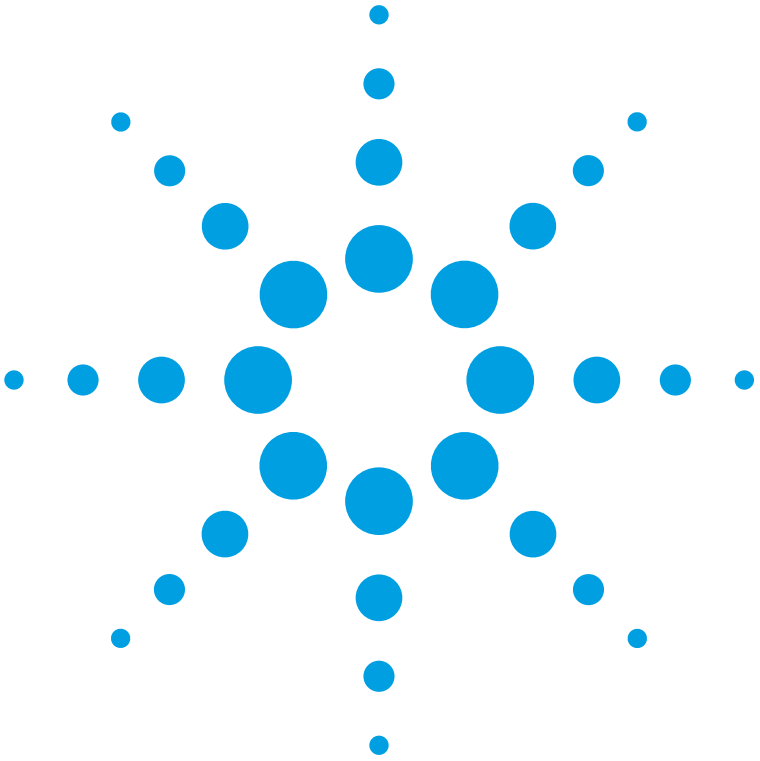
Agilent 8703B
Lightwave Component Analyzer
Programmer's Guide



Agilent Technologies

Notices

Manual Part Number

08703-90058

Edition

July 2004
Printed in Malaysia

Agilent Technologies, Inc.
Digital Signal Analysis
1400 Fountaingrove Parkway
Santa Rosa, CA 95403, USA

Warranty

The material contained in this document is provided “as is,” and is subject to being changed, without notice, in future editions. Further, to the maximum extent permitted by applicable law, Agilent disclaims all warranties, either express or implied, with regard to this manual and any information contained herein, including but not limited to the implied warranties of merchantability and fitness for a particular purpose. Agilent shall not be liable for errors or for incidental or consequential damages in connection with the furnishing, use, or performance of this document or of any information contained herein. Should Agilent and the user have a separate written agreement with warranty terms covering the material in this document that conflict with these terms, the warranty terms in the separate agreement shall control.

Safety Notices

CAUTION

Caution denotes a hazard. It calls attention to a procedure which, if not correctly performed or adhered to, could result in damage to or destruction of the product. Do not proceed beyond a caution sign until the indicated conditions are fully understood and met.

WARNING

Warning denotes a hazard. It calls attention to a procedure which, if not correctly performed or adhered to, could result in injury or loss of life. Do not proceed beyond a warning sign until the indicated conditions are fully understood and met.

Restricted Rights Legend.
Use, duplication, or disclosure by the U.S. Government is subject to restrictions as set forth in subparagraph (c) (1) (ii) of the Rights in Technical Data and Computer Software clause at DFARS 252.227-7013 for DOD agencies, and subparagraphs (c) (1) and (c) (2) of the Commercial Computer Software Restricted Rights clause at FAR 52.227-19 for other agencies.

Contents

1

Introduction to Instrument Control

Introduction to Instrument Control

In this chapter, you can find an introduction to the remote operation of your analyzer using an external controller. You should be familiar with the operation of the analyzer before attempting to remotely control the analyzer over the General Purpose Interface Bus (GPIB). Refer to the user's guide for operating information. For information on the instrument's preset state and memory allocation, refer to the *8703B Lightwave Component Analyzer Reference* manual.

