN:FREQ[1|2]:MODE SSTOP
(Number) Units	SENS[1|2]:FREQ:STOP <num> [MHZ|KHZ|HZ];*WAI
Center	DISP:ANN:FREQ[1|2]:MODE CSPAN
(Number) Units	SENS[1|2]:FREQ:CENT <num> [MHZ|KHZ|HZ];*WAI
Span	DISP:ANN:FREQ[1|2]:MODE CSPAN
(Number) Units	SENS[1|2]:FREQ:SPAN <num> [MHZ|KHZ|HZ];*WAI
CW	DISP:ANN:FREQ[1|2]:MODE CW;:SENS[1|2]:FREQ:SPAN 0;*WAI
(Number) Units	SENS[1|2]:FREQ:CENT <num> [MHZ|KHZ|HZ];*WAI
Fault Loc Frequency [1]	
Low Pass	SENS:FREQ:MODE LOWP;*WAI
Band Pass	SENS:FREQ:MODE CENT;*WAI
Band Pass Max Span (Number) Units	SENS[1|2]:FREQ:SPAN:MAX <num> [MHZ|KHZ|HZ]

1 Option 100 only

FREQ SCPI Commands (continued)

KEYSTROKES	SCPI COMMAND
Disp Freq Resolution	
MHz	DISP:ANN:FREQ:RES MHZ
kHz	DISP:ANN:FREQ:RES KHZ
Hz	DISP:ANN:FREQ:RES HZ

(POWER) SCPI Commands

KEYSTROKES	SCPI COMMAND
(POWER)	
Level (Number) (ENTER)	SOUR[1|2]:POW <num> [dBm];*WAI
RF ON off	OUTP <ON|OFF>;*WAI
Start Power (Number) (ENTER)	SOUR:POW:STAR <num> [dBm];*WAI
Stop Power (Number) (ENTER)	SOUR:POW:STOP <num> [dBm];*WAI
Pwr Sweep Range [1]	
−13 to Max (dBm)	SOUR:POW:RANG ATT0;*WAI
−23 to −8 (dBm)	SOUR:POW:RANG ATT10;*WAI
−33 to −18 (dBm)	SOUR:POW:RANG ATT20;*WAI
−43 to −28 (dBm)	SOUR:POW:RANG ATT30;*WAI
−53 to −38 (dBm)	SOUR:POW:RANG ATT40;*WAI
−60 to −48 (dBm)	SOUR:POW:RANG ATT50;*WAI
−60 to −58 (dBm)	SOUR:POW:RANG ATT60;*WAI

1 The numbers shown on the range keys will depend on the options installed in the analyzer. Also, if the step attenuator option is not installed, these keys will not appear.

(SWEEP) SCPI Commands

KEYSTROKES	SCPI COMMAND
(SWEEP)	
Sweep Time (Number) (ENTER)	SENS[1|2]:SWE:TIME <num> [as|fs|ps|ns|us|ms|s][1];*WAI
Sweep Time AUTO man	SENS[1|2]:SWE:TIME:AUTO <ON|OFF>;*WAI
Alt Sweep on OFF	SENS:COUP <NONE|ALL>;*WAI
Frequency Sweep	POW:MODE:FIX;*WAI
Power Sweep	POW:MODE:SWE;*WAI

1 If using the microsecond suffix ("us"), the letter "u" must be used. Do not use the Greek character "μ."

MENU SCPI Commands

KEYSTROKES	SCPI COMMAND
MENU	
Trigger	
Continuous	ABOR;:INIT[1|2]:CONT ON;*WAI
Hold	ABOR;:INIT[1|2]:CONT OFF;*WAI
Single	ABOR;:INIT[1|2]:CONT OFF;:INIT[1|2];*WAI
Trigger Source	
Internal	TRIG:SOUR IMM;:SENS:SWE:TRIG:SOUR IMM;*WAI
External Sweep	TRIG:SOUR EXT;:SENS:SWE:TRIG:SOUR IMM;*WAI
External Point	TRIG:SOUR EXT;:SENS:SWE:TRIG:SOUR EXT;*WAI
Number of Points (Number) (ENTER)	SENS[1|2]:SWE:POIN <num>;*WAI
Distance[1]	
Start Distance (Number) (ENTER)	SENS[1|2]:DIST:STAR <num> [FEET|MET];*WAI
Stop Distance (Number) (ENTER)	SENS[1|2]:DIST:STOP <num> [FEET|MET];*WAI
Feet	SENS:DIST:UNIT FEET
Meters	SENS:DIST:UNIT MET
SRL Cable Scan[1]	SENS[1|2]:FUNC:SRL:SCAN;*WAI
Ext Ref on OFF	SENS:ROSC:SOUR <EXT|INT>;*WAI

1 Option 100 only

MENU SCPI Commands (continued)

KEYSTROKES	SCPI COMMAND
Spur Avoid Options	
None	DIAG:SPUR:METH NONE;*WAI
Dither	DIAG:SPUR:METH DITH;*WAI
Spur Avoid	DIAG:SPUR:METH AVO;*WAI

(SCALE) SCPI Commands

KEYSTROKES	SCPI COMMAND
(SCALE)	
Autoscale	DISP:WIND[1|2]:TRAC:Y:AUTO ONCE
Scale/Div (Number) (ENTER)	DISP:WIND[1|2]:TRAC:Y:PDIV <num>
Reference Level (Number) (ENTER)	DISP:WIND[1|2]:TRAC:Y:RLEV <num>
Reference Position (Number) (ENTER)	DISP:WIND[1|2]:TRAC:Y:RPOS <num>
⊛ Phase Offset (Number) (ENTER)	SENS[1|2]:CORR:OFFS:PHAS <num> [DEG]
⊛ Electrical Delay (Number) (ENTER)	SENS[1|2]:CORR:EDEL:TIME <num> [as|fs|ps|ns|us|ms|s][1]

1 If using the microsecond unit terminator, the letter "u" must be used. Do not use the Greek character "μ."

MARKER SCPI Commands

KEYSTROKES	SCPI COMMAND
MARKER	
1: or 1>	CALC[1|2]:MARK1 ON
Number Units	CALC[1|2]:MARK1:X <num> [MHZ|KHZ|HZ]
2: or 2>	CALC[1|2]:MARK2 ON
Number Units	CALC[1|2]:MARK2:X <num> [MHZ|KHZ|HZ]
3: or 3>	CALC[1|2]:MARK3 ON
Number Units	CALC[1|2]:MARK3:X <num> [MHZ|KHZ|HZ]
4: or 4>	CALC[1|2]:MARK4 ON
Number Units	CALC[1|2]:MARK4:X <num> [MHZ|KHZ|HZ]
More Markers	
5: or 5>	CALC[1|2]:MARK5 ON
Number Units	CALC[1|2]:MARK5:X <num> [MHZ|KHZ|HZ]
6: or 6>	CALC[1|2]:MARK6 ON
Number Units	CALC[1|2]:MARK6:X <num> [MHZ|KHZ|HZ]
7: or 7>	CALC[1|2]:MARK7 ON
Number Units	CALC[1|2]:MARK7:X <num> [MHZ|KHZ|HZ]
8: or 8>	CALC[1|2]:MARK8 ON
Number Units	CALC[1|2]:MARK8:X <num> [MHZ|KHZ|HZ]
Active Marker Off	CALC[1|2]:MARK[1|2|…8] OFF

MARKER SCPI Commands (continued)

<table>
<tr><th>KEYSTROKES</th><th>SCPI COMMAND</th></tr>
<tr><td>All Off</td><td>CALC[1|2]:MARK:AOFF</td></tr>
<tr><td>Marker Functions</td><td></td></tr>
<tr><td>   Delta Mkr on OFF</td><td>CALC[1|2]:MARK:MODE <REL|ABS></td></tr>
<tr><td>   Marker -> Center</td><td>SENS[1|2]:FREQ:CENT (CALC[1|2]:MARK[1|2| ... 8]:X:ABS?);*WAI</td></tr>
<tr><td>   Marker -> Reference</td><td>DISP:WIND[1|2]:TRAC:Y:RLEV (CALC[1|2]:MARK[1|2| ... 8]:Y?);*WAI</td></tr>
<tr><td>   Marker -> Elec Delay</td><td>SENS[1|2]:CORR:EDEL:TIME (CALC[1|2]:MARK[1|2| ... 8]:GDEL?);*WAI</td></tr>
<tr><td>   Marker Math</td><td></td></tr>
<tr><td>   Statistics</td><td>CALC[1|2]:MARK:FUNC STAT</td></tr>
<tr><td>   Flatness</td><td>CALC[1|2]:MARK:FUNC FLAT</td></tr>
<tr><td>   RF Filter Stats</td><td>CALC[1|2]:MARK:FUNC FST</td></tr>
<tr><td>   Math Off</td><td>CALC[1|2]:MARK:FUNC OFF</td></tr>
<tr><td>Marker Search</td><td></td></tr>
<tr><td>   Max Search</td><td>CALC[1|2]:MARK:FUNC MAX</td></tr>
<tr><td>   Mkr -> Max</td><td>CALC[1|2]:MARK:FUNC MAX</td></tr>
<tr><td>   Next Peak Left</td><td>CALC[1|2]:MARK:MAX:LEFT</td></tr>
<tr><td>   Next Peak Right</td><td>CALC[1|2]:MARK:MAX:RIGH</td></tr>
<tr><td>   Min Search</td><td>CALC[1|2]:MARK:FUNC MIN</td></tr>
<tr><td>   Marker -> Min</td><td>CALC[1|2]:MARK:FUNC MIN</td></tr>
<tr><td>   Next Min Left</td><td>CALC[1|2]:MARK:MIN:LEFT</td></tr>
<tr><td>   Next Min Right</td><td>CALC[1|2]:MARK:MIN:RIGH</td></tr>
</table>

indicates HP 8712B/14B only

(MARKER) SCPI Commands (continued) (continued)

KEYSTROKES	SCPI COMMAND
Target Search	CALC[1|2]:MARK:FUNC TARG
Target Value (Number) (ENTER)	CALC[1|2]:MARK:TARG <LEFT|RIGH>,<num> [DB]
Search left (Number) (ENTER)	CALC[1|2]:MARK:TARG LEFT,<num> [DB]
Search right (Number) (ENTER)	CALC[1|2]:MARK:TARG RIGH,<num> [DB]
Bandwidth	CALC[1|2]:MARK:FUNC BWID
(Number) (ENTER)	CALC[1|2]:MARK:BWID <num> [DB]
Notch	CALC[1|2]:MARK:FUNC NOTC
(Number) (ENTER)	CALC[1|2]:MARK:NOTC <num> [DB]
More	
Multi Peak	CALC[1|2]:MARK:FUNC MPE
MultiNotch	CALC[1|2]:MARK:FUNC MNOT
Search Off	CALC[1|2]:MARK:FUNC OFF
Tracking on OFF	CALC[1|2]:MARK:FUNC:TRAC <ON|OFF>

DISPLAY SCPI Commands

KEYSTROKES	SCPI COMMAND
DISPLAY	
Normalize	TRAC CH[1|2]SMEM,CH[1|2]SDATA; :CALC[1|2]:MATH (IMPL/CH[1|2]SMEM); :DISP:WIND[1|2]:TRAC1 ON;TRAC2 OFF
Data->Mem	TRAC CH[1|2]SMEM,CH[1|2]SDATA
Data	CALC[1|2]:MATH (IMPL); :DISP:WIND[1|2]:TRAC1 ON;TRAC2 OFF
Memory	DISP:WIND[1|2]:TRAC1 OFF;TRAC2 ON
Data/Mem	CALC[1|2]:MATH (IMPL/CH[1|2]SMEM); :DISP:WIND[1|2]:TRAC1 ON;TRAC2 OFF
Data and Memory	CALC[1|2]:MATH (IMPL); :DISP:WIND[1|2]:TRAC1 ON;TRAC2 ON
Limit Menu	CALC[1|2]:LIM:DISP ON
Add Limit	
Add Max Line	CALC[1|2]:LIM:SEGM[1|2|···12]:TYPE LMAX; STAT ON
Add Min Line	CALC[1|2]:LIM:SEGM[1|2|···12]:TYPE LMIN; STAT ON
Add Max Point	CALC[1|2]:LIM:SEGM[1|2|···12]:TYPE PMAX; STAT ON
Add Min Point	CALC[1|2]:LIM:SEGM[1|2|···12]:TYPE PMIN; STAT ON
Delete Limit	CALC[1|2]:LIM:SEGM[1|2|···12]:STAT OFF
Delete All Limits	CALC[1|2]:LIM:SEGM:AOFF

DISPLAY SCPI Commands (continued)

KEYSTROKES	SCPI COMMAND
Edit Limit	
Begin Frequency (Number) (ENTER)	CALC[1|2]:LIM:SEGM[1|2|···12]:FREQ:STAR <num> [MHZ|KHZ|HZ]
End Frequency (Number) (ENTER)	CALC[1|2]:LIM:SEGM[1|2|···12]:FREQ:STOP <num> [MHZ|KHZ|HZ]
Begin Limit (Number) (ENTER)	CALC[1|2|:LIM:SEGM[1|2|···12]:AMPL:STAR <num>
End Limit (Number) (ENTER)	CALC[1|2]:LIM:SEGM[1|2|···12]:AMPL:STOP <num>
Marker	
Limit Line ON off	CALC[1|2]:LIM:DISP <ON|OFF>
Mkr Limits	
Max Limit (Number) (ENTER)	CALC[1|2]:LIM:MARK:STAT:PEAK:MAX <num>
Min Limit (Number) (ENTER)	CALC[1|2]:LIM:MARK:STAT:PEAK:MIN <num>
Mkr Limit ON off	CALC[1|2]:LIM:MARK:STAT:<MEAN|PEAK|FLAT>
Limit Test on OFF	CALC[1|2]:LIM:STAT <ON|OFF>
More Display	
Split Display FULL Split	DISP:FORM [SING|ULOW]
Title and Clock	
Enter Line 1	DISP:ANN:TITL1:DATA <string>
Enter Line 2	DISP:ANN:TITL2:DATA <string>
Show Clock on Line 1	DISP:ANN:CLOC:MODE LINE1
Show Clock on Line 2	DISP:ANN:CLOC:MODE LINE2
Clock Off	DISP:ANN:CLOC:MODE OFF

DISPLAY **SCPI Commands (continued)**

KEYSTROKES	SCPI COMMAND
Title + Clk ON off	DISP:ANN:TITL <ON|OFF>
Y-Axis Lbl ON off	DISP:ANN:YAX <ON|OFF>
Y-Axis Lbl rel ABS	DISP:ANN:YAX:MODE <REL|ABS>
Graticule ON off	DISP:WIND[1|2]:TRAC:GRAT:GRID <ON|OFF>

FORMAT SCPI Commands

KEYSTROKES	SCPI COMMAND
FORMAT	
Log Mag	CALC[1|2]:FORM MLOG
Lin Mag	CALC[1|2]:FORM MLIN
SWR	CALC[1|2]:FORM SWR
Delay	CALC[1|2]:FORM GDEL
Phase	CALC[1|2]:FORM PHAS
Smith Chart	CALC[1|2]:FORM SMIT
75-ohm Formats [1]	
Mag dBuV	CALC[1|2]:FORM DBUV
Mag dBmV	CALC[1|2]:FORM DBMV
Mag dBV	CALC[1|2]:FORM DBV
More Format	
Polar	CALC[1|2]:FORM POL
Real	CALC[1|2]:FORM REAL
Imaginary	CALC[1|2]:FORM IMAG
Impedance Magnitude	CALC[1|2]:FORM MIMP

1 Option 1EC (75 Ω) only

CAL SCPI Commands

KEYSTROKES	SCPI COMMAND
CAL	
Normalize	TRAC CH[1|2]SMEM,CH[1|2]SDATA; :CALC[1|2]:MATH (IMPL/CH[1|2]SMEM); :DISP:WIND[1|2]:TRAC1 ON;TRAC2 OFF
Transmissn	
Restore Defaults	SENS[1|2]:CORR:CSET DEF;*WAI
Response	SENS[1|2]:CORR:COLL:IST OFF;METH TRAN1;*WAI
Measure Standard	SENS[1|2]:CORR:COLL STAN1;*WAI; :SENS[1|2]:CORR:COLL:SAVE;*WAI
Response & Isolation	SENS[1|2]:CORR:COLL:IST OFF;METH TRAN2;*WAI
Measure Standard — Loads	SENS[1|2]:CORR:COLL STAN1;*WAI;
Measure Standard — Through	SENS[1|2]:CORR:COLL STAN2;*WAI; :SENS[1|2]:CORR:COLL:SAVE;*WAI
Reflection	
Restore Defaults	SENS[1|2]:CORR:CSET DEF;*WAI
One Port	SENS[1|2]:CORR:COLL:IST OFF;METH REFL3; *WAI
Measure Standard — Open	SENS[1|2]:CORR:COLL STAN1;*WAI
Measure Standard — Short	SENS[1|2]:CORR:COLL STAN2;*WAI
Measure Standard — Load	SENS[1|2]:CORR:COLL STAN3;*WAI; :SENS[1|2]:CORR:COLL:SAVE;*WAI

CAL SCPI Commands (continued)

KEYSTROKES	SCPI COMMAND
Fault Location [1]	
Default Cal	SENS[1|2]:CORR:CSET DEF;*WAI
Full Band Cal	SENS[1|2]:CORR:COLL:IST ON;METH REFL3; *WAI
Measure Standard — Open	SENS[1|2]:CORR:COLL STAN1;*WAI
Measure Standard — Short	SENS[1|2]:CORR:COLL STAN2;*WAI
Measure Standard — Load	SENS[1|2]:CORR:COLL STAN3;*WAI; :SENS[1|2]:CORR:COLL:SAVE;*WAI
Velocity Factor	SENS[1|2]:CORR:RVEL:COAX <num>
Cable Loss	SENS[1|2]:CORR:LOSS:COAX <num>
Calibrate Cable	
Specify Length	SENS[1|2]:CORR:LENG:COAX <num> [FEET|MET];*WAI
Measure Cable	SENS[1|2]:CORR:RVEL;*WAI
Multi Peak Corr on OFF	SENS[1|2]:CORR:PEAK:COAX [ON|OFF]
Multi Peak Threshold	SENS[1|2]:CORR:THR:COAX <num>
SRL [2]	
Default Cal	SENS[1|2]:CORR:CSET DEF;*WAI
Full Band Cal	SENS[1|2]:CORR:COLL:IST ON;METH REFL3; *WAI
Measure Standard — Open	SENS[1|2]:CORR:COLL STAN1;*WAI
Measure Standard — Short	SENS[1|2]:CORR:COLL STAN2;*WAI
Measure Standard — Load	SENS[1|2]:CORR:COLL STAN3;*WAI; :SENS[1|2]:CORR:COLL:SAVE;*WAI

1 Option 100 only. This selection (and its lower-level menus) only appears when making fault location measurements

2 Option 100 only. This selection (and its lower-level menus) only appears when making SRL measurements

CAL SCPI Commands (continued)

KEYSTROKES	SCPI COMMAND
Connector Model	
Measure Connector	
Measure	SENS[1|2]:CORR:MODEL:CONN
Connector Length	SENS[1|2]:CORR:LENG:CONN <num>
Connector C	SENS[1|2]:CORR:CAP:CONN <num>
Z cutoff Frequency	SENS[1|2]:FREQ:ZST <num> [GHZ|MHZ|KHZ|HZ]
Auto Z ON off	SENS[1|2]:FUNC:SRL:MODE [AUTO|MAN]
Manual Z	SENS[1|2]:FUNC:SRL:IMP <num>
AM Delay[1]	
Restore Defaults	SENS[1|2]:CORR:COLL:IST OFF;METH TRAN1;*WAI
Response	SENS[1|2]:CORR:COLL:IST OFF;METH TRAN1;*WAI
Measure Standard	SENS[1|2]:CORR:COLL STAN1;*WAI; :SENS[1|2]:CORR:COLL:SAVE;*WAI
Cal Kit	
Default Type-N(f)	SENS:CORR:COLL:CKIT 'COAX,7MM,TYPE-N,50,FEMALE' SENS:CORR:COLL:CKIT 'COAX,7MM,TYPE-N,75,FEMALE' (option 1EC)
Type-N(m)	SENS:CORR:COLL:CKIT 'COAX,7MM,TYPE-N,50,MALE' SENS:CORR:COLL:CKIT 'COAX,7MM,TYPE-N,75,MALE' (option 1EC)

1 Options 1DA and 1DB only

CAL SCPI Commands (continued)

KEYSTROKES	SCPI COMMAND
User Defined	SENS:CORR:COLL:CKIT 'USER,IMPLIED,IMPLIED, IMPLIED,IMPLIED'
3.5 mm	SENS:CORR:COLL:CKIT 'COAX,3.5MM,APC-3.5,50,IMPLIED'
Type-F	SENS:CORR:COLL:CKIT 'COAX,7MM,TYPE-F,75,IMPLIED'[1]
Detector Zero	
Autozero	CAL:ZERO:AUTO ON;*WAI
Manual Zero	CAL:ZERO:AUTO ONCE;*WAI
⊛ More Cal	
⊛ Velocity Factor (Number) (ENTER)	SENS[1|2]:CORR:RVEL:COAX <num>
⊛ Smith Chart ZO (Number) (ENTER)	SENS[1|2]:CORR:IMP:INP:MAGN <num> [OHM]
⊛ Port Ext's on OFF	SENS[1|2]:CORR:EXT [ON|OFF]
⊛ Refl Port Extension (Number) (ENTER)	SENS[1|2]:CORR:EXT:REFL:TIME <num> [as|fs|ps|ns|us|ms|s][2]
⊛ Trans Port Extension (Number) (ENTER)	SENS[1|2]:CORR:EXT:TRAN:TIME <num> [as|fs|ps|ns|us|ms|s][2]

1 Implemented in firmware revisions B.03.01 and above.

2 If using the microsecond unit terminator, the letter "u" must be used. Do not use the Greek character "μ."

AVG SCPI Commands

<table>
<tr><th>KEYSTROKES</th><th>SCPI COMMAND</th></tr>
<tr><td>AVG</td><td></td></tr>
<tr><td>Average on OFF</td><td>SENS[1|2]:AVER <ON|OFF>;*WAI</td></tr>
<tr><td>Restart Average</td><td>SENS[1|2]:AVER:CLE;*WAI</td></tr>
<tr><td>Average Factor (Number) (ENTER)</td><td>SENS[1|2]:AVER:COUN <num>;*WAI</td></tr>
<tr><td>System Bandwidth</td><td></td></tr>
<tr><td>Wide</td><td>SENS[1|2]:BWID 6500 HZ;*WAI</td></tr>
<tr><td>Medium</td><td>SENS[1|2]:BWID 3700 HZ;*WAI</td></tr>
<tr><td>Narrow</td><td>SENS[1|2]:BWID 250 HZ;*WAI</td></tr>
<tr><td>Fine</td><td>SENS[1|2]:BWID 15 HZ;*WAI</td></tr>
<tr><td>Fault Window [1]</td><td></td></tr>
<tr><td>Minimum</td><td>SENS[1|2]:WIND RECT</td></tr>
<tr><td>Medium</td><td>SENS[1|2]:WIND HAMM</td></tr>
<tr><td>Maximum</td><td>SENS[1|2]:WIND KBES</td></tr>
<tr><td>Delay Aperture</td><td></td></tr>
<tr><td>Aperture (Hz) (Number) (ENTER)</td><td>CALC[1|2]:GDAP:SPAN <num> [HZ];*WAI</td></tr>
<tr><td>Aperture (%) (Number) (ENTER)</td><td>CALC[1|2]:GDAP:APER <num>;*WAI</td></tr>
</table>

1 Option 100 only

(SAVE RECALL) SCPI Commands

KEYSTROKES	SCPI COMMAND
(SAVE RECALL)	
Save State	MMEM:STOR:STAT 1,<file>[1]
Re-Save State	MMEM:STOR:STAT 1,<file>[1]
Define Save	
Inst State ON off	MMEM:STOR:STAT:IST <ON|OFF>
Cal on OFF	MMEM:STOR:STAT:CORR <ON|OFF>
Data on OFF	MMEM:STOR:STAT:TRAC <ON|OFF>
Save ASCII	
Save Chan 1	MMEM:STOR:TRAC CH1FDATA,<file>[1]
Save Chan 2	MMEM:STOR:TRAC CH2FDATA,<file>[1]
Recall State	MMEM:LOAD:STAT 1,<file>[1]

1 <**file**> may include the mass storage device mnemonic **MEM:**, **INT:**, **EXT:**, or **RAM:** before the actual name of the file. If the mass storage device is not explicitly named the currently selected device is assumed. <**file**>, <**file1**> and <**file2**> are <**string**> parameters. <**string**> parameters appear between single quotes.

SAVE RECALL SCPI Commands (continued)

KEYSTROKES	SCPI COMMAND
Programs	
Save Program	
Re-Save Program	
File Type bin ASCII	
Recall Program	
Save AUTOST	
IBASIC[1]	
Select Disk	
Non-Vol RAM Disk	MMEM:MSIS 'MEM:'
Volatile RAM Disk	MMEM:MSIS 'RAM:'
Internal 3.5" Disk	MMEM:MSIS 'INT:'
External Disk	MMEM:MSIS 'EXT:'[2]
Configure VOL_RAM	
Restore Defaults	No SCPI command
Modify Size	No SCPI command
Current Size	No SCPI command
Configure Ext Disk	
Ext Disk Address (Number) (ENTER)	SYST:COMM:GPIB:MMEM:ADDR <num>
Ext Disk Unit (Number) (ENTER)	SYST:COMM:GPIB:MMEM:UNIT <num>
Ext Disk Volume (Number) (ENTER)	SYST:COMM:GPIB:MMEM:VOL <num>

1 The IBASIC menu is described under the **SYSTEM OPTIONS** key.

2 Active controller status must be passed to the instrument (from IBASIC or the external controller) for external disk access.

SAVE RECALL SCPI Commands (continued)

KEYSTROKES	SCPI COMMAND
File Utilities	
Rename File	MMEM:MOVE <file1>,<file2>[1]
Delete File	MMEM:DEL <file>[1]
Delete All Files	MMEM:DEL '*.*'[1,4]
Copy File	
Copy to NonVol RAM	MMEM:COPY <file1>,<'MEM:file2'>[1]
Copy to Vol RAM	MMEM:COPY <file1>,<'RAM:file2'>[1]
Copy to 3.5" Int Disk	MMEM:COPY <file1>,<'INT:file2'>[1]
Copy to Ext Disk	MMEM:COPY <file1>,<'EXT:file2'>[1,2]
Copy All Files	
Copy to NonVol RAM	MMEM:COPY '*.*','MEM:'[1,3]
Copy to Vol RAM	MMEM:COPY '*.*','RAM:'[1,3]
Copy to 3.5" Int Disk	MMEM:COPY '*.*','INT:'[1,3]
Copy to Ext Disk	MMEM:COPY '*.*','EXT:'[1,2,3]
Format Disk	
Format NonVol RAM	MMEM:INIT 'MEM:',<DOS|LIF>[4]
Format Vol RAM	MMEM:INIT 'RAM:',<DOS|LIF>[4]
Format 3.5" Disk	MMEM:INIT 'INT:',<DOS|LIF>[4]
Format Ext Disk	MMEM:INIT 'EXT:',<DOS|LIF>[2,4]

1 <**file**> may include the mass storage device mnemonic **MEM:**, **INT:**, **EXT:**, or **RAM:** before the actual name of the file. If the mass storage device is not explicitly named the currently selected device is assumed. <**file**>, <**file1**> and <**file2**> are <**string**> parameters.

2 Active controller status must be passed to the instrument (from IBASIC or the external controller) for external disk access.

3 **'*.*'** is the form for "all files" with a DOS formatted disk — a LIF formatted disk uses **'*'** with no extension.

4 When a disk is formatted using the front panel keys the DOS format is always used. The LIF format is available when the mnemonic is used.

(SAVE RECALL) SCPI Commands (continued)

KEYSTROKES	SCPI COMMAND
Directory Utilities	
Change Directory	MMEM:CDIR <directory>[1]
Make Directory	MMEM:MDIR <directory>[1]
Remove Directory	MMEM:RDIR <directory>[1]
FastRecall on OFF	DISP:MENU:REC:FAST <ON|OFF>

1 For use with DOS formatted disks only — the analyzer does not support LIF disks that use HFS (hierarchical file structure). <**directory**> is a <**string**> parameter.

HARD COPY **SCPI Commands**

KEYSTROKES	SCPI COMMAND
HARD COPY	
Start	HCOP;*WAI
Abort	HCOP:ABOR
Select Copy Port	
Restore Defaults	No SCPI Command
Select	HCOP:DEV:LANG <PCL|HPGL|IBM|EPSON|PCX>; PORT <CENT|SER|GPIB|MMEM>
Hardcopy Address (Number) (ENTER)	SYST:COMM:GPIB:HCOP:ADDR <num>
Baud Rate (Number) (ENTER)	SYST:COMM:SER:TRAN:BAUD <num>
Xon/Xoff	SYST:COMM:SER:TRAN:HAND XON
DTR/DSR	SYST:COMM:SER:TRAN:HAND DTR
Define Printer	
Restore Defaults	No SCPI Command
Monochrome	HCOP:DEV1:COL OFF
Color	HCOP:DEV1:COL ON
Portrait	HCOP:PAGE:ORI PORT
Landscape	HCOP:PAGE:ORI LAND
Auto Feed ON off	HCOP:ITEM1:FFE:STAT <ON|OFF>

HARD COPY **SCPI Commands (continued)**

KEYSTROKES	SCPI COMMAND
More Printer	
Restore Defaults	No SCPI Command
Printer Resolution (Number) (ENTER)	HCOP:DEV:RES <num>
Top Margin (Number) (ENTER)	HCOP:PAGE:MARG:TOP <num>
Left Margin (Number) (ENTER)	HCOP:PAGE:MARG:LEFT <num>
Print Width (Number) (ENTER)	HCOP:PAGE:WIDT <num>
Define Plotter	
Restore Defaults	No SCPI Command
Monochrome	HCOP:DEV2:COL OFF
Color	HCOP:DEV2:COL ON
Set Pen Numbers	
Monochrome Pen (Number) (ENTER)	No SCPI Command
Default Pen Colors	No SCPI Command
Trace 1 Pen (Number) (ENTER)	No SCPI Command
Trace 2 Pen (Number) (ENTER)	No SCPI Command
Memory 1 Pen (Number) (ENTER)	No SCPI Command
Memory 2 Pen (Number) (ENTER)	No SCPI Command
Graticule Pen (Number) (ENTER)	No SCPI Command
Graphics Pen (Number) (ENTER)	No SCPI Command
Auto Feed on OFF	HCOP:ITEM2:FFE:STAT <ON|OFF>

HARD COPY SCPI Commands (continued)

KEYSTROKES	SCPI COMMAND
Define Hardcopy	
Restore Defaults	No SCPI Command
Graph and Mkr Table	HCOP:DEV:MODE GMAR
Graph Only	HCOP:DEV:MODE GRAP
Mkr Table Only	HCOP:DEV:MODE MARK
List Trace Values	HCOP:DEV:MODE TABL
Define Graph	
Restore Defaults	No SCPI Command
Trace Data ON off	HCOP:ITEM:TRAC:STAT <ON|OFF>
Graticule ON off	HCOP:ITEM:GRAT:STAT <ON|OFF>
Annotation ON off	HCOP:ITEM:ANN:STAT <ON|OFF>
Mkr Symbol ON off	HCOP:ITEM:MARK:STAT <ON|OFF>
Title + Clk ON off	HCOP:ITEM:TITL:STAT <ON|OFF>

(SYSTEM OPTIONS) SCPI Commands

KEYSTROKES	SCPI COMMAND
(SYSTEM OPTIONS)	
IBASIC	
Run	PROG:STAT RUN
Continue	PROG:STAT CONT
Step	PROG:EXEC 'STEP'
Edit	No SCPI Command
Key Record on OFF	No SCPI Command
Utilities	
Clear Program	PROG:DEL
Stack Size	PROG:MALL <size>
Secure	No SCPI Command
IBASIC Display	
None	DISP:PROG OFF
Full	DISP:PROG FULL
Upper	DISP:PROG UPP
Lower	DISP:PROG LOW
HP-IB	
HP 8714B Address (Number) (ENTER)	SYST:COMM:GPIB:ADDR <num>[1]
Talker Listener	SYST:COMM:GPIB:CONT OFF[2]
System Controller	SYST:COMM:GPIB:CONT ON[2]
HP-IB Echo on OFF	SYST:COMM:GPIB:ECHO <ON|OFF>

1 A five second delay is required before a command is sent to the new address.

2 For use with IBASIC running on the analyzer's internal controller — this command cannot be executed from an external controller. Use ***OPC?** and wait for a reply before sending any **OUTPUT 7xx** commands from IBASIC.

SYSTEM OPTIONS SCPI Commands (continued)

KEYSTROKES	SCPI COMMAND
Operating Parameters	
Hardcopy Screen	No SCPI Command
Hardcopy All	HCOP:DEV:MODE ISET;:HCOP;*WAI
Abort	HCOP:ABOR
System Config	
Set Clock	
Set Year Number ENTER	SYST:DATE <year>,<month>,<day>[1]
Set Month Number ENTER	SYST:DATE <year>,<month>,<day>[1]
Set Day Number ENTER	SYST:DATE <year>,<month>,<day>[1]
Set Hour Number ENTER	SYST:TIME <hour>,<minute>,<second>[1]
Set Minute Number ENTER	SYST:TIME <hour>,<minute>,<second>[1]
Round Seconds	SYST:TIME <hour>,<minute>,0[1]
Clock Format	
YYYY-MM-DD HH:MM	DISP:ANN:CLOC:DATE:FORM YMD
MM-DD-YYYY HH:MM	DISP:ANN:CLOC:DATE:FORM MDY
DD-MM-YYYY HH:MM	DISP:ANN:CLOC:DATE:FORM DMY
Numeric	DISP:ANN:CLOC:DATE:MODE NUM
Alpha	DISP:ANN:CLOC:DATE:MODE ALPH
Seconds ON off	DISP:ANN:CLOC:SEC <ON|OFF>
Done	
Beeper Volume Number ENTER	SYST:BEEP:VOL <num>[2]

1 <**year**>,<**month**>,<**day**>,<**hour**>,<**minute** > and <**second**> are all <**num**> parameters. Also, these keys do not generate keystroke recording BASIC statements.

2 Number is a fraction, for example 90% would be expressed as 0.90

SYSTEM OPTIONS **SCPI Commands (continued)**

KEYSTROKES	SCPI COMMAND
External CRT Adjust	
Restore Defaults	No SCPI Command
Vertical Back Porch (Number) (ENTER)	No SCPI Command
Vertical Frnt Porch (Number) (ENTER)	No SCPI Command
Horizontal Back Porch (Number) (ENTER)	No SCPI Command
Horizontal Frnt Porch (Number) (ENTER)	No SCPI Command
Service [1]	

1 The Service menu is described in the *Service Guide*.

12

SCPI Command Summary

SCPI Command Summary

This chapter contains all of the HP-IB commands recognized by the analyzer and a brief description. `<num>`, `<char>`, `<string>` and `<block>` refer to the parameter type expected by the instrument as part of the command. All commands have both command and query forms unless specified as command only or query only. Unless otherwise specified, add a "?" to create a query from the command form. For example, the command to select the log magnitude format for the data displayed is `CALCulate[1|2]:FORMat MLOGarithmic`. To query which format is active the corresponding command is `CALCulate[1|2]:FORMat?`. The response to the query is the short form of the mnemonic for the active format, in this example `MLOG`.

The FORM column gives the parameter type returned by the instrument in response to a query. NR1, NR2 and NR3 refer to the different types of numeric data. CHAR (character data), STRING (string data) and BLOCK (block data) are also used to describe response types. These parameter types are described in the "Parameter Types" section of Chapter 10.

Some numeric parameters may be followed by an appropriate suffix. Commands that accept a suffix also allow standard metric multipliers to be combined with the suffix. For example, commands that set a frequency will accept HZ, KHZ, MHZ and GHZ. Commands that set a time will accept S, MS, US, NS, PS, FS and AS. Note that case is ignored. The multiplier "M" is interpreted as either milli or Mega, depending on context. If no suffix is included, the default units for the parameter are used.

ABORt

SUBSYSTEM COMMANDS	FORM	DESCRIPTION
ABORt	command only	Abort and reset the sweep in progress.

CALCulate

<table>
<tr><th>SUBSYSTEM COMMANDS</th><th>FORM</th><th>DESCRIPTION</th></tr>
<tr><td>CALCulate[1|2]:DATA?[1]</td><td>query only
BLOCK or NR3[2]</td><td>Query the formatted data trace — functionally equivalent to the command TRAC? CH<1|2>FDATA.</td></tr>
<tr><td>CALCulate[1|2]:FORMat <char></td><td>CHAR</td><td>Select the display format for the measurement data — choose from
MLOGarithmic|MLINear|SWR|DBV[3]|DBMV[3]|
DBUV[3]
or
—PHASe|SMITh|POLar|GDELay|REAL

< ?\vglue -1.75pc> |IMAGinary|MIMPedance.</td></tr>
<tr><td>CALCulate[1|2]:GDAPerture:APERture <num></td><td>NR3</td><td>Set the group delay aperture as a ratio of desired aperture / measured frequency span.</td></tr>
<tr><td>CALCulate[1|2]:GDAPerture:SPAN <num></td><td>NR3</td><td>Specifies the group delay aperture in Hertz.</td></tr>
<tr><td>CALCulate[1|2]:LIMit:DISPlay <ON|OFF>[4]</td><td>NR1</td><td>Turn on/off display of limit lines.</td></tr>
<tr><td>CALCulate[1|2]:LIMit:MARKer:FLATness :MAXimum <num></td><td>NR3</td><td>Set the maximum value for a flatness limit test.</td></tr>
<tr><td>CALCulate[1|2]:LIMit:MARKer:FLATness :MINimum <num></td><td>NR3</td><td>Set the minimum value for a flatness marker limit test.</td></tr>
<tr><td>CALCulate[1|2]:LIMit:MARKer:FLATness :STATe <ON|OFF>[4]</td><td>NR1</td><td>Turn on/off flatness marker limit test.</td></tr>
<tr><td>CALCulate[1|2]:LIMit:MARKer:STATistic :MEAN:MAXimum <num></td><td>NR3</td><td>Set the maximum value for a statistic mean limit test.</td></tr>
<tr><td>CALCulate[1|2]:LIMit:MARKer:STATistic :MEAN:MINimum <num></td><td>NR3</td><td>Set the minimum value for a statistic mean limit test.</td></tr>
<tr><td>CALCulate[1|2]:LIMit:MARKer:STATistic :MEAN:STAte <ON|OFF>[4]</td><td>NR1</td><td>Turn on/off statistic mean marker limit test.</td></tr>
</table>

1 Refer to Chapter 6, “Trace Data Transfers,” and to the “ASCDATA” and “REALDATA” example programs in Chapter 8 for more information on this command.

2 The parameter type of the data is determined by the format selected — **FORMat REAL** uses BLOCK data, **FORMat ASCii** uses NR3 data separated by commas.