This manual is not intended to teach programming or to discuss GPIB theory except at an introductory level. Programming examples that demonstrate the remote operation of the analyzer are documented in Chapter 8, "Programming Examples" and are also provided on the CD-ROM that was shipped with this manual. All example programs are provided in BASIC, and most are also provided in Visual C++ and Visual BASIC for use with the VXI*plug&play* driver.

Instrument Control using the VXI*plug&play* Driver

VXI*plug&play* is a term indicating conformance to a set of system-level standards produced by the VXI*plug&play* Systems Alliance. The charter of the alliance was “to improve the effectiveness of VXI-based solutions by increasing ease-of-use and improving the interoperability of multi-vendor VXI systems.”

Installing the VXI*plug&play* driver on your computer will allow you to control the analyzer via common programming environments without having to learn the instrument-specific mnemonics.

Requirements

The VXI*plug&play* driver for your analyzer is designed for a PC operating Windows 95 or Windows NT version 3.51 or higher. The driver requires a virtual instrument software architecture (VISA)-compatible GPIB interface, and the VISA I/O Library version 1.1 or higher. The driver is compatible with the following programming environments:

- Microsoft Visual Basic, version 4.0 or higher
- Microsoft Visual C++, version 4.0 or higher
- Borland C++, version 4.5 or higher
- Agilent VEE, version 3.2 or higher
- National Instruments LabWindows/CVI, version 4.0.1 or higher
- National Instruments LabVIEW, version 4.0.1 or higher

Installing the VXI*plug&play* Driver

NOTE This procedure assumes that you have installed a VISA-compatible GPIB interface and the VISA I/O library, version 1.1 or higher. It also assumes that you have installed—and are familiar with—one of the programming environments listed above.

1. The install program for the VXI*plug&play* driver for your analyzer is located in the root directory of the CD-ROM that accompanied this manual. The file is titled “875x.exe”

   a. If you need to order a new CD-ROM, contact Agilent Technologies and order part number 08703-10202.

   b. You can also download the file from the Web. Go to http://www.tm.agilent.com and follow the “Software and Driver” and “Instrument Driver” links.

2. Run “875x.exe” to install the VXI*plug&play* driver on your computer. The default directory that is used by the install-shield is vxipnp\winxx\875x, where winxx designates the operating system in use by your computer, such as winnt, win95, etc.

3. If you have difficulty installing the VXI*plug&play* driver, contact Agilent Technologies by calling the nearest sales or service office.

System Setup

1. Use an GPIB interconnect cable (such as 10833A/B/C/D) to connect the analyzer to the GPIB interface card on your computer.
2. Switch on the computer.
3. Switch on the analyzer.

   a. To verify the analyzer's address, press:

   Local, SET ADDRESSES, ADDRESS: 8703

   The analyzer has only one GPIB interface, though it occupies two addresses: one for the instrument and one for the display. The display address is equal to the instrument address with the least-significant bit incremented. The display address is automatically set each time the instrument address is set.

   The default analyzer addresses are:

   — 16 for the instrument

   — 17 for the display

CAUTION Other devices connected to the bus cannot occupy the same address as the analyzer or the display.

   The analyzer should now be displaying the instrument's address in the upper right section of the display. If the address is not 16, return the address to its default setting (16) by pressing:

   16, x1, Preset

   b. Set the system control mode to either “pass-control” or “talker/listener” mode. These are the only control modes in which the analyzer will accept commands over GPIB. To set the system-control mode, press:

   Local, TALKER/LISTENER

   or

   Local, USE PASS CONTROL

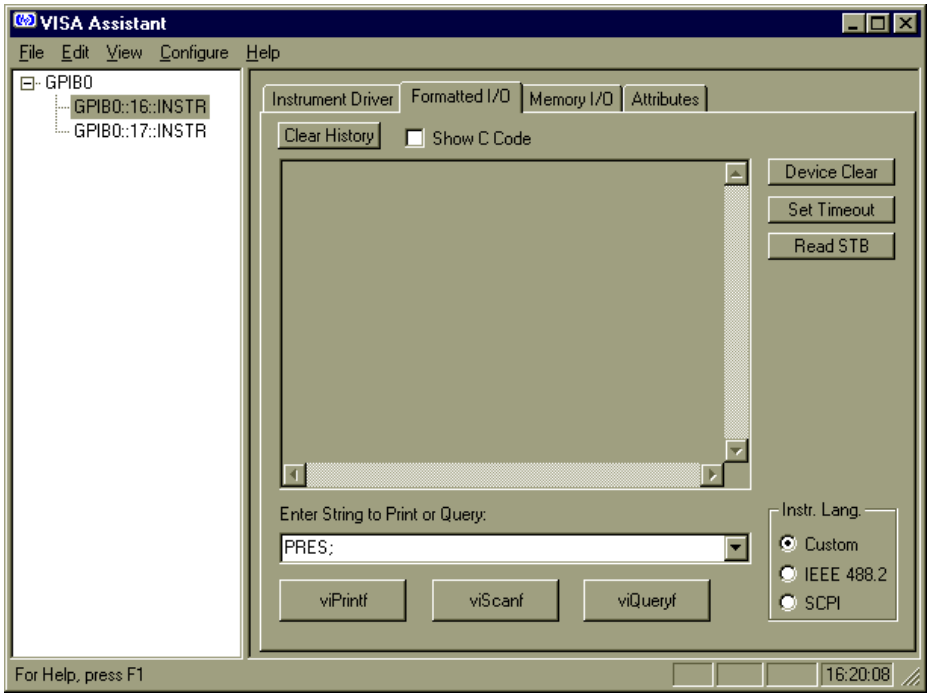
Verifying the Bus Connection

Check the interface bus connection and operation by following the appropriate procedure (for the type of interface card you are using) below.

Interface Bus Verification Procedure (GPIB Interface Card)

1. Check the bus connection by running the VISA Assistant in the I/O Libraries. The VISA Assistant will automatically report what it finds on the bus. Notice that the VISA Assistant is reporting instruments at addresses 16 and 17. As mentioned earlier, these addresses designate the instrument and its display, respectively.

Figure 1-1. VISA Assistant Window



2. To further verify GPIB operation, send a preset command to the analyzer by doing the following in the VISA Assistant window:

   a. Single-click on “GPIB0::16::INSTR” to highlight it.

   b. Make sure that the “Formatted I/O” tab is selected.

   c. Enter `PRES;` in the text box.

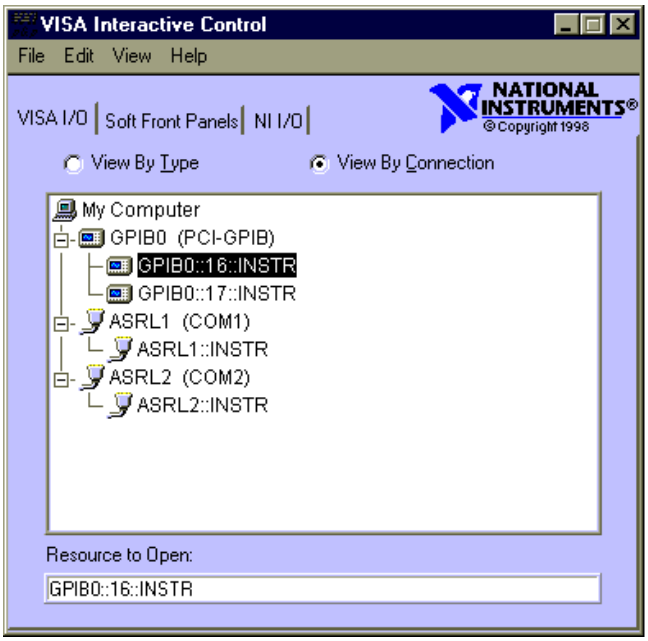
   d. Click on “viPrintf.”

   e. This command should preset the analyzer. If an instrument preset does not occur, there is a problem. Check all GPIB address settings and physical connections. Most GPIB problems are caused by an incorrect address or faulty/loose GPIB cables.