3 Option 1EC (75 Ω) only

4 Binary parameters accept the values of **1** (on) and **0** (off) in addition to **ON** and **OFF**.

CALCulate (continued)

SUBSYSTEM COMMANDS	FORM	DESCRIPTION
CALCulate[1|2]:LIMit:MARKer:STATistic :PEAK:MAXimum <num>	NR3	Set the maximum value for a statistic peak-to-peak limit test.
CALCulate[1|2]:LIMit:MARKer:STATistic :PEAK:MINimum <num>	NR3	Set the minimum value for a statistic peak-to-peak limit test.
CALCulate[1|2]:LIMit:MARKer:STATistic :PEAK:STAte <ON|OFF>[1]	NR1	Turn on/off statistic peak-to-peak marker limit test.
CALCulate[1|2]:LIMit:SEGMent[1|2|…12]:AMPLitude:STARt <num>[2]	NR3	Set the Begin Limit for the specified limit segment.
CALCulate[1|2]:LIMit:SEGMent[1|2|…12]:AMPLitude:STOP <num>[2]	NR3	Set the End Limit for the specified limit segment.
CALCulate[1|2]:LIMit:SEGMent:AOFF	command only	Turn off all limit segments for a given channel — deletes all segments in the channel's limit table.
CALCulate[1|2]:LIMit:SEGMent[1|2|…12]:FREQuency:STARt <num>[2]	NR3	Set the Begin Frequency for the specified limit segment.
CALCulate[1|2]:LIMit:SEGMent[1|2|…12]:FREQuency:STOP <num>[2]	NR3	Set the End Frequency for the specified limit segment.
CALCulate[1|2]:LIMit:SEGMent[1|2|…12]:POWer:STARt <num>	NR3	Set the Begin Power for the specified limit segment.
CALCulate[1|2]:LIMit:SEGMent[1|2|…12]:POWer:STOP <num>	NR3	Set the End Power for the specified limit segment.
CALCulate[1|2]:LIMit:SEGMent[1|2|…12]:STATe <ON|OFF>[1]	NR1	Turn on/off the specified limit segment — adds or deletes the segment.
CALCulate[1|2]:LIMit:SEGMent[1|2|…12]:TYPE <char>	CHAR	Set the limit type for the specified segment, choose from **LMAX|LMIN|PMAX|PMIN** (Max Line, Min Line, Max Point, Min Point) — sets all of the segment's limit parameters to their default values.

1 Binary parameters accept the values of **1** (on) and **0** (off) in addition to **ON** and **OFF**.

2 Numeric parameters may include an appropriate suffix; if no suffix is included, the default (**HZ** for frequency or **S** for time) is assumed.

CALCulate (continued)

SUBSYSTEM COMMANDS	FORM	DESCRIPTION
CALCulate[1|2]:LIMit:STATe <ON|OFF>[1]	NR1	Turn on/off the limit test.
CALCulate[1|2]:MARKer:AOFF	command only	Turn off all markers for a given channel — this has the effect of turning off marker functions and tracking as well.
CALCulate[1|2]:MARKer:BWIDth <num>[2]	NR3	Calculate the bandwidth of a bandpass filter — **num** is the target bandwidth (-3 for the 3 dB bandwidth).
CALCulate[1|2]:MARKer:FUNCtion :RESult?	query only NR3[,NR3, NR3,NR3]	Query the results of the active marker function — **MAX** and **MIN** return the amplitude; **TARG** returns the frequency; **BWID** returns bandwidth, center frequency, Q and loss; **STAT** returns the frequency span, the mean and standard deviation of the amplitude response, and the peak-to-peak ripple; **FLAT** returns the frequency span, gain, slope and flatness; and **FSTAT** returns the insertion loss and peak-to-peak ripple of the passband of a filter, as well as the maximum signal amplitude in the stopband. Refer to the "MARKERS" example program in Chapter 8 for more information.
CALCulate[1|2]:MARKer:FUNCtion [:SELect] <char>	CHAR	Select the active marker function — choose from **OFF|MAXimum|MINimum|TARGet|BWIDth|NOTCh|MPEak|MNOTch|STATistics|FLATness|FSTATistics**.
CALCulate[1|2]:MARKer:FUNCtion :TRACking <ON|OFF>[1]	NR1	Turn on/off marker function tracking.
CALCulate[1|2]:MARKer[1|2|···8]:GDELay?	query only	Returns the group delay value, in seconds, at the specified marker.
CALCulate[1|2]:MARKer[1|2|···8] :MAXimum	command only	Set the specified marker to the maximum value on the trace.
CALCulate[1|2]:MARKer[1|2|···8] :MAXimum:LEFT	command only	Moves the specified marker to the next local maximum to the left.
CALCulate[1|2]:MARKer[1|2|···8] :MAXimum:RIGHt	command only	Moves the specified marker to the next local maximum to the right.

1 Binary parameters accept the values of **1** (on) and **0** (off) in addition to **ON** and **OFF**.

2 Numeric parameters may include an appropriate suffix; if no suffix is included, the default (**HZ** for frequency or **S** for time) is assumed.

CALCulate (continued)

SUBSYSTEM COMMANDS	FORM	DESCRIPTION
CALCulate[1|2]:MARKer[1|2|…8] :MINimum	command only	Set the specified marker to the minimum value on the trace.
CALCulate[1|2]:MARKer[1|2|…8] :MINimum:LEFT	command only	Moves the specified marker to the next local minimum to the left.
CALCulate[1|2]:MARKer[1|2|…8] :MINimum:RIGHt	command only	Moves the specified marker to the next local minimum to the right.
CALCulate[1|2]:MARKer:MODE <char>	CHAR	Turn on/off delta marker state — choose ABSolute or RELative.
CALCulate[1|2]:MARKer:NOTCh <num>[1]	NR3	Calculate the notch width of a notch filter — num is the target notch width (-6 for the 6dB bandwidth).
CALCulate[1|2]:MARKer[1|2|…8] :POINt[2]	NR3	Set the specified marker point.
CALCulate[1|2]:MARKer[1|2|…8]:X <num>	NR3	Set the specified marker frequency (or power if in power sweep)
CALCulate[1|2]:MARKer:REFerence:X?	query only NR3	Query the frequency of the reference marker.
CALCulate[1|2]:MARKer:REFerence:Y?	query only NR3	Query the amplitude of the reference marker.
CALCulate[1|2]:MARKer[1|2|…8] [:STATe] <ON|OFF>	NR1	Turn on/off the specified marker.
CALCulate[1|2]:MARKer[1|2|…8] :TARGet <char>,<num>[1]	CHAR,NR3	Perform a marker search for a target value — char is the direction LEFT or RIGHt.
⊕CALCulate[1|2]:MARKer[1|2|…8]:Y :INDuctance ?	query only NR3	Query the specified marker's inductance when in Smith chart format.
⊕CALCulate[1|2]:MARKer[1|2|…8]:Y :MAGNitude?	query only NR3	Query the specified marker's magnitude when in polar format.

1 Numeric parameters may include an appropriate suffix; if no suffix is included the default (**HZ** for frequency or **S** for time) is assumed.

2 Refer to "Displaying Measurement Results" in Chapter 7 of the *User's Guide* for more information on using this command.

CALCulate (continued)

<table>
<tr><th>SUBSYSTEM COMMANDS</th><th>FORM</th><th>DESCRIPTION</th></tr>
<tr><td>CALCulate[1|2]:MARKer[1|2|...8]:Y

:PHASe?</td><td>query only
NR3</td><td>Query the specified marker's phase value when in polar format.</td></tr>
<tr><td>CALCulate[1|2]:MARKer[1|2|...8]:Y

:REACtance?</td><td>query only
NR3</td><td>Query the specified marker's reactance value when in Smith chart format.</td></tr>
<tr><td>CALCulate[1|2]:MARKer[1|2|...8]:Y

:RESistance ?</td><td>query only
NR3</td><td>Query the specified marker's resistance value when in Smith chart format.</td></tr>
<tr><td>CALCulate[1|2]:MARKer[1|2|...8]:Y?</td><td>query only
NR3</td><td>Query the specified marker amplitude.</td></tr>
<tr><td>CALCulate[1|2]:MATH[:EXPRession] <expr>[1]</td><td>EXPR[1]</td><td>Select a trace math expression — choose measurement trace from (IMPL) for "data only" or (IMPL/CH<1|2>SMEM) for "data / memory".</td></tr>
</table>

1 <**expr**> and EXPR represent expressions, a parameter type that consists of mathematical expressions that use character parameters and are enclosed in parentheses.

CALibration

<table>
<tr><th>SUBSYSTEM COMMANDS</th><th>FORM</th><th>DESCRIPTION</th></tr>
<tr><td>CALibration:ZERO:AUTO <ON|OFF|ONCE>[1]</td><td>NR1</td><td>Turn on/off the broadband detector autozeroing function.</td></tr>
</table>

1 Binary parameters accept the values of **1** (on) and **0** (off) in addition to **ON** and **OFF**.

CONFigure

SUBSYSTEM COMMANDS	FORM	DESCRIPTION
CONFigure <string>	STRING	Configure the analyzer to measure a specific device type and parameter (the BEGIN function) — choose from one of the following strings: 'AMPLifier:TRANsmission' 'AMPLifier:REFLection' 'AMPLifier:POWer' 'FILTer:TRANsmission' 'FILTer:REFLection' 'BBANd:TRANsmission' 'BBANd:REFLection' 'MIXer:CLOSs' 'MIXer:GDEL' 'MIXer:REFLection' 'CABLe:TRANsmission' 'CABLe:REFLection' 'CABLe:FAULT' 'CABLe:SRL'

DIAGnostic

SUBSYSTEM COMMANDS	FORM	DESCRIPTION
DIAGnostic:CCONstans:INSTalled?	query only NR1	Query if correction constants are installed in flash. Returns a 1 if true, and a 0 if false.
DIAGnostic:CCONstants:LOAD	command only	Load default factory calibration constants from floppy disk to memory.
DIAGnostic:CCONstants:STORE:DISK	command only	Store default factory calibration constants from memory to floppy disk.
DIAGnostic:CCONstants:STORE:EEPRom	command only	Store default factory calibration constants from memory to flash EEPROM.
DIAGnostic:PORT:READ? <port><register>[1]	query only NR1, NR1	Reads the rear panel I/O ports.
DIAGnostic:PORT:WRITE <port><register>[1]	NR1, NR1, NR1	Writes to the rear panel I/O ports.
DIAGnostic:SNUMber <string>?	query only STRING	Query the instrument's serial number.
DIAGnostic:SPUR:METHod <NONE|DITHer|AVOid>	NR1	Select the spur avoid mode.

1 Refer to "Controlling Peripherals" in Chapter 7 of the *User's Guide* for more information on using this command.

Table 12-1. Writeable Ports

Port Number	Register	Description
15	0	Outputs 8-bit data to the Cent_D0 through D7 lines of the Centronics port. Cent_D0 is the least significant bit, Cent_D7 is the most significant bit. Checks Centronics status lines for: Out of Paper Printer Not on Line BUSY ACKNOWLEDGE
15	1	Sets/clears the user bit according to the least significant bit of A. A least significant bit equal to 1 sets the user bit high. A least significant bit of 0 clears the user bit.
15	2	Sets/clears the limit pass/fail bit according to the least significant bit of A. A least significant bit equal to 1 sets the pass/fail bit high. A least significant bit of 0 clears the pass/fail bit.
15	3	Outputs 8-bit data to the Cent_D0 through D7 lines of the Centronics port. Cent_D0 is the least significant bit, Cent_D7 is the most significant bit. Does not check Centronics status lines.
9	0	Outputs a byte to the serial port. The byte is output serially according to the configuration for the serial port.

NOTE

When using the `WRITEIO(15,0)` or `WRITEIO(15,3)` command, the Printer_Select Line is set High. However, when the instrument is doing hardcopy, the Printer_Select Line is set low. The Printer_Select line may or may not be used by individual printers. Check with your printer manual.

Table 12-2. Readable Ports

Port Number	Register	Description
9	0	Reads the serial port.
15	0	Reads the 8-bit data port Cent_D0 through D7.
15	1	Reads the user bit.
15	2	Reads the limit test pass/fail bit.
15	10	Reads the 8-bit status port. D0—Cent_acknowledge D1—Cent_busy D2—Cent_out_of_paper D3—Cent_on_line D4—Cent_printer_err

DISPlay

<table>
<tr><th>SUBSYSTEM COMMANDS</th><th>FORM</th><th>DESCRIPTION</th></tr>
<tr><td>DISPlay:ANNotation:CHANnel[1|2]:USER :STATe <OFF|ON>[1,2]</td><td>NR1</td><td>Enables user-defined channel annotation.</td></tr>
<tr><td>DISPlay:ANNotation:CHANnel[1|2]:USER :LABel:DATA <string>[2]</td><td>STRING</td><td>Specifies the string to be displayed in the channel annotation area (above the graticule).</td></tr>
<tr><td>DISPlay:ANNotation:CLOCk:DATE :FORMat <char></td><td>CHAR</td><td>Select the Year/Month/Day ordering of the date in the clock display — choose from YMD|MDY|DMY.</td></tr>
<tr><td>DISPlay:ANNotation:CLOCk:DATE:MODE <char></td><td>CHAR</td><td>Select the format for the date in the clock display — choose NUMeric or ALPHa.</td></tr>
<tr><td>DISPlay:ANNotation:CLOCk:MODE <char></td><td>CHAR</td><td>Select how the clock will appear in the measurement display title area — choose from LINE1|LINE2|OFF.</td></tr>
<tr><td>DISPlay:ANNotation:CLOCk:SEConds [:STATe] <ON|OFF>[1]</td><td>NR1</td><td>Turn on/off display of seconds in the clock display.</td></tr>
<tr><td>DISPlay:ANNotation:FREQuency[1|2] :MODE <char></td><td>CHAR</td><td>Set the frequency annotation on the display — choose SSTOP (start/stop), CSPAN (center/span) or CW.</td></tr>
<tr><td>DISPlay:ANNotation:FREQuency[1|2] :RESolution <char></td><td>CHAR</td><td>Set the resolution of display frequency values — choose from MHZ|KHZ|HZ.</td></tr>
<tr><td>DISPlay:ANNotation:FREQuency[1|2] :USER:STATe [OFF|ON][1,2]</td><td>NR1</td><td>Enables user-defined frequency annotation.</td></tr>
<tr><td>DISPlay:ANNotation:FREQuency[1|2] :USER:STARt <num>[2]</td><td>NR3</td><td>Specifies the start value for user-defined frequency annotation.</td></tr>
<tr><td>DISPlay:ANNotation:FREQuency[1|2] :USER:STOP <num>[2]</td><td>NR3</td><td>Specifies the stop value for user-defined frequency annotation.</td></tr>
<tr><td>DISPlay:ANNotation:FREQuency[1|2] :USER:SUFFix:DATA <string>[2]</td><td>STRING</td><td>Specifies the suffix for user defined frequency annotation.</td></tr>
<tr><td>DISPlay:ANNotation:FREQuency[1|2] :USER:LABel:DATA <string></td><td>STRING</td><td>A user-defined X-axis label.</td></tr>
<tr><td>DISPlay:ANNotation:MARKer[1|2] [:STATe] <ON|OFF>[1]</td><td>NR1</td><td>Enable/disable the active marker annotation for channels 1 and 2.</td></tr>
</table>

1 Binary parameters accept the values of **1** (on) and **0** (off) in addition to **ON** and **OFF**.

2 Refer to “Displaying measurement Results” in Chapter 7 of the *User's Guide* for more information on using this command.

DISPlay (continued)

SUBSYSTEM COMMANDS	FORM	DESCRIPTION
DISPlay:ANNotation:MESSage:AOFF	command only	Turns off any currently showing message window — includes message window, active entry and IBASIC window.
DISPlay:ANNotation:MESSage:CLEar[1]	command only	Removes a user-defined pop-up message window.
DISPlay:ANNotation:MESSage:STATe <ON|OFF>[2]	NR1	Enable/disable the message window — CAUTION: this suppresses display of all messages (even ERROR messages).
DISPlay:ANNotation:MESSage:DATA <string>[1]	STRING	Displays a user-defined message in the pop-up message window. Optional argument specifies the timeout: choose from NONE|SHORt|MEDium|LONG.
DISPlay:ANNotation:TITLe[1|2]:DATA <string>[1]	STRING	Enter a string for the specified title line.
DISPlay:ANNotation:TITLe[:STATe] <ON|OFF>[2]	NR1	Turn on/off display of the title and clock.
DISPlay:ANNotation:YAXis:MODE <char>	CHAR	Set mode for the Y-axis labels — choose RELative or ABSolute
DISPlay:ANNotation:YAXis[:STATe] <ON|OFF>[2]	NR1	Turn on/off Y-axis labels.
DISPlay:FORMat <char>	CHAR	Select the format (full or split screen) for displaying trace data — choose SINGle (overlay) or ULOWer (split).
DISPlay:MENU:KEY[1|2|···7] <string>[1]	STRING	Specifies the softkey menu labels from a remote controller or IBASIC
DISPlay:MENU[2]:KEY[1|2|···7] <string>[1]	STRING	Specifies the softkey menu labels when using user-defined BEGIN key. (For option 1C2, IBASIC, only)
DISPlay:MENU:RECall:FAST[:STATe] <ON|OFF>[2]	NR1	Turn on/off fast recall mode.
DISPlay:PROGram[:MODE] <char>	CHAR	Select the portion of the analyzer's screen to be used as an HP Instrument BASIC display — choose from OFF|FULL|UPPer|LOWer.
DISPlay:WINDow[1|2|10]:GEOMetry :LLEFT?	query only NR1,NR1	Query the absolute pixel coordinates of the lower left corner of the selected display window.

1 Refer to "Operator Interaction" in Chapter 7 of the *User's Guide* for more information on using this command.

2 Binary parameters accept the values of **1** (on) and **0** (off) in addition to **ON** and **OFF**.

DISPlay (continued)

<table>
<tr><th>SUBSYSTEM COMMANDS</th><th>FORM</th><th>DESCRIPTION</th></tr>
<tr><td>DISPlay:WINDow[1|2|10]:GEOMetry
:SIZE?</td><td>query only
NR1,NR1</td><td>Query the width and height (in pixels) of the selected display window.</td></tr>
<tr><td>DISPlay:WINDow[1|2|10]:GEOMetry
:URIGHT?</td><td>query only
NR1,NR1</td><td>Query the absolute pixel coordinates of the upper right corner of the selected display window.</td></tr>
<tr><td>DISPlay:WINDow:GRAPhics:BUFFer
[:STATe] <ON|OFF>[1]</td><td>NR1</td><td>Turn on/off buffering of user graphics commands.</td></tr>
<tr><td>DISPlay:WINDow[1|2|10]:GRAPhics[2]
:CIRCle <num></td><td>command only</td><td>Draw a circle of the specified Y-axis radius centered at the current pen location — num is the diameter in pixels.[3]</td></tr>
<tr><td>DISPlay:WINDow[1|2|10]:GRAPhics[2]
:CLEar</td><td>command only</td><td>Clear the user graphics and graphics buffer for the specified window.</td></tr>
<tr><td>DISPlay:WINDow[1|2|10]:GRAPhics[2]
:COLor <num></td><td>NR1</td><td>Set the color of the user graphics pen — choose from 0 for erase, 1 for bright, and 2 for dim.</td></tr>
<tr><td>DISPlay:WINDow[1|2|10]:GRAPhics[2]
[:DRAW] <num1>,<num2></td><td>command only</td><td>Draw a line from the current pen position to the specified new pen position — num1 and num2 are the new absolute X and Y coordinates in pixels.[3]</td></tr>
<tr><td>DISPlay:WINDow[1|2|10]:GRAPhics[2]
:LABel <string></td><td>command only</td><td>Draw a label with the lower left corner at the current pen location.[3]</td></tr>
<tr><td>DISPlay:WINDow[1|2|10]:GRAPhics[2]
:LABel:FONT <char></td><td>CHAR</td><td>Select the user graphics label font — choose from SMALl|HSMall|NORMal|
HNORmal|BOLD|HBOLd|SLANt|HSLant.</td></tr>
<tr><td>DISPlay:WINDow[1|2|10]:GRAPhics[2]
:MOVE <num1>,<num2></td><td>NR1,NR1</td><td>Move the pen to the specified new pen position — num1 and num2 are the new absolute X and Y coordinates in pixels.[3]</td></tr>
<tr><td>DISPlay:WINDow[1|2|10]:GRAPhics[2]
:RECTangle <num1>,<num2></td><td>command only</td><td>Draw a rectangle of the specified size with lower left corner at the current pen position — num1 and num2 are the width and height in pixels.[3]</td></tr>
<tr><td>DISPlay:WINDow[1|2|10]:GRAPhics[2]
:STATe?</td><td>query only
NR1</td><td>Query whether a window is enabled for user graphics commands.</td></tr>
<tr><td>DISPlay:WINDow[1|2]:TRACe:
GRATicule:GRID[:STATe] <ON|OFF>[1]</td><td>NR1</td><td>Turn on/off display graticule.</td></tr>
</table>

1 Binary parameters accept the values of **1** (on) and **0** (off) in addition to **ON** and **OFF**.

2 Refer to Chapter 7, "Using Graphics," for more information.

3 Refer to Chapter 7, and to the example program titled "GRAPHICS" in Chapter 8 for more information.

DISPlay (continued)

SUBSYSTEM COMMANDS	FORM	DESCRIPTION
DISPlay:WINDow[1|2]:TRACe[1|2] [:STATe] <ON|OFF>[1]	NR1	Turn on/off the display of trace and memory data from the specified channel.
DISPlay:WINDow[1|2]:TRACe:Y [:SCALe]:AUTO ONCE	command only	Scale the measurement data for a best fit display.
DISPlay:WINDow[1|2]:TRACe:Y [:SCALe]:PDIVision <num>[2]	NR3	Specify the height (dB or units per division) of each vertical division of the specified channel.
DISPlay:WINDow[1|2]:TRACe:Y [:SCALe]:RLEVel <num>[2]	NR3	Specify the value for the Y-axis reference position for the specified channel.
DISPlay:WINDow[1|2]:TRACe:Y [:SCALe]:RPOSition <num>	NR3	Specify the Y-axis reference position for the specified channel.

1 Binary parameters accept the values of **1** (on) and **0** (off) in addition to **ON** and **OFF**.

2 Numeric parameters may include an appropriate suffix; if no suffix is included, the default (**HZ** for frequency or **S** for time) is assumed.

FORMat

SUBSYSTEM COMMANDS	FORM	DESCRIPTION
FORMat:BORDer <char>	CHAR	Specify the byte order used for HPIB data transfer — choose NORMal or SWAPped (for PC-compatible systems).
FORMat[:DATA] <char>[,<num>]	CHAR[,NR1]	Specify the data format for use during data transfer — choose from REAL,64|REAL,32|INTeger,16| ASCii.

HCOPy

<table>
<tr><th>SUBSYSTEM COMMANDS</th><th>FORM</th><th>DESCRIPTION</th></tr>
<tr><td>HCOPy:ABORt</td><td>command only</td><td>Aborts any hardcopy currently in progress.</td></tr>
<tr><td>HCOPy:DEVice[1|2]:COLor <ON|OFF>[1]</td><td>NR1</td><td>Select monochrome OFF or color ON mode for hardcopy output — use device 1 for printers and 2 for plotters.</td></tr>
<tr><td>HCOPy:DEVice:LANGuage <char></td><td>CHAR</td><td>Select the language for hardcopy output — choose from PCL|HPGL|EPSon|IBM|PCX[2]</td></tr>
<tr><td>HCOPy:DEVice:MODE <char></td><td>CHAR</td><td>Select the graph and/or table(s) to appear on a hardcopy plot — choose from GMARker|GRAPh|ISETtings|MARKer|TABLe.</td></tr>
<tr><td>HCOPy:DEVice:PORT <char></td><td>CHAR</td><td>Select the communications port for hardcopy output — choose from CENTronics|SERial|GPIB|MMEMory.</td></tr>
<tr><td>HCOPy:DEVice:RESolution <num></td><td>NR1</td><td>Set the printer resolution in millimeters.</td></tr>
<tr><td>HCOPy[:IMMediate]</td><td>command only</td><td>Initiates a hardcopy output (print or plot).</td></tr>
<tr><td>HCOPy:ITEM:ANNotation:STATe <ON|OFF>[1]</td><td>NR1</td><td>Turns on/off channel and frequency annotation as part of hardcopy output.</td></tr>
<tr><td>HCOPy:ITEM[1|2]:FFEed:STATe <ON|OFF>[1]</td><td>NR1</td><td>Turns on/off an automatic form feed at the completion of hardcopy output — use item 1 for printers and 2 for plotters.</td></tr>
<tr><td>HCOPy:ITEM:GRATicule:STATe <ON|OFF>[1]</td><td>NR1</td><td>Turns on/off graticule as part of hardcopy output.</td></tr>
<tr><td>HCOPy:ITEM:MARKer:STATe <ON|OFF>[1]</td><td>NR1</td><td>Turns on/off marker symbols as part of hardcopy output.</td></tr>
<tr><td>HCOPy:ITEM:TITLe:STATe <ON|OFF>[1]</td><td>NR1</td><td>Turns on/off title and clock lines as part of hardcopy output.</td></tr>
<tr><td>HCOPy:ITEM:TRACe:STATe <ON|OFF>[1]</td><td>NR1</td><td>Turns on/off trace data as part of hardcopy output.</td></tr>
<tr><td>HCOPy:PAGE:MARGin:TOP <num></td><td>NR2</td><td>Sets the top margin (for printer output) in millimeters.</td></tr>
<tr><td>HCOPy:PAGE:MARGin:LEFT <num></td><td>NR2</td><td>Sets the left margin (for printer output) in millimeters.</td></tr>
<tr><td>HCOPy:PAGE:ORIentation <char></td><td>CHAR</td><td>Sets printer output page orientation — choose PORTrait or LANDscape.</td></tr>
<tr><td>HCOPy:PAGE:WIDTh <num></td><td>NR2</td><td>Sets the print width (for printer output) in millimeters.</td></tr>
</table>

1 Binary parameters accept the values of **1** (on) and **0** (off) in addition to **ON** and **OFF**.

2 **EPSon** and **IBM** produce the same results.

INITiate

SUBSYSTEM COMMANDS	FORM	DESCRIPTION
INITiate[1|2]:CONTinuous <ON|OFF>[1]	NR1	Set the trigger system to continuously sweep or to stop sweeping.
INITiate[1|2][:IMMediate]	command only	Initiate a new measurement sweep.

1 Binary parameters accept the values of **1** (on) and **0** (off) in addition to **ON** and **OFF**.

MMEMory

SUBSYSTEM COMMANDS	FORM	DESCRIPTION
MMEMory:CDIRectory <string>	STRING	Change the current directory on a DOS formatted disk — new directory must be on the same mass storage device.
MMEMory:COPY <string1>,<string2>[1]	command only	Copy a file — string1 is the source file, string2 is the destination file.
MMEMory:DELete <string>[1]	command only	Delete a file — string is the filename.
MMEMory:INITialize [<string>[,<char>[,<num>]]]	command only	Format a disk — string is the mass storage device MEM: (internal memory), INT: (internal floppy disk drive) or EXT:. Choose the disk format char from DOS or LIF, and the interleave factor num.
MMEMory:LOAD:STATe 1, <string>[1,2]	command only	Recall an instrument state from mass storage — string is the filename.
MMEMory:MDIRectory <string>	command only	Make a new directory on a DOS formatted disk.
MMEMory:MSIS <string>	STRING	Select a mass storage device — choose MEM: (internal memory), INT: (internal floppy disk drive) or EXT:.
MMEMory:MOVE <string1>,<string2>[1]	command only	Move or rename a file — string1 is the source (or old) filename and string2 is the destination (or new) filename.
MMEMory:RDIRectory <string>	command only	Delete a directory from a DOS formatted disk.
MMEMory:STORe:STATe 1, <string>[1,2]	command only	Save an instrument state to mass storage — string is the filename.
MMEMory:STORe:STATe:ISTate <ON|OFF>[3]	NR1	Turn on/off the instrument state — part of the definition of a saved file.
MMEMory:STORe:STATe:CORRection <ON|OFF>[3]	NR1	Turn on/off the calibration — part of the definition of a saved file.
MMEMory:STORe:STATe:TRACe <ON|OFF>[3]	NR1	Turn on/off the data trace — part of the definition of a saved file.
MMEMory:STORe:TRACe <char>,<string>[1,2]	command only	Stores an ASCII list of trace and frequency values to a file — char is the formatted data trace CH<1|2>FDATA and string is the filename.

1 Filenames may include the mass storage device — **MEM:** (internal non-volatile memory), **RAM:** (internal volatile memory), **INT:** (internal 3.5" disk drive) or **EXT:**. Wildcards **?** and * may be used.

2 Refer to "Measurement Setup and Control" in Chapter 7 of the *User's Guide* for more information on using this command.

3 Binary parameters accept the values of **1** (on) and **0** (off) in addition to **ON** and **OFF**.

MMEMory (continued)

SUBSYSTEM COMMANDS	FORM	DESCRIPTION
MMEMory:TRANsfer:BDAT <string>[1][,<block>][2]	STRING, BLOCK	Copy a file to or from the analyzer's disk drive.[3]
MMEMory:TRANsfer[:HFS] <string>[1][,<block>][2]	STRING,	Copy a file to or from the analyzer's disk drive.

1 Filenames may include the mass storage device — **MEM:** (internal non-volatile memory), **RAM:** (internal volatile memory), **INT:** (internal 3.5" disk drive) or **EXT:**. Wildcards **?** and ***** may be used.

2 Refer to Chapter 8, "Example Programs" for more information on using this command.

3 Refer to the example programs PUTFILE and GETFILE in Chapter 8.

OUTPut

SUBSYSTEM COMMANDS	FORM	DESCRIPTION
OUTPut[:STATe] <ON|OFF>[1]	NR1	Turn on/off RF power from the source.

1 Binary parameters accept the values of **1** (on) and **0** (off) in addition to **ON** and **OFF**.

POWer

SUBSYSTEM COMMANDS	FORM	DESCRIPTION
POWer[1|2]:MODE <char>	CHAR	Specify either frequency sweep (FIXed) or power sweep (SWEep).

PROGram

SUBSYSTEM COMMANDS	FORM	DESCRIPTION
PROGram[1]:CATalog?	query only STRING	List the names of the defined IBASIC programs — response is "PROG" (if a program is present) or the null string ("").
PROGram[1][:SELected][2]:DEFine <block>	BLOCK	Download an IBASIC program from an external controller.
PROGram[1][:SELected][2]:DELete:ALL	command only	Delete all IBASIC programs from the program buffer — equivalent to an HP BASIC SCRATCH A command.
PROGram[1][:SELected][2]:DELete [:SELected]	command only	Delete the active IBASIC program — equivalent to an HP BASIC SCRATCH A command.
PROGram[1][:SELected][2]:EXECute <string>	command only	Execute an IBASIC command.
PROGram[1][:SELected][2]:MALLocate <num>	NR1	Allocate memory space for IBASIC programs — choose from a real number between 2048 and 4000000 bytes.
PROGram[1][:SELected][2]:NAME 'PROG'	STRING	Select the IBASIC program in the program buffer to be active.
PROGram[1][:SELected][2]:NUMBer <string>,<data>[3]	BLOCK or NR3[3]	Load a new value for a numeric variable string in the active IBASIC program — num is the new value.
PROGram[1][:SELected][2]:STATe <char>	CHAR	Select the state of the active IBASIC program — choose from STOP|PAUSe|RUN|CONTinue.
PROGram[1][:SELected][2]:STRing <string1>,<string2>	STRING	Load a new value for a string variable string1 in the active IBASIC program — string2 is the new value.
PROGram[1][:SELected][2]:WAIT	NR1	Wait until the IBASIC program completes.

1 Commands in the **PROGram** subsystem are only available when the HP Instrument BASIC (IBASIC) option is installed (option 1C2). They allow you to generate and control IBASIC programs in the analyzer.

2 Commands grouped under the **SELected** mnemonic in the **PROGram** subsystem operate on the active program buffer.

3 The parameter type of the data is determined by the format selected — **FORMat REAL** uses BLOCK data, **FORMat ASCii** uses NR3 data separated by commas.

SENSe[1|2]

SUBSYSTEM COMMANDS	FORM	DESCRIPTION
SENSe[1|2]:AVERage:CLEar	command only	Re-start the trace averaging function.
SENSe[1|2]:AVERage:COUNt <num>	NR1	Specify a count or weighting factor for the averaged measurement data.
SENSe[1|2]:AVERage[:STATe] <ON|OFF>[1]	NR1	Turn on/off the trace averaging function.
SENSe[1|2]:BWIDth[:RESolution] <num> HZ	NR2	Specify the bandwidth of the IF receiver (fine, narrow, medium or wide) to be used in the measurement — choose 15 (fine) 250 (narrow) 3700 (medium) or 6500 (wide).
SENSe[1|2]:CORRection:CAPacitance :CONNector <num>	NR3	Select connector compensating capacitance value. (For use with structural return loss measurements on analyzers with Option 100 only.)
SENSe[1|2]:CORRection:COLLect:ABORt	command only	Aborts the calibration that is currently in progress.
SENSe[1|2]:CORRection:COLLect [:ACQuire] <char>	command only	Measure a calibration standard — select from STANdard1|STANdard2|STANdard3.
SENSe[1|2]:CORRection:COLLect :CKIT[:SELect]	STRING	Select Cal Kit Choose from one of the following strings: 'COAX,7MM,TYPE-N,50,FEMALE' 'COAX,7MM,TYPE-N,50,MALE' 'COAX,3.5,APC-3.5,50,IMPLIED' 'USER,IMPLIED,IMPLIED,IMPLIED, IMPLIED' 'COAX,7MM,TYPE-F,75,IMPLIED'[2] 'COAX,7MM,TYPE-N,75,FEMALE' 'COAX,7MM,TYPE-N,75,MALE'
SENSe[1|2]:CORRection:COLLect :IState[:AUTO] <ON|OFF>[1]	NR1	Select the instrument state for calibration — choose Full Band (ON) or User Defined (OFF).
SENSe[1|2]:CORRection:COLLect :METHod <char>	command only	Select the type of calibration — choose from TRAN1|TRAN2|REFL3|NONE.
SENSe[1|2]:CORRection:COLLect:SAVE	command only	Complete and save current calibration.
SENSe[1|2]:CORRection:CSET [:SELect] DEFault	command only	Restore the "factory" default calibration for the current measurement and channel.
SENSe[1|2]:CORRection:CSET [:SELect]?	query only CHAR	Query the current calibration type — returns DEF (factory default), FULL (full band) or USER (user defined).

1 Binary parameters accept the values of **1** (on) and **0** (off) in addition to **ON** and **OFF**.

2 Implemented in firmware revisions B.03.01 and above.

SENSe[1|2] (continued)

SUBSYSTEM COMMANDS	FORM	DESCRIPTION
SENSe[1|2]:CORRection:EDELay:TIME <num>[1]	NR3	Specifies the electrical delay in seconds.
SENSe[1|2]:CORRection:EXTension [:STATe] <ON|OFF>[2]	NR1	Enables port extensions.
SENSe[1|2]:CORRection:EXTension :REFLection[:TIME] <num>[1]	NR3	Specifies the port extension at the reflection port, in seconds.
SENSe[1|2]:CORRection:EXTension :TRANsmission[:T IME] <num>[1]	NR3	Specifies the port extension at the transmission port, in seconds.
SENSe[1|2]:CORRection:IMPedance :INPut:MAGNitude <num>[1]	NR3	Specifies the reference impedance for the Smith chart display. The default is the analyzer's system impedance.
SENSe[1|2]:CORRection:LENGth:COAX <num>	NR2	Specifies the length of cable to be calibrated, in feet or meters. (For use with fault location measurements on analyzers with Option 100 only.)
SENSe[1|2]:CORRection:LENGth :CONNector <num>	NR2	Specifies the length of an interface connector, in mm or inches. (For use with structural return loss measure;ments on analyzers with Option 100 only.)
SENSe[1|2]:CORRection:LOSS:COAX <num>	NR2	Specifies the loss of a cable under test, in dB/100 ft. (For use with fault location measurements on analyzers with Option 100 only.)
SENSe[1|2]:CORRection:OFFSet:PHASe	NR3	Specifies the phase offset.

1 Numeric parameters may include an appropriate suffix; if no suffix is included, the default (**HZ** for frequency or **S** for time) is assumed.

2 Binary parameters accept the values of **1** (on) and **0** (off) in addition to **ON** and **OFF**.

SENSe[1|2] (continued)

SUBSYSTEM COMMANDS	FORM	DESCRIPTION
SENSe[1|2]:CORRection:MODel:CONNector [:IMMediate]	command only	Measure the cable connector and determine the optimum values for connector length and connector capacitance. (For use with structural return loss measurements on analyzers with Option 100 only.)
SENSe[1|2]:CORRection:PEAK:COAX [:STATe] <ON|OFF>[1]	NR1	Turns multi-peak correction on or off. (For use with fault location measurements on analyzers with Option 100 only.)
SENSe[1|2]:CORRection:RVELocity:COAX <num>[2]	NR3	Specifies the velocity factor to be used when displaying the distance for electrical length and port extensions. 1.0 = the speed of light.
SENSe[1|2]:CORRection:RVELocity [:IMMediate]	command only	Measure the cable and determine the optimum values for cable loss and velocity factor. (For use with fault location measurements on analyzers with Option 100 only.)
SENSe[1|2]:CORRection:THReshold :COAX <num>	NR2	Selects multi-peak threshold value, in dB. (For use with fault location measurements on analyzers with Option 100 only.)
SENSe[1|2]:COUPle <char>	CHAR	Turn on/off the alternate sweep mode — choose ALL (coupled sweep) or NONE (alternate sweep).
SENSe[1|2]:DETector[:FUNCtion] <char>	CHAR	Specify which detection mode is used to make the measurement — choose BBANd (broadband) or NBANd (narrowband).
SENSe[1|2]:DISTance:STARt <num>	NR3	Set the start distance for a fault location measurement, in feet or meters. (For use with fault location measurements on analyzers with Option 100 only.)
SENSe[1|2]:DISTance:STOP <num>	NR3	Set the stop distance for a fault location measurement, in feet or meters. (For use with fault location measurements on analyzers with Option 100 only.)
SENSe[1|2]:DISTance:UNITs <char>	CHAR	Specifies distance units. Choose METers or FEET. (For use with fault location measurements on analyzers with Option 100 only.)
SENSe[1|2]:FREQuency:CENTer <num>[2]	NR3	Set the center frequency of the RF source.
SENSe[1|2]:FREQuency:MODE <char>	CHAR	Set the fault location measurement to CENTer (bandpass) or LOWPass.(For use with fault location measurements on analyzers with Option 100 only.)

1 Binary parameters accept the values of **1** (on) and **0** (off) in addition to **ON** and **OFF**.

2 Numeric parameters may include an appropriate suffix; if no suffix is included the default (**HZ** for frequency or **S** for time) is assumed.

SENSe[1|2] (continued)

SUBSYSTEM COMMANDS	FORM	DESCRIPTION
SENSe[1|2]:FREQuency:SPAN <num>[1]	NR3	Set the frequency span of the RF source.
SENSe[1|2]:FREQuency:SPAN :MAXimum <num>[1]	NR3	Set the maximum frequency span of the RF source for bandpass fault location measurements. (For use with fault location measurements on analyzers with Option 100 only.)
SENSe[1|2]:FREQuency:STARt <num>[1]	NR3	Set the start frequency of the RF source.
SENSe[1|2]:FREQuency:STOP <num>[1]	NR3	Set the stop frequency of the RF source.
SENSe[1|2]:FREQuency:ZSTOp <num>[1]	NR3	Set the Z cutoff frequency for cable impedance calculations. (For use with structural return loss measurements on analyzers with Option 100 only.)
SENSe[1|2]:FUNCtion?	query only STRING	Query the measurement function — returns one of the 'XFR:POW . . .' or 'XFR:POW:RAT ... ' strings described below.
SENSe[1|2]:FUNCtion:SRL:IMPedance <num>	NR2	Set the cable impedance. (For use with structural return loss measurements on analyzers with Option 100 only.)
SENSe[1|2]:FUNCtion:SRL:MODE <char>	CHAR	Set the auto z function to AUTO or MANual. (For use with structural return loss measurements on analyzers with Option 100 only.)
SENSe[1|2]:FUNCtion:SRL:SCAN [:IMMediate]	command only	Start a cable scan. (For use with structural return loss measurements on analyzers with Option 100 only.)
SENSe[1|2]:FUNCtion 'XFRequency :POWer <num>'	command only	Specify that the receiver will measure the power into a the single channel — choose from channels 0 (R), 1 (A), 2 (B), 11 (Ext X) or 12 (Ext Y).
SENSe[1|2]:FUNCtion 'XFRequency :POWer:RATio <num>,<num>'	command only	Specify that the receiver will measure a ratio of the power into the two selected channels — choose from ratios 1,0 (A/R), 2,0 (B/R), 12,0 (Ext Y/R), 11,12 (Ext X/Ext Y), 12,11 (Ext Y/Ext X), or 12,11 (AM Delay).
SENSe[1|2]:ROSCillator:SOURce <char>	CHAR	Specify the source of the reference oscillator — select INTernal or EXTernal.
SENSe[1|2]:STATe <ON|OFF>[2]	NR1	Turn on/off the specified channel.