Interface Bus Verification Procedure (National Instruments Card)

1. Check the bus connection by running Win32 VISA Interactive Control. When this program is run, it automatically reports what it finds on the bus. Notice that the program is reporting instruments at addresses 16 and 17. As mentioned earlier, these addresses designate the instrument and its display, respectively.

Figure 1-2. Win32 VISA Interactive Control Window: Bus Report



2. To further verify GPIB operation, double click on “GPIB0::16::INSTR” and then perform the following steps.

   a. Make sure that the “Basic I/O” tab is selected.

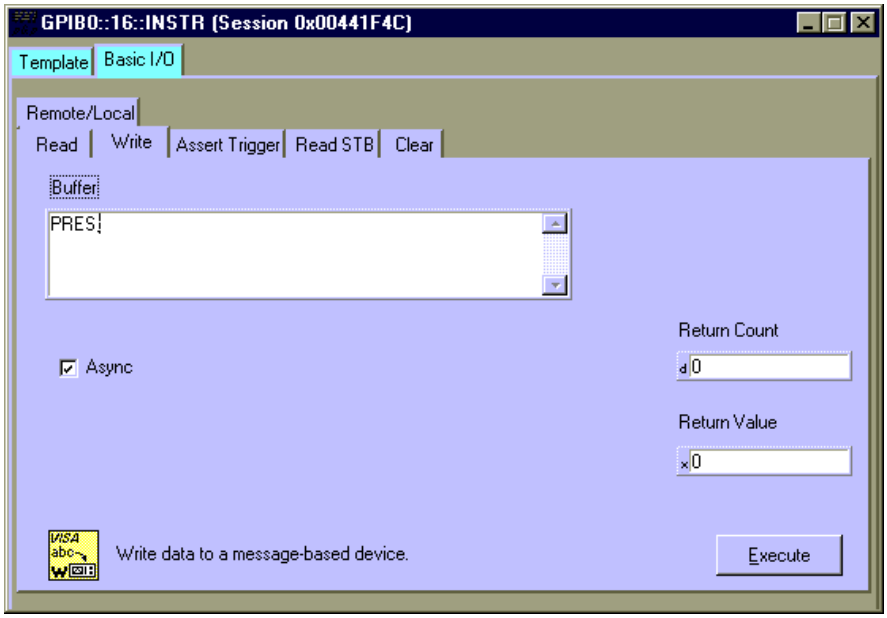
   b. Click on the “Write” tab.

   c. Enter `PRES;` in the “Buffer” text box.

   d. Click on “Execute.”

   e. This command should preset the analyzer. If an instrument preset does not occur, there is a problem. Check all GPIB address settings and physical connections. Most GPIB problems are caused by an incorrect address or faulty/loose GPIB cables.

Figure 1-3. Win32 VISA Interactive Control: Sending a Command



Controlling the Analyzer with the VXI*plug&play* Driver

The “Programming Examples” CD-ROM that was shipped with this manual includes many example programs that can be used to control your analyzer. The following sections provide some information on using the VXI*plug&play* driver with the Visual C++ and Visual BASIC programming environments.

Using Visual BASIC to Control the Analyzer

When using Visual BASIC, you will need to include the two files listed below in your project. They were installed on your computer in the following directories when you installed the driver:

- \vxipnp\winxx\875x\875x.bas
- \vxipnp\winxx\include\visa32.bas

NOTE The directories shown above are the default locations for these files. (“winxx” indicates the operating system you are using, such as winnt, win95, etc.) If you designated a different path during installation, you will need to amend the path above to include the specific path that you indicated during installation.

Using Visual C++ to Control the Analyzer

When using Visual C++, you will need to include the file listed below in your project. The file was installed on your computer in the following directory when you installed the driver:

\vxipnp\winxx\lib\msc\875x_32.lib

NOTE The directory shown above is the default location for this file. ("winxx" indicates the operating system you are using, such as winnt, win95, etc.) If you designated a different path during installation, you will need to amend the path above to include the specific path that you indicated during installation.