1 Numeric parameters may include an appropriate suffix; if no suffix is included, the default (**HZ** for frequency or **S** for time) is assumed.

2 Binary parameters accept the values of **1** (on) and **0** (off) in addition to **ON** and **OFF**.

SENSe[1|2] (continued) (continued)

SUBSYSTEM COMMANDS	FORM	DESCRIPTION
SENSe[1|2]:SWEep:POINts <num>	NR1	Set the number of data points for the measurement — choose from 3|5|11|21|51|101|201|401|801|1601.
SENSe[1|2]:SWEep:TIME <num>[1]	NR3	Set the sweep time.
SENSe[1|2]:SWEep:TIME:AUTO <ON|OFF|ONCE>[2]	NR1	Turn on/off the automatic sweep time function.
SENSe:SWEep:TRIGger:SOURce <char>	CHAR	Set the trigger source for each point in a sweep — choose IMMediate or EXTernal (used in conjunction with TRIGger[:SEQuence]:SOURce).
SENSe:WINDow[:TYPE] <char>	CHAR	Set the window selection for fault location measurements. Choose from RECTangular (Minimum), HAMMing (Medium), or KBESsel (Maximum). (For use with fault location measurements on analyzers with Option 100 only.)

1 Numeric parameters may include an appropriate suffix; if no suffix is included, the default (**HZ** for frequency or **S** for time) is assumed.

2 Binary parameters accept the values of **1** (on) and **0** (off) in addition to **ON** and **OFF**.

SOURce

SUBSYSTEM COMMANDS	FORM	DESCRIPTION
SOURce[1|2]:POWer[:LEVel][:IMMediate][:AMPLitude] <num>[1]	NR3	Set the RF power output from the source.
SOURce[1|2]:POWer:RANGe <char>	CHAR	Specifies the power sweep range. Choose from ATTen0 |ATTen10|ATTen20|ATTen30|ATTen40 |ATTen50|ATTen60.
SOURce[1|2]:POWer:STARt <num>	NR3	Sets the power sweep start power.
SOURce[1|2]:POWer:STOP <num>	NR3	Sets the power sweep stop power.

1 Numeric parameters may include an appropriate suffix; if no suffix is included, the default (**HZ** for frequency or **S** for time) is assumed.

STATus

SUBSYSTEM COMMANDS	FORM	DESCRIPTION
STATus:DEVice:CONDition?	query only NR1	Read and clear the Device Status condition register[1].
STATus:DEVice:ENABle <num>	NR1	Set and query bits in the Device Status enable register.[2]
STATus:DEVice[:EVENt]?	query only NR1	Read and clear the Device Status event register.[1]
STATus:DEVice:NTRansition <num>	NR1	Set and query bits in the Device Status negative transition register.[2]
STATus:DEVice:PTRansition <num>	NR1	Set and query bits in the Device Status positive transition register.[2]
STATus:OPERation:AVERaging :CONDition?	query only NR1	Read the Averaging status condition register.[1]
STATus:OPERation:AVERaging:ENABle <num>	NR1	Set and query bits in the Averaging status enable register.[2]
STATus:OPERation:AVERaging[:EVENt]?	query only NR1	Read and clear the Averaging status event register.[1]
STATus:OPERation:AVERaging :NTRansition <num>	NR1	Set and query bits in the Averaging status negative transition register.[2]
STATus:OPERation:AVERaging :PTRansition <num>	NR1	Set and query bits in the Averaging status positive transition register.[2]
STATus:OPERation:CONDition?	query only NR1	Read the Operational Status condition register.[1]
STATus:OPERation:ENABle <num>	NR1	Set and query bits in the Operational Status enable register.[2]
STATus:OPERation[:EVENt]?	query only NR1	Read and clear the Operational Status event register.[1]
STATus:OPERation:MEASuring :CONDition?	query only NR1	Read the Measuring status condition register.[1]
STATus:OPERation:MEASuring:ENABle <num>	NR1	Set and query bits in the Measuring status enable register.[2]

1 Returns the sum of the decimal weights (2^n where n is the bit number) of all bits currently set. For more information on using the status registers refer to Chapter 5, "Using Status Registers."

2 **num** is the sum of the decimal weights of all bits to be set.

STATus (continued)

SUBSYSTEM COMMANDS	FORM	DESCRIPTION
STATus:OPERation:MEASuring[:EVENt]?	query only NR1	Read and clear the Measuring status event register.[1]
STATus:OPERation:MEASuring :NTRansition <num>	NR1	Set and query bits in the Measuring status negative transition register.[2]
STATus:OPERation:MEASuring :PTRansition <num>	NR1	Set and query bits in the Measuring status positive transition register.[2]
STATus:OPERation:NTRansition <num>	NR1	Set and query bits in the Operational Status negative transition register.[2]
STATus:OPERation:PTRansition <num>	NR1	Set and query bits in the Operational Status positive transition register.[2]
STATus:PRESet	command only	Set bits in most enable and transition registers to their default state.
STATus:QUEStionable:CONDition?	query only NR1	Read and clear the Questionable Status condition register.[1]
STATus:QUEStionable:ENABle <num>	NR1	Set and query bits in the Questionable Status enable register.[2]
STATus:QUEStionable[:EVENt]?	query only NR1	Read and clear the Questionable Status event register.[1]
STATus:QUEStionable:LIMit :CONDition?	query only NR1	Read and clear the Limit Fail condition register.[1]
STATus:QUEStionable:LIMit:ENABle <num>	NR1	Set and query bits in the Limit Fail enable register.[2]
STATus:QUEStionable:LIMit[:EVENt]?	query only NR1	Read and clear the Limit Fail event register.[1]
STATus:QUEStionable:LIMit :NTRansition <num>	NR1	Set and query bits in the Limit Fail negative transition register.[2]
STATus:QUEStionable:LIMit :PTRansition <num>	NR1	Set and query bits in the Limit Fail positive transition register.[2]
STATus:QUEStionable:NTRansition <num>	NR1	Set and query bits in the Questionable Status negative transition register.[2]

1 Returns the sum of the decimal weights (2^n where n is the bit number) of all bits currently set. For more information on using the status registers refer to Chapter 5, "Using Status Registers."

2 **num** is the sum of the decimal weights of all bits to be set.

STATus (continued)

SUBSYSTEM COMMANDS	FORM	DESCRIPTION
`STATus:QUEStionable:PTRansition <num>`	NR1	Set and query bits in the Questionable Status positive transition register.[1]

1 **num** is the sum of the decimal weights of all bits to be set.

SYSTem

SUBSYSTEM COMMANDS	FORM	DESCRIPTION
SYStem:BEEPer[:IMMediate] [<freq>[,<dur>[,<vol>]]] [1]	NR3, NR3, NR3	Instructs the analyzer to beep. Arguments are frequency (Hz), duration (seconds), and volume (0 to 1).
SYSTem:BEEPer:VOLume <num>	NR2	Set the volume of the beeper — num is a number between 0 for 0% and 1 for 100%.
SYSTem:COMMunicate:GPIB:CONTroller [:STATe] <ON|OFF> [2,3]	NR1	Makes the HP 8711 the system controller.
SYSTem:COMMunicate:GPIB:ECHO <ON|OFF> [2]	NR1	Turn on/off HP-IB mnemonic echo.
SYSTem:COMMunicate:GPIB:HCOPy :ADDRess <num>	NR1	Set the address of an HP-IB printer or plotter for hardcopy output — num must be an integer between 0 and 30.
SYSTem:COMMunicate:GPIB:MMEMory :ADDRess <num>	NR1	Set the HP-IB address of an external disk drive — num must be an integer between 0 and 30.
SYSTem:COMMunicate:GPIB:MMEMory :UNIT <num>	NR1	Set the unit number of an external disk drive.
SYSTem:COMMunicate:GPIB:MMEMory :VOLume <num>	NR1	Set the volume number of an external disk drive.
SYSTem:COMMunicate:GPIB[:SELF] :ADDRess <num> [4]	NR1	Set the HP 8711's HP-IB address — num must be an integer between 0 and 30.
SYSTem:COMMunicate:SERial:TRANsmit :BAUD <num>	NR1	Set the baud rate for hardcopy output to a device on the serial port — choose from 1200|2400|4800|9600|19200.
SYSTem:COMMunicate:SERial:TRANsmit :HANDshake <char>	CHAR	Set the handshake for communication to a hardcopy device on the serial port — choose XON or DTR.
SYSTem:COMMunicate:TTL:USER:FEED:KEY [:STATe] <ON|OFF> [2]	NR1	Enable/disable softkey auto-step function to work with fast recall (all models) or with user begin function (option 1C2 only).

1 <**freq**>, <**dur**>, and <**vol**> are optional <**num**> parameters.

2 Binary parameters accept the values of **1** (on) and **0** (off) in addition to **ON** and **OFF**.

3 For use with IBASIC — this command cannot be executed from an external controller.

4 A delay of 5 seconds is required before a command is sent to the new address.

SYSTem (continued)

SUBSYSTEM COMMANDS	FORM	DESCRIPTION
SYSTem:DATE <num1>,<num2>,<num3>	NR1,NR1, NR1	Set the year (num1), month (num2) and day (num3) of the real time clock.
SYSTem:ERRor?[1]	query only NR1,STRING	Query the error queue — returns the error number and message.
SYSTem:KEY:MASK?	query only NR1	Query the mask (shift, ctrl, alt) associated with a keypress on an external keyboard.
SYSTem:KEY:QUEue:CLEar	command only	Clears the key queue.
SYSTem:KEY:QUEue:COUNt?	query only NR1	Query the number of key codes in the queue.
SYSTem:KEY:QUEue:MAXimum?	query only NR1	Query the size of the key queue (the maximum number of key codes it can hold).
SYSTem:KEY:QUEue[:STATe] <ON|OFF>[2]	NR1	Turn on/off the key queue.
SYSTem:KEY:TYPE?	query only CHAR	Query the type of key that was pressed — returns NONE, RPG, KEY (front panel key) or ASC (external keyboard).
SYSTem:KEY:USER	command only	Sets the User Request bit of the Standard Event Status Register.
SYSTem:KEY[:VALue]?	query only NR1	Query the key code value for the last key pressed — RPG type returns the knob count, positive for clockwise rotation, KEY type returns the front panel keycode,[3] and ASC type returns the ASCII code number.
SYSTem:PRESet	command only	Perform a system preset — this is the same as the front panel (PRESET) key.
SYSTem:SET <block>	command only	Send a learn string (obtained using *LRN?) to the analyzer — this command is included in the learn string.
SYSTem:SET:LRN? [<USER>][4]	BLOCK	Query or set the instrument state.
SYSTem:SET:LRNLong? [<USER>][4]	BLOCK	Query or set the instrument state, data, and calibration. Similar to save/recall.
SYSTem:TIME <num1>,<num2>,<num3>	NR1,NR1, NR1	Set the hour (num1), minute (num2) and second (num3) of the real time clock.

1 For more information on errors, refer to Chapter 14, "SCPI Error Messages."

2 Binary parameters accept the values of **1** (on) and **0** (off) in addition to **ON** and **OFF**.

3 A list of the analyzer's front panel keycodes is provided in Chapter 9.

4 Refer to "Measurement Setup and Control" in Chapter 7 of the *User's Guide* for more information on using this command.

SYSTem (continued) (continued)

SUBSYSTEM COMMANDS	FORM	DESCRIPTION
SYSTem:VERSion?	query only NR2	Query the SCPI version of the analyzer. See *IDN? to query the firmware revision.

TEST

SUBSYSTEM COMMANDS	FORM	DESCRIPTION
TEST:RESult?	query only CHAR	Query the result of the selected adjustment or self-test — the response will be NULL|PASS|FAIL.
TEST:SELect <num>	NR1	Select the adjustment or self-test to execute.
TEST:STATe <char>	CHAR	Select the state of the active adjustment or self-test — choose from RUN|CONTinue|STOP for the command. Query returns NULL|RUN|PAUS|DONE.
TEST:VALue <num>	NR1	Set or query a value for an adjustment or self-test.

TRACe

SUBSYSTEM COMMANDS	FORM	DESCRIPTION
TRACe[:DATA]? <char>	query only BLOCK or NR3[1]	Query trace data — choose from CH<1|2>FDATA formatted data, CH<1|2>FMEM formatted memory, CH<1|2>SDATA unformatted data, CH<1|2>SMEM unformatted memory, CH<1|2><A|B|R>FWD raw data, or CH<1|2>SCORR<1|2|3> correction data. **Note:** See Chapter 6, "Trace Data Transfers," for data array details.
TRACe[:DATA] <char>,<data>	command only	Input trace data — choose from the above list of arrays. The data can be either BLOCK or NR3 type.[1] See Chapter 6 for more information.
TRACe[:DATA] <char1>,<char2>	command only	Move data from one internal array to another — char1 is the target array (CH<1|2>SMEM) while char2 is the source array (CH<1|2>SDATA). Note that the source and target arrays must be from the same measurement channel.

1 The parameter type of the data is determined by the format selected — **FORMat REAL** uses BLOCK data, **FORMat ASCii** uses NR3 data separated by commas.

TRIGger

SUBSYSTEM COMMANDS	FORM	DESCRIPTION
`TRIGger[:SEQuence]:SOURce <char>`	CHAR	Set the source for the sweep trigger signal — choose `IMMediate` or `EXTernal` (used in conjunction with `SENSe:SWEep:TRIGger:SOURce`).

13

SCPI Conformance Information

SCPI Conformance Information

The HP 8711B/12B/13B/14B RF Network Analyzers conform to the 1994.0 version of SCPI.

SCPI Standard Commands

The analyzer implements the following IEEE 488.2 standard commands:

```
*CLS
*ESE
*ESE?
*ESR?
*IDN?
*LRN?
*OPC
*OPC?
*OPT?
*PCB
*PSC
*RST
*SRE
*SRE?
*STB?
*TRG
*TST?
*WAI
```

The analyzer implements the following SCPI 1994.0 standard commands:

```
ABORt

CALCulate[1|2]:DATA?
CALCulate[1|2]:GDAPerture:APERture
CALCulate[1|2]:GDAPerture:SPAN
CALCulate[1|2]:FORMat
CALCulate[1|2]:FORMat?
CALCulate[1|2]:LIMit:STATe
CALCulate[1|2]:LIMit:STATe?
CALCulate[1|2]:MATH[:EXPRession]
CALCulate[1|2]:MATH[:EXPRession]?

CALibration:ZERO:AUTO
CALibration:ZERO:AUTO?

DISPlay:MENU:KEY[1|2| ... 7]
DISPlay:MENU[1|2]:KEY[1|2| ... 7]?
```

(symbol) **indicates HP 8712B/14B only**

```
DISPlay:WINDow[1|2|10]:GEOMetry:LLEFT?
DISPlay:WINDow[1|2|10]:GEOMetry:SIZE?
DISPlay:WINDow[1|2|10]:GEOMetry:URIGHT?
DISPlay:WINDow[1|2|10]:GRAPhics:CLEar
DISPlay:WINDow[1|2|10]:GRAPhics:COLor
DISPlay:WINDow[1|2|10]:GRAPhics:COLor?
DISPlay:WINDow[1|2|10]:GRAPhics[:DRAW]
DISPlay:WINDow[1|2|10]:GRAPhics:LABel
DISPlay:WINDow[1|2|10]:GRAPhics:MOVE
DISPlay:WINDow[1|2|10]:GRAPhics:MOVE?
DISPlay:WINDow[1|2|10]:GRAPhics:STATe?
DISPlay:WINDow[1|2]:TRACe:GRATicule:GRID[:STATe]
DISPlay:WINDow[1|2]:TRACe:GRATicule:GRID[:STATe]?
DISPlay:WINDow[1|2]:TRACe[1|2][:STATe]
DISPlay:WINDow[1|2]:TRACe[1|2][:STATe]?
DISPlay:WINDow[1|2]:TRACe:Y[:SCALe]:AUTO
DISPlay:WINDow[1|2]:TRACe:Y[:SCALe]:PDIVision
DISPlay:WINDow[1|2]:TRACe:Y[:SCALe]:PDIVision?
DISPlay:WINDow[1|2]:TRACe:Y[:SCALe]:RLEVel
DISPlay:WINDow[1|2]:TRACe:Y[:SCALe]:RLEVel?
DISPlay:WINDow[1|2]:TRACe:Y[:SCALe]:RPOSition
DISPlay:WINDow[1|2]:TRACe:Y[:SCALe]:RPOSition?

FORMat:BORDer
FORMat:BORDer?
FORMat[:DATA]
FORMat[:DATA]?

INITiate[1|2]:CONTinuous
INITiate[1|2]:CONTinuous?
INITiate[1|2][:IMMediate]

MMEMory:CDIRectory
MMEMory:CDIRectory?
MMEMory:COPY
MMEMory:DELete
MMEMory:INITialize
MMEMory:LOAD:STATe
MMEMory:MOVE
MMEMory:MSIS
MMEMory:MSIS?
MMEMory:STORe:STATe
```

MMEMory:STORe:TRACe
MMEMory:TRANsfer:BDAT
MMEMory:TRANsfer[:HFS]

OUTPut[:STATe]
OUTPut[:STATe]?

PROGram:CATalog?
PROGram[:SELected]:DEFine
PROGram[:SELected]:DEFine?
PROGram[:SELected]:DELete:ALL
PROGram[:SELected]:DELete[:SELected]
PROGram[:SELected]:EXECute
PROGram[:SELected]:MALLocate
PROGram[:SELected]:MALLocate?
PROGram[:SELected]:NAME
PROGram[:SELected]:NAME?
PROGram[:SELected]:NUMBer
PROGram[:SELected]:NUMBer?
PROGram[:SELected]:STATe
PROGram[:SELected]:STATe?
PROGram[:SELected]:STRing
PROGram[:SELected]:STRing?
PROGram[:SELected]:WAIT
PROGram[:SELected]:WAIT?

SENSe[1|2]:AVERage:COUNt
SENSe[1|2]:AVERage:COUNt?
SENSe[1|2]:AVERage[:STATe]
SENSe[1|2]:AVERage[:STATe]?
SENSe[1|2]:BWIDth[:RESolution]
SENSe[1|2]:BWIDth[:RESolution]?
SENSe[1|2]:CORRection:COLLect[:ACQuire]
SENSe[1|2]:CORRection:COLLect:METHod
SENSe[1|2]:CORRection:COLLect:SAVE
SENSe[1|2]:CORRection:CSET[:SELect]
SENSe[1|2]:CORRection:CSET[:SELect]?
SENSe[1|2]:CORRection:EDELay:TIME
SENSe[1|2]:CORRection:IMPedance:INPut:MAGNitude
SENSe[1|2]:CORRection:OFFSet:PHASe
SENSe[1|2]:CORRection:RVELocity:COAX
SENSe[1|2]:CORRection[:STATe]

```
SENSe[1|2]:CORRection[:STATe]?
SENSe[1|2]:DETector:SHAPe
SENSe[1|2]:FREQuency:CENTer
SENSe[1|2]:FREQuency:CENTer?
SENSe[1|2]:FREQuency:SPAN
SENSe[1|2]:FREQuency:SPAN?
SENSe[1|2]:FREQuency:STARt
SENSe[1|2]:FREQuency:STARt?
SENSe[1|2]:FREQuency:STOP
SENSe[1|2]:FREQuency:STOP?
SENSe[1|2]:FUNCtion
SENSe[1|2]:FUNCtion?
SENSe:ROSCillator:SOURce
SENSe:ROSCillator:SOURce?
SENSe[1|2]:SWEep:POINts
SENSe[1|2]:SWEep:POINts?
SENSe[1|2]:SWEep:TIME
SENSe[1|2]:SWEep:TIME?
SENSe[1|2]:SWEep:TIME:AUTO
SENSe[1|2]:SWEep:TIME:AUTO?

SOURce[1|2]:POWer[:LEVel][:IMMediate][:AMPLitude]
SOURce[1|2]:POWer[:LEVel][:IMMediate][:AMPLitude]?
SOURce[1|2]:POWer:RANGe
SOURce[1|2]:POWer:STARt
SOURce[1|2]:POWer:STOP

STATus:OPERation:CONDition?
STATus:OPERation:ENABle
STATus:OPERation:ENABle?
STATus:OPERation[:EVENt]?
STATus:OPERation:NTRansition
STATus:OPERation:NTRansition?
STATus:OPERation:PTRansition
STATus:OPERation:PTRansition?
STATus:PRESet
STATus:QUEStionable:CONDition?
STATus:QUEStionable:ENABle
STATus:QUEStionable:ENABle?
STATus:QUEStionable[:EVENt]?
STATus:QUEStionable:NTRansition
STATus:QUEStionable:NTRansition?
```

```
STATus:QUEStionable:PTRansition
STATus:QUEStionable:PTRansition?

SYSTem:BEEPer[:IMMediate]?
SYSTem:BEEPer:VOLume
SYSTem:BEEPer:VOLume?
SYSTem:COMMunicate:GPIB[:SELF]:ADDRess
SYSTem:COMMunicate:GPIB[:SELF]:ADDRess?
SYSTem:COMMunicate:SERial:TRANsmit:BAUD
SYSTem:COMMunicate:SERial:TRANsmit:BAUD?
SYSTem:DATE
SYSTem:DATE?
SYSTem:ERRor?
SYSTem:KEY[:VALue]?
SYSTem:PRESet
SYSTem:SET
SYSTem:SET:LRN?
SYSTem:TIME
SYSTem:TIME?
SYSTem:VERSion?

TRACe[:DATA]
TRACe[:DATA]?

TRIGger[:SEQuence]:SOURce
TRIGger[:SEQuence]:SOURce?
```

Instrument Specific Commands

The following are instrument specific commands implemented by the HP 8711B/12B/13B/14B RF Network Analyzers which are not part of the present SCPI 1992.0 definition.

```
CALCulate[1|2]:LIMit:DISPlay
CALCulate[1|2]:LIMit:DISPlay?
CALCulate[1|2]:LIMit:MARKer:FLATness:MAXimum
CALCulate[1|2]:LIMit:MARKer:FLATness:MINimum
CALCulate[1|2]:LIMit:MARKer:FLATness[:STATe]
CALCulate[1|2]:LIMit:MARKer:STATistic:MEAN:MAXimum
CALCulate[1|2]:LIMit:MARKer:STATistic:MEAN:MINimum
CALCulate[1|2]:LIMit:MARKer:STATistic:MEAN[:STATe]
CALCulate[1|2]:LIMit:MARKer:STATistic:PEAK:MAXimum
CALCulate[1|2]:LIMit:MARKer:STATistic:PEAK:MINimum
CALCulate[1|2]:LIMit:MARKer:STATistic:PEAK[:STATe]
CALCulate[1|2]:LIMit:SEGMent[1|2| ... 12]:AMPLitude:STARt
CALCulate[1|2]:LIMit:SEGMent[1|2| ... 12]:AMPLitude:STARt?
CALCulate[1|2]:LIMit:SEGMent[1|2| ... 12]:AMPLitude:STOP
CALCulate[1|2]:LIMit:SEGMent[1|2| ... 12]:AMPLitude:STOP?
CALCulate[1|2]:LIMit:SEGMent:AOFF
CALCulate[1|2]:LIMit:SEGMent[1|2| ... 12]:FREQuency:STARt
CALCulate[1|2]:LIMit:SEGMent[1|2| ... 12]:FREQuency:STARt?
CALCulate[1|2]:LIMit:SEGMent[1|2| ... 12]:FREQuency:STOP
CALCulate[1|2]:LIMit:SEGMent[1|2| ... 12]:FREQuency:STOP?
CALCulate[1|2]:LIMit:SEGMent[1|2| ... 12]:POWer:STOP
CALCulate[1|2]:LIMit:SEGMent[1|2| ... 12]:POWer:STOP?
CALCulate[1|2]:LIMit:SEGMent[1|2| ... 12]:STATe
CALCulate[1|2]:LIMit:SEGMent[1|2| ... 12]:STATe?
CALCulate[1|2]:LIMit:SEGMent[1|2| ... 12]:TYPE
CALCulate[1|2]:LIMit:SEGMent[1|2| ... 12]:TYPE?
CALCulate[1|2]:MARKer:AOFF
CALCulate[1|2]:MARKer:BWIDth
CALCulate[1|2]:MARKer:BWIDth?
CALCulate[1|2]:MARKer:FUNCtion:RESult?
CALCulate[1|2]:MARKer:FUNCtion[:SELect]
CALCulate[1|2]:MARKer:FUNCtion[:SELect]?
CALCulate[1|2]:MARKer:FUNCtion:TRACking
CALCulate[1|2]:MARKer:FUNCtion:TRACking?
```

```
CALCulate[1|2]:MARKer[1|2| ... 8]:GDELay?
CALCulate[1|2]:MARKer[1|2| ... 8]:MAXimum
CALCulate[1|2]:MARKer[1|2| ... 8]:MAXimum:LEFT
CALCulate[1|2]:MARKer[1|2| ... 8]:MAXimum:RIGHt
CALCulate[1|2]:MARKer[1|2| ... 8]:MINimum
CALCulate[1|2]:MARKer[1|2| ... 8]:MINimum:LEFT
CALCulate[1|2]:MARKer[1|2| ... 8]:MINimum:RIGHt
CALCulate[1|2]:MARKer:MODE
CALCulate[1|2]:MARKer:MODE?
CALCulate[1|2]:MARKer:NOTCh
CALCulate[1|2]:MARKer[1|2| ... 8]:POINt
CALCulate[1|2]:MARKer[1|2| ... 8]:POINt?
CALCulate[1|2]:MARKer:REFerence:X?
CALCulate[1|2]:MARKer:REFerence:Y?
CALCulate[1|2]:MARKer[1|2| ... 8][:STATe]
CALCulate[1|2]:MARKer[1|2| ... 8][:STATe]?
CALCulate[1|2]:MARKer[1|2| ... 8]:TARGet
CALCulate[1|2]:MARKer[1|2| ... 8]:TARGet?
CALCulate[1|2]:MARKer[1|2| ... 8]:X
CALCulate[1|2]:MARKer[1|2| ... 8]:X?
CALCulate[1|2]:MARKer[1|2| ... 8]:X:ABS
CALCulate[1|2]:MARKer[1|2| ... 8]:Y?
CALCulate[1|2]:MARKer[1|2| ... 8]:Y:INDuctance?
CALCulate[1|2]:MARKer[1|2| ... 8]:Y:MAGNitude?
CALCulate[1|2]:MARKer[1|2| ... 8]:Y:PHASe?
CALCulate[1|2]:MARKer[1|2| ... 8]:Y:REACtance?
CALCulate[1|2]:MARKer[1|2| ... 8]:Y:RESistance?

CONFigure
CONFigure?

DIAGnostic:CCONstants:INSTalled?
DIAGnostic:CCONstants:LOAD
DIAGnostic:CCONstants:STORe:DISK
DIAGnostic:CCONstants:STORe:EEPRom
DIAGnostic:DITHer
DIAGnostic:DITHer?
DIAGnostic:SNUMber
DIAGnostic:SNUMber?
DIAGnostic:SPUR:AVOid
DIAGnostic:SPUR:AVOid?
```

```
DISPlay:ANNotation:CHANnel[1|2]:USER:LABel[:DATA]
DISPlay:ANNotation:CHANnel[1|2]:USER[:STATe]
DISPlay:ANNotation:CLOCk:DATE:FORMat
DISPlay:ANNotation:CLOCk:DATE:FORMat?
DISPlay:ANNotation:CLOCk:DATE:MODE
DISPlay:ANNotation:CLOCk:DATE:MODE?
DISPlay:ANNotation:CLOCk:MODE
DISPlay:ANNotation:CLOCk:MODE?
DISPlay:ANNotation:CLOCk:SEConds[:STATe]
DISPlay:ANNotation:CLOCk:SEConds[:STATe]?
DISPlay:ANNotation:FREQuency[1|2]:MODE
DISPlay:ANNotation:FREQuency[1|2]:MODE?
DISPlay:ANNotation:FREQuency:RESolution
DISPlay:ANNotation:FREQuency:RESolution?
DISPlay:ANNotation:FREQuency[1|2]:USER:LABel[:DATA]
DISPlay:ANNotation:FREQuency[1|2]:USER:STARt
DISPlay:ANNotation:FREQuency[1|2]:USER[:STATe]
DISPlay:ANNotation:FREQuency[1|2]:USER:STOP
DISPlay:ANNotation:FREQuency[1|2]:USER:SUFFIX
DISPlay:ANNotation:MARKer[1|2][:STATe]
DISPlay:ANNotation:MARKer[1|2][:STATe]?
DISPlay:ANNotation:MESSage:AOFF
DISPlay:ANNotation:MESSage:CLEar
DISPlay:ANNotation:MESSage[:DATA]?
DISPlay:ANNotation:MESSage:STATe
DISPlay:ANNotation:MESSage:STATe?
DISPlay:ANNotation:TITLe[1|2]:DATA
DISPlay:ANNotation:TITLe[1|2]:DATA?
DISPlay:ANNotation:TITLe[:STATe]
DISPlay:ANNotation:TITLe[:STATe]?
DISPlay:ANNotation:YAXis:MODE
DISPlay:ANNotation:YAXis:MODE?
DISPlay:ANNotation:YAXis[:STATe]
DISPlay:ANNotation:YAXis[:STATe]?
DISPlay:FORMat
DISPlay:FORMat?
DISPlay:MENU:RECall:FAST[:STATe] * DISPlay:PROGram[:MODE]
DISPlay:PROGram[:MODE]?
DISPlay:WINDow:GRAPhics:BUFFer[:STATe]
DISPlay:WINDow:GRAPhics:BUFFer[:STATe]?
```

```
DISPlay:WINDow[1|2|10]:GRAPhics:CIRCle
DISPlay:WINDow[1|2|10]:GRAPhics:LABel:FONT
DISPlay:WINDow[1|2|10]:GRAPhics:LABel:FONT?
DISPlay:WINDow[1|2|10]:GRAPhics:RECTangle

HCOPy:ABORt
HCOPy:DEVice[1|2]:COLor
HCOPy:DEVice[1|2]:COLor?
HCOPy:DEVice:LANGuage
HCOPy:DEVice:LANGuage?
HCOPy:DEVice[1|2]:LANGuage
HCOPy:DEVice:MODE
HCOPy:DEVice:MODE?
HCOPy:DEVice:PORT
HCOPy:DEVice:PORT?
HCOPy:DEVice:RESolution
HCOPy:DEVice:RESolution?
HCOPy[:IMMediate]
HCOPy:ITEM:ANNotation:STATe
HCOPy:ITEM:ANNotation:STATe?
HCOPy:ITEM[1|2]:FFEed:STATe
HCOPy:ITEM[1|2]:FFEed:STATe?
HCOPy:ITEM:GRATicule:STATe
HCOPy:ITEM:GRATicule:STATe?
HCOPy:ITEM:MARKer:STATe
HCOPy:ITEM:MARKer:STATe?
HCOPy:ITEM:TITLe:STATe
HCOPy:ITEM:TITLe:STATe?
HCOPy:ITEM:TRACe:STATe
HCOPy:ITEM:TRACe:STATe?
HCOPy:PAGE:MARGin:LEFT
HCOPy:PAGE:MARGin:LEFT?
HCOPy:PAGE:MARGin:TOP
HCOPy:PAGE:MARGin:TOP?
HCOPy:PAGE:ORIentation
HCOPy:PAGE:ORIentation?
HCOPy:PAGE:WIDTh
HCOPy:PAGE:WIDTh?

INPut:GAIN:AUTO
INPut:GAIN:SETTing
```

MMEMory:MDIRectory
MMEMory:RDIRectory
MMEMory:STORe:STATe:CORRection
MMEMory:STORe:STATe:CORRection?
MMEMory:STORe:STATe:ISTate
MMEMory:STORe:STATe:ISTate?
MMEMory:STORe:STATe:TRACe
MMEMory:STORe:STATe:TRACe?
MMEMory:TRANsfer:BDAT
MMEMory:TRANsfer[:HFS]

POWer\[1|2]:MODE

SENSe[1|2]:AVERage:CLEar
SENSe[1|2]:CORRection:CAPacitance:CONNector (Option 100 only)
SENSe[1|2]:CORRection:CAPacitance:CONNector? (Option 100 only)
SENSe[1|2]:CORRection:COLLect:ABORt
SENSe[1|2]:CORRection:COLLect:CKIT[:SELect]
SENSe[1|2]:CORRection:COLLect:CKIT[:SELect]?
SENSe[1|2]:CORRection:COLLect:ISTate[:AUTO]
SENSe[1|2]:CORRection:COLLect:ISTate[:AUTO]?
⊛SENSe[1|2]:CORRection:EXTension[:STATe]
⊛SENSe[1|2]:CORRection:EXTension:REFLection[:TIME]
⊛SENSe[1|2]:CORRection:EXTension:TRANsmission[:TIME]
SENSe[1|2]:CORRection:LENGth:COAX (Option 100 only)
SENSe[1|2]:CORRection:LENGth:COAX? (Option 100 only)
SENSe[1|2]:CORRection:LENGth:CONNector (Option 100 only)
SENSe[1|2]:CORRection:LENGth:CONNector? (Option 100 only)
SENSe[1|2]:CORRection:LOSS:COAX (Option 100 only)
SENSe[1|2]:CORRection:LOSS:COAX? (Option 100 only)
SENSe[1|2]:CORRection:MODel:CONNector[:IMMediate] (Option 100 only)
SENSe[1|2]:CORRection:PEAK:COAX (Option 100 only)
SENSe[1|2]:CORRection:PEAK:COAX? (Option 100 only)
SENSe[1|2]:CORRection:RVELocity[:IMMediate] (Option 100 only)
SENSe[1|2]:CORRection:THReshold:COAX (Option 100 only)
SENSe[1|2]:CORRection:THReshold:COAX? (Option 100 only)
SENSe:COUPle
SENSe:COUPle?
SENSe[1|2]:DETector[:FUNCtion]
SENSe[1|2]:DETector[:FUNCtion]?

SENSe:DISTance:STARt (Option 100 only)
SENSe:DISTance:STARt? (Option 100 only)
SENSe:DISTance:STOP (Option 100 only)
SENSe:DISTance:STOP? (Option 100 only)
SENSe:DISTance:UNITs (Option 100 only)
SENSe:DISTance:UNITs? (Option 100 only)
SENSe:FREQuency:MODE (Option 100 only)
SENSe:FREQuency:MODE? (Option 100 only)
SENSe:FREQuency:SPAN:MAXimum? (Option 100 only)
SENSe:FREQuency:SPAN:MAXimum (Option 100 only)
SENSe:FREQuency:ZSTop (Option 100 only)
SENSe:FREQuency:ZSTop? (Option 100 only)
SENSe:FUNCtion:SRL:IMPedance (Option 100 only)
SENSe:FUNCtion:SRL:IMPedance? (Option 100 only)
SENSe:FUNCtion:SRL:MODE (Option 100 only)
SENSe:FUNCtion:SRL:MODE? (Option 100 only)
SENSe:FUNCtion:SRL:SCAN[:IMMediate] (Option 100 only)
SENSe[1|2]:STATe
SENSe[1|2]:STATe?
SENSe:SWEep:TRIGger:SOURce
SENSe:SWEep:TRIGger:SOURce?
SENSe:WINDow[:TYPE] (Option 100 only)
SENSe:WINDow[:TYPE]? (Option 100 only)

STATus:DEVice:CONDition?
STATus:DEVice:ENABle
STATus:DEVice:ENABle?
STATus:DEVice[:EVENt]?
STATus:DEVice:NTRansition
STATus:DEVice:NTRansition?
STATus:DEVice:PTRansition
STATus:DEVice:PTRansition?
STATus:OPERation:AVERaging:CONDition?
STATus:OPERation:AVERaging:ENABle
STATus:OPERation:AVERaging:ENABle?
STATus:OPERation:AVERaging[:EVENt]?
STATus:OPERation:AVERaging:NTRansition
STATus:OPERation:AVERaging:NTRansition?
STATus:OPERation:AVERaging:PTRansition
STATus:OPERation:AVERaging:PTRansition?
STATus:OPERation:MEASuring:CONDition?

```
STATus:OPERation:MEASuring:ENABle
STATus:OPERation:MEASuring:ENABle?
STATus:OPERation:MEASuring[:EVENt]?
STATus:OPERation:MEASuring:NTRansition
STATus:OPERation:MEASuring:NTRansition?
STATus:OPERation:MEASuring:PTRansition
STATus:OPERation:MEASuring:PTRansition?
STATus:QUEStionable:LIMit:CONDition?
STATus:QUEStionable:LIMit:ENABle
STATus:QUEStionable:LIMit:ENABle?
STATus:QUEStionable:LIMit[:EVENt]?
STATus:QUEStionable:LIMit:NTRansition
STATus:QUEStionable:LIMit:NTRansition?
STATus:QUEStionable:LIMit:PTRansition
STATus:QUEStionable:LIMit:PTRansition?

SYSTem:COMMunicate:GPIB:CONTroller[:STATe]
SYSTem:COMMunicate:GPIB:CONTroller[:STATe]?
SYSTem:COMMunicate:GPIB:ECHO
SYSTem:COMMunicate:GPIB:ECHO?
SYSTem:COMMunicate:GPIB:HCOPy:ADDRess
SYSTem:COMMunicate:GPIB:HCOPy:ADDRess?
SYSTem:COMMunicate:GPIB:MMEMory:ADDRess
SYSTem:COMMunicate:GPIB:MMEMory:ADDRess?
SYSTem:COMMunicate:GPIB:MMEMory:UNIT
SYSTem:COMMunicate:GPIB:MMEMory:UNIT?
SYSTem:COMMunicate:GPIB:MMEMory:VOLume
SYSTem:COMMunicate:GPIB:MMEMory:VOLume?
SYSTem:COMMunicate:SERial:TRANsmit:HANDshake
SYSTem:COMMunicate:SERial:TRANsmit:HANDshake?
SYSTem:COMMunicate:TTL:USER:FEED:KEY[:STATe]
SYSTem:COMMunicate:TTL:USER:FEED:KEY[:STATe]?
SYSTem:KEY:MASK?
SYSTem:KEY:QUEue:CLEar
SYSTem:KEY:QUEue:COUNt?
SYSTem:KEY:QUEue:MAXimum?
SYSTem:KEY:QUEue[:STATe]
SYSTem:KEY:QUEue[:STATe]?
SYSTem:KEY:TYPE?
SYSTem:KEY:USER
SYSTem:SET:LRNLong
```

```
TEST:RESult?
TEST:SELect
TEST:SELect?
TEST:STATe
TEST:STATe?
TEST:VALue
TEST:VALue?
```

14

SCPI Error Messages

SCPI Error Messages

This chapter contains the same error message information that can be found in the *SCPI 1994 Volume 2: Command Reference*. There are four sections in this chapter:

- Command Errors
- Execution Errors
- Device-Specific Errors
- Query Errors

NOTE

Your analyzer does not use all of the error messages listed in this chapter.