Instrument Control using BASIC

This section describes how to control the analyzer using BASIC 6.2 (or higher), or BASIC for Windows 6.3 (or higher) on one of the following computers:

- HP 9000 Series 200/300
- HP 9000 Series 700 with BASIC-UX
- PC with a GPIB interface card installed

.

Table 1-1. Additional BASIC 6.2 Programming Information

Description	**Agilent Part Number**
BASIC 6.2 Programming Guide	98616-90010
BASIC 6.2 Language Reference (2 Volumes)	98616-90004
Using BASIC for Instrument Control, Volume I	82303-90001
Using BASIC for Instrument Control, Volume II	82303-90002
BASIC for Windows Manual Set	E2060-90100

Table 1-2. Additional GPIB Information

Description	**Agilent Part Number**
BASIC 6.2 Interface Reference	98616-90013
Tutorial Description of the General Purpose Interface Bus	5021-1927

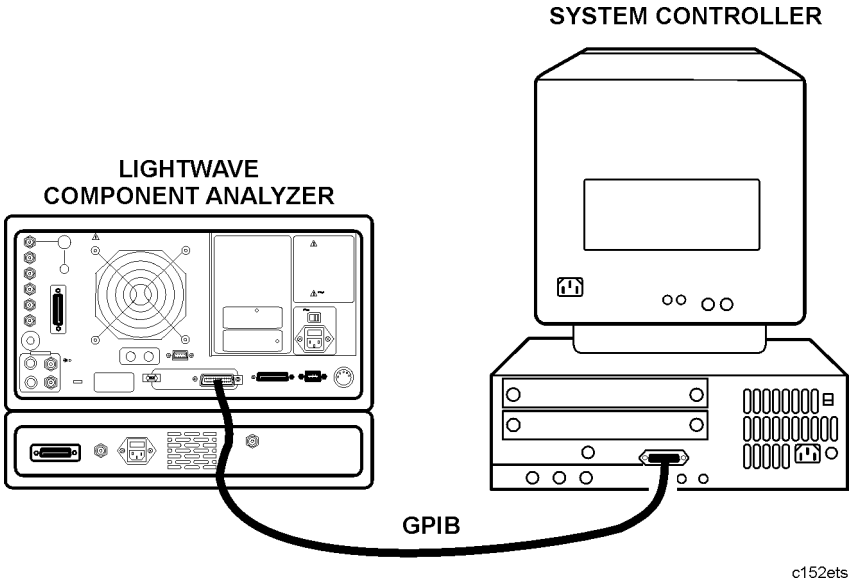
Required Equipment

- Computer running BASIC 6.2 (or higher) or BASIC for Windows 6.3 (or higher)
- Supported GPIB interface card
- GPIB interconnect cables (such as 10833A/B/C/D)

System Setup and GPIB Verification

1. Connect the analyzer to the computer with an GPIB cable.

Figure 1-4. The Analyzer System with Controller



2. Switch on the computer, and launch BASIC or BASIC for Windows.
3. Switch on the analyzer.
   a. To verify the analyzer's address, press:

      Local, SET ADDRESSES, ADDRESS: 8703

      The analyzer has only one GPIB interface, though it occupies two addresses: one for the instrument and one for the display. The display address is equal to the instrument address with the least-significant bit incremented. The display address is automatically set each time the instrument address is set.

      The default analyzer addresses are:

      — 16 for the instrument

      — 17 for the display

CAUTION Other devices connected to the bus cannot occupy the same address as the analyzer.

The analyzer displays the instrument's address in the upper right section of the display. If the address is not 16, return the address to its default setting (16) by pressing:

16, x1, Preset

b. Set the system control mode to either “pass-control” or “talker/listener” mode. These are the only control modes in which the analyzer will accept commands over GPIB. To set the system-control mode, press:

Local, TALKER/LISTENER

or

Local, USE PASS CONTROL

4. Check the interface bus by performing a simple command from the computer controller. Type the following command on the controller:

`OUTPUT 716;"PRES;"` **Execute,** or **Return**

NOTE HP 9000 Series 300 computers use the **Return** key as both execute and enter. Some other computers may have an **Enter, Execute,** or **Exec** key that performs the same function. For reasons of simplicity, the notation **Return** is used throughout this document.