Command Errors

An error/event number in the range −199 to −100 indicates that an IEEE 488.2 syntax error has been detected by the instrument's parser. The occurrence of any error in this class shall cause the command error bit (bit 5) in the event status register (IEEE 488.2, section 11.5.1) to be set. One of the following events has occurred:

- An IEEE 488.2 syntax error has been detected by the parser. That is, a controller-to-device message was received which is in violation of the IEEE 488.2 standard. Possible violations include a data element which violates the device listening formats or whose type is unacceptable to the device.
- An unrecognized header was received. Unrecognized headers include incorrect device-specific headers and incorrect or unimplemented IEEE 488.2 common commands.
- A Group Execute Trigger (GET) was entered into the input buffer inside of an IEEE 488.2 program message.

Events that generate command errors shall not generate execution errors, device-specific errors, or query errors; see the other error definitions in this chapter.

Command Errors

Table 14-1. SCPI Command Errors

Error Number	Error Description
−100	Command error — This is the generic syntax error for devices that cannot detect more specific errors. This code indicates only that a Command Error has occurred.
−101	Invalid character — A syntactic element contains a character which is invalid for that type; for example, a header containing an ampersand, SETUP&. This error might be used in place of errors −114, −121, −141, and perhaps some others.
−102	Syntax error — An unrecognized command or data type was encountered; for example, a string was received when the device does not accept strings.
−103	Invalid separator — The parser was expecting a separator and encountered an illegal character; for example, the semicolon was omitted after a program message unit, `*EMC 1:CH1:VOLTS 5`.
−104	Data type error — The parser recognized a data element different than one allowed; for example, numeric or string data was expected but block data was encountered.
−105	GET not allowed — A Group Execute Trigger was received within a program message.
−108	Parameter not allowed — More parameters were received than expected for the header; for example, the `*EMC` common command only accepts one parameter, so receiving `*EMC 0,1` is not allowed.
−109	Missing parameter — Fewer parameters were received than required for the header; for example, the `*EMC` common command requires one parameter, so receiving `*EMC` is not allowed.
−110	Command header error — An error was detected in the header. This error message should be used when the device cannot detect the more specific errors described for errors −111 through −119.
−111	Header separator error — A character which is not a legal header separator was encountered while parsing the header; for example, no white space followed the header, thus `*GMC"MACRO"` is an error.
−112	Program mnemonic too long — The header contains more that twelve characters.
−113	Undefined header — The header is syntactically correct, but it is undefined for this specific device; for example, `*XYZ` is not defined for any device.
−114	Header suffix out of range — The value of a numeric suffix attached to a program mnemonic makes the header invalid.
−120	Numeric data error — This error, as well as errors −121 through −129, are generated when parsing a data element which appears to be numeric, including the nondecimal numeric types. This particular error message should be used if the device cannot detect a more specific error.
−121	Invalid character in number — An invalid character for the data type being parsed was encountered; for example, an alpha in a decimal numeric or a "9" in octal data.
−123	Exponent too large — The magnitude of the exponent was larger than 32000.

Table 14-1. SCPI Command Errors (continued)

Error Number	Error Description
−124	Too many digits — The mantissa of a decimal numeric data element contained more than 255 digits excluding leading zeros.
−128	Numeric data not allowed — A legal numeric data element was received, but the device does not accept one in this position for the header.
−130	Suffix error — This error, as well as errors −131 through −139, are generated when parsing a suffix. This particular error message should be used if the device cannot detect a more specific error.
−131	Invalid suffix — The suffix does not follow the correct syntax, or the suffix is inappropriate for this device.
−134	Suffix too long — The suffix contained more than 12 characters.
−138	Suffix not allowed — A suffix was encountered after a numeric element which does not allow suffixes.
−140	Character data error — This error, as well as errors −141 through −149, are generated when parsing a character data element. This particular error message should be used if the device cannot detect a more specific error.
−141	Invalid character data — Either the character data element contains an invalid character or the particular element received is not valid for the header.
−144	Character data too long — The character data element contains more than twelve characters.
−148	Character data not allowed — A legal character data element was encountered where prohibited by the device.
−150	String data error — This error, as well as errors −151 through −159, are generated when parsing a string data element. This particular error message should be used if the device cannot detect a more specific error.
−151	Invalid string data — A string data element was expected, but was invalid for some reason. For example, an END message was received before the terminal quote character.
−158	String data not allowed — A string data element was encountered but was not allowed by the device at this point in parsing.
−160	Block data error — This error, as well as errors −161 through −169, are generated when parsing a block data element. This particular error message should be used if the device cannot detect a more specific error.
−161	Invalid block data — A block data element was expected, but was invalid for some reason. For example, an END message was received before the length was satisfied.
−168	Block data not allowed — A legal block data element was encountered but was not allowed by the device at this point in parsing.
−170	Expression error — This error, as well as errors −171 through −179, are generated when parsing an expression data element. This particular error message should be used if the device cannot detect a more specific error.

Table 14-1. SCPI Command Errors (continued)

Error Number	Error Description
−171	Invalid expression — The expression data element was invalid (for example, unmatched parentheses or an illegal character).
−178	Expression data not allowed — A legal expression data was encountered but was not allowed by the device at this point in parsing.
−180	Macro error — This error, as well as errors −181 through −189, are generated when defining or executing a macro. This particular error message should be used if the device cannot detect a more specific error.
−181	Invalid outside macro definition — Indicates that a macro parameter placeholder ($<number) was encountered outside of a macro definition.
−183	Invalid inside macro definition — Indicates that the program message unit sequence, sent with a `*DDT` or `*DMC` command, is syntactically invalid.
−184	Macro parameter error — Indicates that a command inside the macro definition had the wrong number or type of parameters.

Execution Errors

An error/event number in the range −299 to −200 indicates that an error has been detected by the instrument's execution control block. The occurrence of any error in this class shall cause the execution error bit (bit 4) in the event status register to be set. One of the following events has occurred:

- A program data element following a header was evaluated by the device as outside of its legal input range or is otherwise inconsistent with the device's capabilities.
- A valid program message could not be properly executed due to some device condition.

Execution errors shall be reported by the device after rounding and expression evaluation operations have taken place. Rounding a numeric data element, for example, shall not be reported as an execution error. Events that generate execution errors shall not generate Command Errors, device-specific errors, or Query Errors; see the other error definitions in this section.

Table 14-2. SCPI Execution Errors

Error Number	Error Description
−200	Execution error — This is the generic syntax error for devices that cannot detect more specific errors. This code indicates only that an Execution Error has occurred.
−201	Invalid while in local — Indicates that a command is not executable while the device is in local due to a hard local control; for example, a device with a rotary switch receives a message which would change the switches state, but the device is in local so the message can not be executed.
−202	Settings lost due to rtl — Indicates that a setting associated with a hard local control was lost when the device changed to LOCS from REMS or to LWLS from RWLS.
−203	Command protected — Indicates that a legal password-protected program command or query could not be executed because the command was disabled.
−210	Trigger error
−211	Trigger ignored — Indicates that a GET, *TRG, or triggering signal was received and recognized by the device but was ignored because of device timing considerations; for example, the device was not ready to respond.[1]
−212	Arm ignored — Indicates that an arming signal was received and recognized by the device but was ignored.
−213	Init ignored — Indicates that a request for a measurement initiation was ignored as another measurement was already in progress.
−214	Trigger deadlock — Indicates that the trigger source for the initiation of a measurement is set to GET and subsequent measurement query is received. The measurement cannot be started until a GET is received, but the GET would cause an INTERRUPTED error.
−215s	Arm deadlock — Indicates that the arm source for the initiation of a measurement is set to GET and subsequent measurement query is received. The measurement cannot be started until a GET is received, but the GET would cause an INTERRUPTED error.
−220	Parameter error — Indicates that a program data element related error occurred. This error message should be used when the device cannot detect the more specific errors −221 through −229.
−221	Settings conflict — Indicates that a legal program data element was parsed but could not be executed due to the current device state.
−222	Data out of range — Indicates that a legal program data element was parsed but could not be executed because the interpreted value was outside the legal range as defined by the device.
−223	Too much data — Indicates that a legal program data element of block, expression, or string type was received that contained more data than the device could handle due to memory or related device-specific requirements.
−224	Illegal parameter value — Used where an exact value, from a list of possible values, was expected.

1 A DT0 device always ignores GET and treats *TRG as a Command Error.

Table 14-2. SCPI Execution Errors (continued)

Error Number	Error Description
−225	Out of memory — The device has insufficient memory to perform the requested operation.
−226	Lists not same length — Attempted to use LIST structure having individual LIST's of unequal lengths.
−230	Data corrupt or stale — Possibly invalid data; new reading started but not completed since last access.
−231	Data questionable — Indicates that measurement accuracy is suspect.
−232	Invalid format — Indicates that a legal program data element was parsed but could not be executed because the data format or structure is inappropriate, such as when loading memory tables or when sending a `SYSTem:SET` parameter from an unknown instrument.
−233	Invalid version — Indicates that a legal program data element was parsed but could not be executed because the version of the data is incorrect to the device. This particular error should be used when file or block data formats are recognized by the instrument but cannot be executed for reasons of version incompatibility. For example, an unsupported file version, or an unsupported instrument version.
−240	Hardware error — Indicates that a legal program command or query could not be executed because of a hardware problem in the device. Definition of what constitutes a hardware problem is completely device-specific. This error message should be used when the device cannot detect the more specific errors described for errors −241 through −249.
−241	Hardware missing — Indicates that a legal program command or query could not be executed because of missing device hardware; for example, an option was not installed. Definition of what constitutes missing hardware is completely device-specific.
−250	Mass storage error — Indicates that a mass storage error occurred. This error message should be used when the device cannot detect the more specific errors described for errors −251 through −259.
−251	Missing mass storage — Indicates that a legal program command or query could not be executed because of missing mass storage; for example, an option that was not installed. Definition of what constitutes missing mass storage is device-specific.
−252	Missing media — Indicates that a legal program command or query could not be executed because of a missing media; for example, no disk. The definition of what constitutes missing media is device-specific.
−253	Corrupt media — Indicates that a legal program command or query could not be executed because of corrupt media; for example, bad disk or wrong format. The definition of what constitutes corrupt media is device-specific.
−254	Media full — Indicates that a legal program command or query could not be executed because the media was full; for example, there is no room on the disk. The definition of what constitutes a full media is device-specific.
−255	Directory full — Indicates that a legal program command or query could not be executed because the media directory was full. The definition of what constitutes a full media directory is device-specific.

Table 14-2. SCPI Execution Errors (continued)

Error Number	Error Description
−256	File name not found — Indicates that a legal program command or query could not be executed because the file name on the device media was not found; for example, an attempt was made to read or copy a nonexistent file. The definition of what constitutes a file not being found is device-specific.
−257	File name error — Indicates that a legal program command or query could not be executed because the file name on the device media was in error; for example, an attempt was made to copy to a duplicate file name. The definition of what constitutes a file name error is device-specific.
−258	Media protected — Indicates that a legal program command or query could not be executed because the media was protected; for example, the write-protect tab on a disk was present. The definition of what constitutes protected media is device-specific.
−260	Expression error — Indicates that an expression program data element related error occurred. This error message should be used when the device cannot detect the more specific errors described for errors −261 through −269.
−261	Math error in expression — Indicates that a syntactically legal expression program data element could not be executed due to a math error; for example, a divide-by-zero was attempted. The definition of math error is device-specific.
−270	Macro error — Indicates that a macro-related execution error occurred. This error message should be used when the device cannot detect the more specific errors −271 through −279.
−271	Macro syntax error — Indicates that a syntactically legal macro program data sequence could not be executed due to a syntax error within the macro definition.
−272	Macro execution error — Indicates that a syntactically legal macro program data sequence could not be executed due to some error in the macro definition.
−273	Illegal macro label — Indicates that the macro label defined in the `*DMC` command was a legal string syntax, but could not be accepted by the device; for example, the label was too long, the same as a common command header, or contained invalid header syntax.
−274	Macro parameter error — Indicates that the macro definition improperly used a macro parameter placeholder.
−275	Macro definition too long — Indicates that a syntactically legal macro program data sequence could not be executed because the string or block contents were too long for the device to handle.
−276	Macro recursion error — Indicates that a syntactically legal macro program data sequence could not be executed because the device found it to be recursive.

Table 14-2. SCPI Execution Errors (continued)

Error Number	Error Description
−277	Macro redefinition not allowed — Indicates that a syntactically legal macro label in the *DMC command could not be executed because the macro label was already defined.
−278	Macro header not found — Indicates that a syntactically legal macro label in the *GMC? query could not be executed because the header was not previously defined.
−280	Program error — Indicates that a downloaded program-related execution error occurred. This error message should be used when the device cannot detect the more specific errors −281 through −289. A downloaded program is used to add algorithmic capability to a device. The syntax used in the program and the mechanism for downloading a program is device-specific.
−281	Cannot create program — Indicates that an attempt to create a program was unsuccessful. One reason for failure might include not enough memory.
−282	Illegal program name — The name used to reference a program was invalid; for example, redefining an existing program, deleting a nonexistent program, or in general, referencing a nonexistent program.
−283	Illegal variable name — An attempt was made to reference a nonexistent variable in a program.
−284	Program currently running — Certain operations dealing with programs may be illegal while the program is running; for example, deleting a running program might not be possible.
−285	Program syntax error — Indicates that a syntax error appears in a downloaded program. The syntax used when parsing the downloaded program is device-specific.
−286	Program runtime error
−290	Memory use error — Indicates that a user request has directly or indirectly caused an error related to memory or data_handles (this is not the same as "bad" memory).
−291	Out of memory
−292	Referenced name does not exist
−293	Referenced name already exists
−294	Incompatible type — Indicates that the type or structure of a memory item is inadequate.

Device-Specific Errors

An error/event number in the range −399 to −300 or 1 to 32767 indicates that the instrument has detected an error which is not a command error, a query error, or an execution error. It indicates that some device operations did not properly complete, possibly due to an abnormal hardware or firmware condition. These codes are also used for self-test response errors. The occurrence of any error in this class should cause the device-specific error bit (bit 3) in the event status register to be set.

The meaning of positive error codes is device-dependent and may be enumerated or bit mapped; the error message string for positive error codes is not defined by SCPI and available to the device designer. Note that the string is not optional; if the designer does not wish to implement a string for a particular error, the null string should be sent (for example, 42,""). The occurrence of any error in this class should cause the device-specific error bit (bit 3) in the event status register to be set. Events that generate device-specific errors shall not generate command errors, execution errors, or query errors; see the other error definitions in this section.

Table 14-3. SCPI Device-Specific Errors

Error Number	Error Description
−300	Device-specific error — This is the generic device-dependent error for devices that cannot detect more specific errors. This code indicates only that a Device-Dependent Error has occurred.
−310	System error — Indicates that some error, termed "system error" by the device, has occurred. This code is device-dependent.
−311	Memory error — Indicates that an error was detected in the device's memory. The scope of this error is device-dependent.
−312	PUD memory lost — Indicates that the protected user data saved by the `*PUD` command has been lost.
−313	Calibration memory lost — Indicates that nonvolatile calibration data used by the `*CAL?` command has been lost.
−314	Save/recall memory lost — Indicates that the nonvolatile data saved by the `*SAV?` command has been lost.
−315	Configuration memory lost — Indicates that nonvolatile configuration data saved by the device has been lost. The meaning of this error is device-specific.
−330	Self-test failed
−350	Queue overflow — A specific code entered into the queue in lieu of the code that caused the error. This code indicates that there is no room in the queue and an error occurred but was not recorded.
−360	Communication error — This is the generic communication error for devices that cannot detect the more specific errors −361 through −363.
−361	Parity error in program message — Parity bit not correct when data received, for example, on a serial port.
−362	Framing error in program message — A stop bit was not detected when data was received, for example, on a serial port (for example, a baud rate mismatch).
−363	Input buffer overrun — Software or hardware input buffer on serial port overflows with data caused by improper or nonexistent pacing.

Query Errors

An error/event number in the range −499 to −400 indicates that the output queue control of the instrument has detected a problem with the message exchange protocol. The occurrence of any error in this class shall cause the query error bit (bit 2) in the event status register to be set. These errors correspond to message exchange protocol errors. One of the following is true:

- An attempt is being made to read data from the output queue when no output is either present or pending;
- Data in the output queue has been lost.

Events that generate query errors shall not generate command errors, execution errors, or device-specific errors; see the other error definitions in this section.

Table 14-4. SCPI Query Errors

Error Number	Error Description
−400	Query error — This is the generic query error for devices that cannot detect more specific errors. This code indicates only that a Query Error has occurred.
−410	Query **INTERRUPTED** — Indicates that a condition causing an **INTERRUPTED** Query error occurred; for example, a query followed by **DAB** or **GET** before a response was completely sent.
−420	Query **UNTERMINATED** — Indicates that a condition causing an **UNTERMINATED** Query error occurred; for example, the device was addressed to talk and an incomplete program message was received.
−430	Query **DEADLOCKED** — Indicates that a condition causing a **DEADLOCKED** Query error occurred; for example, both input buffer and output buffer are full and the device cannot continue.
−440	Query **UNTERMINATED** after indefinite response — Indicates that a query was received in the same program message after an query requesting an indefinite response was executed.