This command should preset the analyzer. If an instrument preset does not occur, there is a problem. Check all GPIB addresses and connections. Most GPIB problems are caused by an incorrect address or faulty/loose GPIB cables.

Sending Commands

A remote controller can manipulate the functions of the analyzer by sending commands to the analyzer via the General Purpose Interface Bus (GPIB). The commands used are specific to the analyzer. Remote commands executed over the bus take precedence over manual commands executed from the instrument's front panel. Remote commands are executed as soon as they are received by the analyzer. A command only applies to the active channel (except in cases where functions are coupled between channels). Most commands are equivalent to front-panel hardkeys and softkeys.

Command Structure in BASIC

Consider the following BASIC command for setting the analyzer's start frequency to 50 MHz:

```
OUTPUT 716;"STAR 50 MHZ;"
```

The command structure in BASIC has several different elements:

the BASIC command statement	`OUTPUT` - The BASIC data-output statement.
the appendage	`716` - The data is directed to interface 7 (GPIB), and on to the device at address 16 (the analyzer). This appendage is terminated with a semicolon. The next appendage is `STAR`, the instrument mnemonic for setting the analyzer's start frequency.
data	`50` - a single operand used by the root mnemonic `STAR` to set the value.
unit	`MHZ` - the units that the operand is expressed in.
terminator	`;` - indicates the end of a command, enters the data, and deactivates the active-entry area.

The `"STAR 50 MHZ;"` command performs the same function as pressing the following keys on the analyzer's front panel:

Start, 50, M/u

`STAR` is the root mnemonic for the start key, 50 is the data, and `MHZ` are the units. Where possible, the analyzer's root mnemonics are derived from the equivalent key label. Otherwise they are derived from the common name for the function. Chapter 2, "Alphabetical Command Reference" lists all the root mnemonics and all the different units accepted.

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

The OUTPUT 716; statement will transmit all items listed (as long as they are separated by commas or semicolons) including:

- literal information enclosed in quotes
- numeric variables
- string variables
- arrays

A carriage return/line feed is transmitted after each item. Again, this can be suppressed by terminating the commands with a semicolon. The analyzer automatically goes into remote mode when it receives an OUTPUT command from the controller. When this happens, the front-panel remote (R) and listen (L) GPIB status indicators illuminate. In remote mode, the analyzer ignores any data that is input with the front-panel keys, with the exception of **Local.** Pressing **Local,** returns the analyzer to manual operation, unless the universal GPIB command LOCAL LOCKOUT 7 has been issued. There are two ways to exit from a local lockout. Either issue the LOCAL 7 command from the controller or cycle the line power on the analyzer.

Setting a parameter such as start frequency is just one form of command the analyzer will accept. It will also accept simple commands that require no operand at all. For example, execute:

```
OUTPUT 716;"AUTO;"
```

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

The analyzer will also accept commands that switch various functions on and off. For example, to switch on dual-channel display, execute:

```
OUTPUT 716;"DUACON;"
```

DUACON is the analyzer root mnemonic for “dual-channel display on.” This causes the analyzer to display both channels. To go back to single-channel display mode, for example, switching off dual-channel display, execute:

```
OUTPUT 716;"DUACOFF;"
```

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

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

```
OUTPUT 716;"auto;"
```

NOTE The analyzer also has a debug mode to aid in troubleshooting systems. When the debug mode is ON, the analyzer scrolls incoming GPIB commands across the display. To manually activate the debug mode, press **Local, GPIB DIAG ON.** To deactivate the debug mode from the controller, execute:

```
OUTPUT 716;"DEBUOFF;"
```

Command Query

Suppose the operator has changed the power level from the front panel. The computer can find

the new power level using the analyzer's command-query function. If a question mark is appended to the root of a command, the analyzer will output the value of that function.

For instance, `POWE 7 DB;` sets the analyzer's output power to 7 dB, and `POWE?;` outputs the current RF output power at the test port to the system controller. For example:

Type `SCRATCH` and press **Return,** to clear old programs.

Type `EDIT` and press **Return,** to access the edit mode. Then type in:

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

NOTE Most commands can also be queried by sending the command (without a value) and then sending the `OUTPACTI` command, as in the following example that queries the power value:

```
10 OUTPUT 716;”POWE;OUTPACTI;”
```

Running the Program The computer will display the preset source-power level in dBm. Change the power level by pressing **Local, Power, XX, x1.** Now run the program again.

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

The ENTER statement takes the stream of binary-data output from the analyzer and reformats it back into numbers and ASCII strings. With the formatting set to its default state, the ENTER statement will format the data into real variables, integers, or ASCII strings, depending on the variable being filled. The variable list must match the data the analyzer has to transmit. If there are not enough variables, data is lost. If there are too many variables for the data available, a BASIC error is generated.

The formatting done by the ENTER statement can be changed. The formatting can be deactivated to allow binary transfers of data. Also, the ENTER USING statement can be used to selectively control the formatting.

ON/OFF commands can be also be queried. The reply is a one (1) if the function is active, a zero (0) if it is not active. Similarly, if a command controls a function that is underlined on the analyzer softkey menu when active, querying that command yields a one (1) if the command is underlined, a zero (0) if it is not. For example, press **Meas.** Though there are seven options on the measurement menu, only one is underlined at a time. The underlined option will return a one (1) when queried.

For instance, rewrite line 10 as:

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

Run the program once and note the result. Then press **Local, Display, DUAL CHAN,** to toggle the display mode, and run the program again.

Another example is to rewrite line 10 as:

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

In this case, the program will display a one (1) if phase is currently being displayed. Since the command only applies to the active channel, the response to the PHAS? inquiry depends on which channel is active.

Operation Complete

Occasionally, there is a need to query the analyzer as to when certain analyzer operations have completed. For instance, a program should not have the operator connect the next calibration standard while the analyzer is still measuring the current one. To provide such information, the analyzer has an “operation complete” reporting mechanism, or OPC command, that will indicate when certain key commands have completed operation. The mechanism is activated by sending either OPC or OPC? immediately before an OPC-compatible command. When the command completes execution, bit 0 of the event-status register will be set. If OPC was queried with OPC?, the analyzer will also output a one (1) when the command completes execution.

As an example, type SCRATCH and press **Return.**

Type EDIT and press **Return.**

Type in the following program:

```
10 OUTPUT 716;"SWET 3 S;OPC?;SING;"
```
Set the sweep time to 3 seconds, and OPC a single sweep.

```
20 DISP "SWEEPING"
30 ENTER 716;Reply
```
The program will halt at this point until the analyzer completes the sweep and issues a one (1).

```
40 DISP "DONE"
50 END
```

Running the Program Running this program causes the computer to display the sweeping message as the instrument executes the sweep. The computer will display DONE just as the instrument goes into hold. When DONE appears, the program could then continue on, being assured that there is a valid data trace in the instrument.

Preparing for Remote (GPIB) Control

At the beginning of a program, the analyzer is taken from an unknown state and brought under remote control. This is done with an abort/clear sequence. ABORT 7 is used to halt bus activity and return control to the computer. CLEAR 716 will then prepare the analyzer to receive commands by:

- clearing syntax errors
- clearing the input-command buffer
- clearing any messages waiting to be output

The abort/clear sequence readies the analyzer to receive GPIB commands. The next step involves programming a known state into the analyzer. The most convenient way to do this is to preset the analyzer by sending the PRES (preset) command. If preset cannot be used, the status-reporting mechanism may be employed. When using the status-reporting register, CLES (Clear Status) can be transmitted to the analyzer to clear all of the status-reporting registers and their enables.

Type SCRATCH and press **Return.**

Type EDIT and press **Return.** Type in the following program:

`10 ABORT 7`	*This halts all bus action and gives active control to the computer.*
`20 CLEAR 716`	*This clears all GPIB errors, resets the GPIB interface, and clears the syntax errors. It does not affect the status-reporting system.*
`30 OUTPUT 716;"PRES;"`	*Presets the instrument. This clears the status-reporting system, as well as resets all of the front-panel settings, except for the GPIB mode and the GPIB addresses.*
`40 END`	*Running this program brings the analyzer to a known state, ready to respond to GPIB control.*

The analyzer will not respond to GPIB commands unless the remote line is asserted. When the remote line is asserted, the analyzer is addressed to listen for commands from the controller. In remote mode, all the front-panel keys are disabled (with the exception of **Local,** and the line-power switch). ABORT 7 asserts the remote line, which remains asserted until a LOCAL 7 statement is executed.