Index

Index

3 **3.5 mm**, 11-25

7 **75-ohm Formats**, 11-21

A **A**, 11-6
abbreviation
  of commands, 10-8
Abort
  copy, 11-31
ABORt, 10-4, 12-2
active controller
  defined, 1-2
Active Marker Off, 11-15
Add Max Line, 11-18
Add Min Line, 11-18
Add Min Point, 11-18
address
  HP-IB, 1-2
address capability, 1-7
AH1, 1-8
allocate memory, 12-20
All Off, 11-16
Alt Sweep on OFF, 11-11
AM Delay, 11-6
  cal, 11-24
Annotation, 11-33
ANNotation, 12-12, 12-13
Aperture, 11-26
A/R, 11-6
arrays
  correction, example program to up- and down-load, 8-40
  data, corrected, 6-25
  formatted, 6-27
  measurement, 6-21
  memory, corrected, 6-25
  raw data, 6-22
ASCDATA

example program, 8-24
ASCii, 4-7
ASCII encoding, 4-8
ATN, 1-4, 1-10
attention
control line, 1-4
Auto Feed
plotter, 11-32
printer, 11-31
Autoscale, 11-14
Auto Z, 11-24
Autozero, 11-25
AUX Input, 11-7
AVERage, 12-21
Average Factor, 11-26
Average on OFF, 11-26
averaging, 6-25
averaging status register set, 5-19
AVG, 11-26

B
B, 11-6
B*, 11-7
Band Pass, 11-8
Band Pass Max Span, 11-8
Bandwidth, 11-17
Baud Rate, 11-31
beep, 12-29
Beeper Volume, 11-35
BEGIN, 11-3, 12-8
Begin Frequency, 11-19
Begin Limit, 11-19
binary encoding, 1-4, 4-8
<block>, 10-14
block data, 4-5
block length
definite, 4-5
indefinite, 4-6
block parameters, 10-14
blocks
definite and indefinite length, 4-5, 4-6, 10-14
boolean parameters, 10-12
B/R, 11-6

B*/R*, 11-7
brackets
use of in this manual, 1-3, 10-15
branching, 10-5
Broadband External, 11-7
Broadband Internal, 11-7
buffer
graphics, 7-5
buffering user graphics, 12-14
bus
data, 1-4
bus management commands, 1-6
byte order, 12-15
bytes per point, during data transfer, 6-9
byte swapping, 4-9

C

C1, 1-8
C10, 1-8
C12, 1-8
C2, 1-8
C3, 1-8
C4, 1-8
C6, 1-8
Cable Loss, 11-23
cables, 1-2
CAL, 11-22
CALCulate, 10-4, 12-3, 12-4, 12-5, 12-6, 12-7
Calibrate Cable, 11-23
calibration
full band, 12-21
reflection, example program, 8-36
transmission, example program, 8-34
CALibration, 10-4, 12-7
Cal Kit, 11-24
Cal on OFF, 11-27
case-sensitivity, 10-8
CATalog, 12-20
Center, 11-8
CH1AFWD, 6-22
CH1BFWD, 6-22
CH1FDATA, 6-22
CH1RFWD, 6-22
CH1SCORR1, 6-24
CH1SCORR2, 6-24
CH1SCORR3, 6-24
CH1SDATA, 6-22

CH1SMEM, 6-22
CH2AFWD, 6-22
CH2BFWD, 6-22
CH2FDATA, 6-22
CH2RFWD, 6-22
CH2SCORR1, 6-24
CH2SCORR2, 6-24
CH2SCORR3, 6-24
CH2SDATA, 6-22
CH2SMEM, 6-22
(CHAN 1), 11-6
(CHAN 2), 11-6
change directory, 12-18, 12-19
Chan OFF, 11-7
<char>, 10-11
character data, 4-4
character parameters, 10-11
circle
  to draw, 12-14
clear graphics, 12-14
clearing registers, 5-4
Clear Program, 11-34
clock, 11-35
Clock Format, 11-35
Clock Off, 11-19
*CLS, 2-5, 5-4, 10-17
colons
  use of, 10-5, 10-16
Color, 11-31, 11-32
color of pen, 12-14
command abbreviation, 10-8
command errors, 14-3
command parser, 1-14
commands
  bus management, 1-6
  device, 1-6
  IEEE 488.2, 10-17
  overlapped, 2-3
  SCPI standard, 13-3
  sequential, 2-2
command sending, 1-6
command tree, 10-3
commas
  use of, 10-10, 10-16
condition register, 5-4
CONFigure, 12-8
Configure Ext Disk, 11-28

Configure VOL_RAM, 11-28
configuring measurements, 8-5
Connector C, 11-24
Connector Length, 11-24
Connector Model, 11-24
Continue, 11-34
Continuous, 11-12
control
  passing, 3-2
controller
  active, defined, 1-2
  defined, 1-2
  multiple, 1-7
  system, defined, 1-2
controller capabilities, 1-9
control lines, 1-4
Conversion Loss, 11-6
Copy All Files, 11-29
copy file, 12-19
Copy File, 11-29
corrected data arrays, 6-25
corrected memory arrays, 6-25
CORRection, 12-21
correction arrays
  up- and down-loading, example program, 8-40
COUPle, 11-11
coupling, 11-11
Current Size, 11-28
customized procedure
  example program, 8-75
CW, 11-8

D

data
  block, 4-5
  character, 4-4
  expression, 4-4
  numeric, 4-3
  raw, 6-22
  string, 4-4
Data, 11-18
DATA, 12-31
Data and Memory, 11-18
data arrays
  corrected, 6-25

mnemonics, 6-22
data bus, 1-4
data encoding, 4-7
data format, 12-15
Data/Mem, 11-18
Data -> Mem, 11-18
Data on OFF, 11-27
data rate, 1-7
data trace, 12-3
data transfer, 4-2
in ASCII, example program, 8-24
in REAL format, example program, 8-27
using INTEGER format, example program, 8-30
data transfer size, 6-9
data types, 4-2, 4-3
Day
set clock, 11-35
DC1, 1-8
DCL, 1-10, 1-12
Default Cal, 11-22, 11-23
Default Type-N(f), 11-24
Define Graph, 11-33
Define Hardcopy, 11-33
Define Plotter, 11-32
Define Printer, 11-31
Define Save, 11-27
definite length blocks, 4-5, 10-14
delay
electrical, 6-26
Delay
format, 11-21
Delay Aperture, 11-26
Delete All Files, 11-29
Delete All Limits, 11-18
Delete File, 11-29
Delete Limit, 11-18
delete program, 12-20
delimiters, 10-13
Delta Mkr, 11-16
Detection Options, 11-6
Detector Zero, 11-25
device clear, 1-10, 1-12

device commands, 1-6
device-specific errors, 14-12
device status register set, 5-15
DIAGnostic, 12-9
Directory Utilities, 11-30
direct-read method of accessing registers, 5-6
discrete parameters, 10-11
Disp Freq Resolution, 11-9
DISPlay, 10-4, 12-12, 12-13, 12-14, 12-15
(DISPLAY), 11-18
display window
  pixel coordinates, 12-13
  width and height, 12-14
Distance, 11-12
Dither, 11-13
double quotes
  use of, 10-13
download an IBASIC program, 12-20
draw
  circle, 12-14
  line, 12-14
  rectangle, 12-14
DTR/DSR, 11-31

E

E2, 1-8
Edit Limit, 11-19
electrical delay, 6-26
Electrical Delay, 11-14
enable register, 5-5
encoding data, 4-2, 4-7
End Frequency, 11-19
End Limit, 11-19
end or identify
  control line, 1-5
EOI, 1-5
error coefficient arrays, 6-23
error correction, 6-23
error messages, 12-30, 12-31, 14-2
error queue, 1-14
  to query, 12-30, 12-31
errors
  command, 14-3
  device-specific, 14-12
  execution, 14-7
  query, 14-14
*ESE, 10-17

*ESE?, 5-18, 10-17
*ESR?, 5-18, 10-17
event register, 5-4
example program
  ASCDATA, 8-24
  FAST_PRT, 8-62
  GRAPHICS, 8-75
  INTDATA, 8-30
  LEARNSTR, 8-49
  LIMITEST, 8-9
  LOADCALS, 8-40
  MARKERS, 8-14
  PASSCTRL, 8-59
  PRINTPLT, 8-56
  REFLCAL, 8-36
  SAVERCL, 8-52
  SETUP, 8-6
  SRQ, 8-65
  TRANCAL, 8-34
example programs, 8-2–81
execute an IBASIC command, 12-20
execution errors, 14-7
expression data, 4-4
Ext Disk, 11-28
External CRT Adjust, 11-36
External Disk, 11-28
External Point
  trigger, 11-12
External Sweep
  trigger, 11-12
Ext Ref, 11-12

F FAST_PRT example program, 8-62
FastRecall on OFF, 11-30
Fault Location, 11-6, 11-22
Fault Location, calibration, 11-23
Fault Loc Frequency, 11-8
Fault Window, 11-26
Feet, 11-12
file copy, 12-19
File Type bin ASCII, 11-28
File Utilities, 11-29
Fine, 11-26

Flatness , 11-16
font
  label, 12-14
FORMat, 10-4, 12-15
(FORMAT), 11-21
Format Disk , 11-29
format of numerics, 10-11
formatted arrays, 6-27
formatting, 6-27
(FREQ), 11-8
FREQuency, 12-23
frequency, stop
  how to set, 10-10
Frequency Sweep , 11-11
front panel keycodes, 9-2
Full
  IBASIC display, 11-34
Full Band Cal , 11-22, 11-23

G general status register model, 5-3
go to local, 1-10
Graph , 11-33
graphics
  buffering, 12-14
  to clear, 12-14
  user, 7-2
GRAPhics, 12-13, 12-14
GRAPHICS
  example program, 8-75
graphics buffer, 7-5
Graticule ON off , 11-20, 11-33
GTL, 1-10

H handshake lines, 1-4
(HARD COPY), 11-31
Hardcopy Address , 11-31
Hardcopy All , 11-35
hardcopy output
  example program, 8-56
Hardcopy Screen , 11-35
HCOPy, 10-4, 12-16
Hold , 11-12
Horizontal Back Porch , 11-36
Horizontal Frnt Porch , 11-36

Hour
set clock, 11-35
HP 871_B Address, 11-34
HP-IB addresses, 1-2
HP-IB cables, 1-2
HP-IB Echo, 11-34
HP-IB queues, 1-13
HP-IB requirements, 1-7

I IBASIC, 8-2
IBASIC, 11-34
IBASIC Display, 11-34
IBASIC program
to delete, 12-20
to download, 12-20
to load a value, 12-20
to select, 12-20
*IDN?, 10-17
IEEE 488.2 common commands, 10-17
IFC, 1-4, 1-10
Imaginary
format, 11-21
Impedance Magnitude
format, 11-21
implied mnemonics, 10-9
how identified in this manual, 1-3
implied variable, 10-9
implied variables
how identified in this manual, 1-3
indefinite block length, 4-6
indefinite length blocks, 10-14
INITiate, 10-4, 12-17
INITIATE, 10-9
input queue, 1-13
Instrument BASIC, 8-2
Inst State ON off, 11-27
INTDATA
example program, 8-30
INTeger, 4-7
interface capabilities, 1-8
interface clear, 1-4, 1-10
Internal
trigger, 11-12
Internal 3.5" Disk, 11-28
internal measurement arrays, 6-21

L L4, 1-8
label
  to draw, 12-14
label font, 12-14
Landscape, 11-31
language, 12-16
LE0, 1-8
LEARNSTR
  example program, 8-49
learn string, 12-30, 12-31
  example program, 8-49
Level, 11-10
LIMit, 12-3, 12-4
limitations
  length of HP-IB cables, 1-7
  number of devices, 1-7
LIMITEST
  example program, 8-9
limit fail register set, 5-15
Limit Line, 11-18
Limit Line ON off, 11-19
limit lines
  example program, 8-9
Limit Test on OFF, 11-19
line
  to draw, 12-14
Lin Mag, 11-21
listener
  defined, 1-2
List Trace Values, 11-33
LLO, 1-11
load a value in an IBASIC program, 12-20
LOADCALS
  example program, 8-40
local lockout, 1-11
Log Mag, 11-21
Lower
  IBASIC display, 11-34
lower-case
  use of, 10-15
lower-case lettering, 10-8
Low Pass, 11-8
*LRN?, 10-17
LRN?, 12-30, 12-31
LRNLong?, 12-30, 12-31

M Mag dBmV, 11-21
Mag dBuV, 11-21
Mag dBV, 11-21
Manual Z, 11-24
Manual Zero, 11-25
Margin
  hardcopy, 11-32
MARKer, 12-4, 12-6
MARKER, 11-15
Marker Functions, 11-16
Marker Math, 11-16
Marker -> Center, 11-16
Marker -> Elec Delay, 11-16
Marker -> Reference, 11-16
MARKERS
  example program, 8-14
Marker Search, 11-16
math
  trace, 6-26
MATH, 12-7
Math Off, 11-16
Maximum, 11-26
MAXimum, 10-10
Max Limit, 11-19
Max Search, 11-16
Measure Cable, 11-23
Measure Connector, 11-24
measurement
  basic setup example program, 8-6
measurement arrays
  internal, 6-21
measurements
  to configure, 8-5
measurement setup
  example program, 8-6
Measure Standard, 11-22, 11-24
measuring status register set, 5-19
Medium, 11-26
Memory, 11-18
memory allocation, 12-20
memory arrays

corrected, 6-25
MENU, 11-12
message exchange, 1-13
messages
error, 12-30, 12-31, 14-2
message transfer scheme, 1-7
message window
clear current, 12-13
enable/disable, 12-13
off, 12-13
remove user-defined, 12-13
user-defined, 12-13
Meters, 11-12
Minimum, 11-26
MINimum, 10-10
Min Limit, 11-19
Min Search, 11-16
Minute
set clock, 11-35
Mkr Limit ON off, 11-19
Mkr Limits, 11-19
Mkr -> Max, 11-16
Mkr -> Min, 11-16
Mkr Symbol, 11-33
Mkr Table, 11-33
MMEMory, 10-4, 12-18, 12-19
mnemonics
implied, 10-9
implied, how identified in this manual, 1-3
Modify Size, 11-28
Monochrome, 11-31, 11-32
Month
set clock, 11-35
MultiNotch, 11-17
Multi Peak, 11-17
Multi Peak Corr, 11-23
Multi Peak Threshold, 11-23
multiple commands, 10-7
multiple controller capability, 1-7

N

Narrow, 11-26
Narrowband Internal, 11-6
Next Min, 11-16
Next Peak, 11-16
None
  IBASIC display, 11-34
Non-Vol RAM Disk, 11-28
no pending operation, 2-4
Normalize, 11-18, 11-22
Notch, 11-17
NPO, 2-4
NR1, 10-11
NR2, 10-11
NR3, 10-11
<num>, 10-10
Number of Points, 11-12
numeric data, 4-3
numeric formats, 10-11
numeric parameters, 10-10

O

offset and scale, 6-27
One Port
  cal, 11-22
&<ON|OFF>, 10-12
*OPC, 2-4, 10-17
*OPC?, 2-4, 10-18
Operating Parameters, 11-35
operational status register set, 5-20
*OPT?, 10-18
options, 10-18
OUTPut, 10-4, 12-19
output queue, 1-14
overlapped commands, 2-3

P

parallel poll, 1-11
parameters
  block, 10-14
  boolean, 10-12
  character, 10-11
  discrete, 10-11
  numeric, 10-10
  string, 10-13
parameter types, 10-10
parser

command, 1-14
pass control, 3-2
PASSCTRL
example program, 8-59
passing control, 3-2
example program, 8-59
*PCB, 1-12, 10-18
pen, 11-32
move, 12-14
Phase
format, 11-21
Phase Offset, 11-14
pixel coordinates
display window, 12-13
plotting and printing
example program, 8-56
Polar
format, 11-21
Port Ext's on OFF, 11-25
Portrait, 11-31
Power, 11-6
POWer, 12-19, 12-25
(POWER), 11-10
Power Sweep, 11-11
PPC, 1-11
PPD, 1-11
PPE, 1-11
PPU, 1-11
(PRESET), 11-2, 12-30, 12-31
Printer Resolution, 11-32
printing and plotting
example program, 8-56
PRINTPLT
example program, 8-56
Print Width, 11-32
program
how to download, 12-20
PROGram, 10-4, 12-20
programming fundamentals, 1-9
programs
examples, 8-2–81
*PSC, 10-18
Pwr Sweep Range, 11-10

Q
query errors, 14-14
query response generation, 1-15
query the error queue, 12-30, 12-31
questionable status register set, 5-16
queue
  error, 1-14
  input, 1-13
  output, 1-14
queues, 1-13
quotes
  use of, 10-13

R
R, 11-6
R*, 11-7
ratio calculations, 6-23
raw data arrays, 6-22
readable ports, 12-11
Real
  format, 11-21
REAL, 4-7
recalling and saving
  example program, 8-52
Recall Program, 11-28
Recall State, 11-27
rectangle
  to draw, 12-14
Reference Level, 11-14
Reference Position, 11-14
REFLCAL
  example program, 8-36
Reflection, 11-6
  cal, 11-22
reflection calibration
  example program, 8-36
Refl Port Extension, 11-25
register model
  status, 5-3
registers, 5-2
  how to use, 5-6
register sets, 5-10
remote enable, 1-11
  control line, 1-5
REN, 1-5, 1-11
Rename File, 11-29
Re-Save Program, 11-28

Re-Save State, 11-27
Response
  cal, 11-22
Response & Isolation
  cal, 11-22
Restart Average, 11-26
Restore Defaults
  cal, 11-22
RF Filter Stats, 11-16
RF ON off, 11-10
RL1, 1-8
RQS, 1-12
*RST, 2-5, 10-18
Run, 11-34

S Save ASCII, 11-27
Save AUTOST, 11-28
Save Chan, 11-27
Save Program, 11-28
SAVERCL
  example program, 8-52
SAVE RECALL, 11-27
Save State, 11-27
saving and recalling
  example program, 8-52
SCALE, 11-14
scale and offset, 6-27
Scale/Div, 11-14
SCPI
  defined, 10-2
SCPI conformance, 13-2
SCPI errors, 14-2
SCPI standard commands, 13-3
SDC, 1-12
Search left, 11-17
Search Off, 11-17
Search right, 11-17
select a program, 12-20
Select Copy Port, 11-31
Select Disk, 11-28
selected device clear, 1-12
semicolons

use of, 10-7, 10-16
sending commands, 1-6
SENSe, 10-4, 12-21, 12-22, 12-23
sequential commands, 2-2
serial poll, 1-12
service request
control line, 1-5
service request method accessing registers, 5-7
Set Clock , 11-35
Set Pen Numbers , 11-32
setting the stop frequency, 10-10
SETUP
example program, 8-6
SH1, 1-8
Show Clock, 11-19
Single, 11-12
single quotes
use of, 10-13
size
disk, 11-28
trace data transfer, 6-9
Smith chart , 11-21
Smith Chart ZO, 11-25
softkey labels
user-defined, 12-13
SOURce, 10-4, 12-25
source menu, 11-12
spaces
use of, 10-10, 10-16
Span , 11-8
SPD, 1-12
SPE, 1-12
Specify Length, 11-23
Split Display , 11-19
Spur Avoid, 11-13
Spur Avoid Options , 11-13
SR1, 1-8
*SRE, 10-18
*SRE?, 10-18
SRL , 11-6, 11-22
SRL Cable Scan, 11-12
SRQ, 1-5, 5-7
example program, 8-65
standard event status register, 5-3
standard event status register set, 5-17

Start, 11-8
copy, 11-31
Start Distance, 11-12
Start Power, 11-10
Statistics, 11-16
STATus, 10-5, 12-25, 12-26, 12-27, 12-28
PRESet Settings, 5-21
status byte register, 5-3, 5-12
status register model, 5-3
status registers, 5-2
*STB?, 10-19
Step, 11-34
Stop, 11-8
Stop Distance, 11-12
stop frequency
how to set, 10-10
Stop Power, 11-10
<string>, 10-13
string data, 4-4
string parameters, 10-13
subsystems, 10-3
SWAPped, 12-15
SWEep, 12-23
(SWEEP), 11-11
Sweep Time, 11-11
Sweep Time AUTO Man, 11-11
SWR, 11-21
synchronizing, 2-2
syntax summary and conventions, 10-15
SYSTem, 10-5, 12-29, 12-30, 12-31
System Bandwidth, 11-26
System Config, 11-35
system controller
defined, 1-2
System Controller, 11-34
(SYSTEM OPTIONS), 11-34

T T6, 1-8
take control talker, 1-12
talker
defined, 1-2
Talker Listener, 11-34
Target Search, 11-17
Target Value, 11-17
TCT, 1-12
TE0, 1-8
TEST, 12-31
Title and Clock, 11-19
Title + Clk ON off, 11-20, 11-33
TRACe, 10-5, 12-31
Trace Data, 11-33
trace data transfer size, 6-9
trace math, 12-7
trace math operation, 6-26
Tracking on OFF, 11-17
TRANCAL
example program, 8-34
transferring data, 4-2
using INTEGER format, example program, 8-30
transferring data in ASCII
example program, 8-24
transferring data in REAL format
example program, 8-27
transform, 6-26
transition registers, 5-4
transmission calibration
example program, 8-34
Transmissn, 11-6
cal, 11-22
Trans Port Extension, 11-25
*TRG, 10-19
trigger, 12-17
Trigger, 11-12
TRIGger, 10-5, 12-32
Trigger Source, 11-12
*TST?, 10-19
Type-F, 11-25
Type-N(m), 11-24

U
Upper
  IBASIC display, 11-34
upper-case
  use of, 10-15
upper-case lettering, 10-8
User Defined, 11-25
user-defined message, 12-13
user graphics
  example program, 8-75
using graphics, 7-2

V
variable
  implied, 10-9
variables
  implied, how identified in this manual, 1-3
Velocity Factor, 11-23, 11-25
Vertical Back Porch, 11-36
Vertical Frnt Porch, 11-36
Volatile RAM Disk, 11-28

W
*WAI, 2-4, 8-6, 10-19
Wide, 11-26
WINDow, 12-13, 12-14, 12-15
WINDow1, 7-2
WINDow10, 7-2
WINDow2, 7-2
window geometry, 7-4
window queries, 7-4

X
X, 11-7
Xon/Xoff, 11-31
X/Y, 11-7

Y
Y, 11-7
Y-Axis Lbl ON off, 11-20
Y-Axis Lbl rel ABS, 11-20
Year
  set clock, 11-35
Y/X, 11-7

Z Z cutoff Frequency, 11-24
Zeroing
auto, 11-25, 12-7
manual, 11-25