Another way to assert the remote line is to execute:

```
REMOTE 716
```

This statement asserts the analyzer's remote-operation mode and addresses the analyzer to listen for commands from the controller. Press any front-panel key except **Local.** Note that none of the front-panel keys will respond until **Local,** has been pressed.

Local, can also be disabled with the sequence:

```
REMOTE 716
LOCAL LOCKOUT 7
```

After executing the code above, none of the front-panel keys will respond. The analyzer can be returned to local mode temporarily with:

```
LOCAL 716
```

As soon as the analyzer is addressed to listen, it goes back into local-lockout mode. The only way to clear the local-lockout mode, aside from cycling line power, is to execute:

```
LOCAL 7
```

This command un-asserts the remote line on the interface. This puts the instrument into local mode and clears the local-lockout command. Return the instrument to remote mode by pressing:

Local, TALKER/LISTENER

or

Local, USE PASS CONTROL

I/O Paths

One of the features of BASIC is the use of input/output paths. The instrument may be addressed directly by the instrument's device number as shown in the previous examples. However, a more sophisticated approach is to declare I/O paths such as: `ASSIGN @Nwa TO 716`. Assigning an I/O path builds a look-up table in the computer's memory that contains the device-address codes and several other parameters. It is easy to quickly change addresses throughout the entire program at one location. I/O operation is more efficient because it uses a table, in place of calculating or searching for values related to I/O. In the more elaborate examples where file I/O is discussed, the look-up table contains all the information about the file. Execution time is decreased, because the computer no longer has to calculate a device's address each time that device is addressed.

For example:

Type `SCRATCH` and press **Return.**

Type `EDIT` and press **Return.**

Type in the following program:

```
10 ASSIGN @Nwa TO 716      Assigns the analyzer to ADDRESS 716.
20 OUTPUT @Nwa;"STAR 50 MHZ;"    Sets the analyzer's start frequency to 50 MHz.
```

NOTE The use of I/O paths in binary-format transfers allows the user to quickly distinguish the type of transfer taking place. I/O paths are used throughout the examples and are highly recommended for use in device input/output.

2

Alphabetical Command Reference

Alphabetical Command Reference

In this chapter, you can find an alphabetical list and brief descriptions of the supported commands for controlling the Agilent 8703B remotely.

NOTE Some commands have a range of values associated with them. If you send a value that is beyond the analyzer's capability, the analyzer will default to the closest allowed value. Refer to the individual commands for the specific range of values allowed.

Symbol Conventions

<num>	Required numerical data.
<choice1|choice2|…|choice*n*>	An appendage that is part of the command. For example, `FORMAT<DOS|LIF>` indicates that the actual commands are `FORMATDOS` and `FORMATLIF`.
<$>	Indicates a character string operand which must be enclosed by double quotes.
|	An either/or choice in an appendage or optional data.
[ ]	Optional data.

A terminator indicates the end of a command string, and this manual uses a semicolon as the terminator in all syntax examples. The analyzer also interprets line feeds and GPIB end or identify (EOI) messages as terminators. Terminators are not necessary for the analyzer to interpret commands correctly, however in the case of a syntax error, the analyzer will attempt to recover at the next terminator. Therefore, it is recommended that you conclude each command with a terminator.

Because this chapter is an "Alphabetical Command Reference," the commands have been listed alphabetically, rather than by function, in both the "Syntax" sections and the "Description " sections. Therefore, commands grouped together in the "Syntax" sections, are grouped alphabetically and/or due to common syntax form, not necessarily due to common functionality.

The softkeys listed in the "Front Panel Equivalents" tables may not be in the first menu viewed when the associated hardkey is pressed. In many cases, more than one key press will be required to locate the softkey. Refer to your analyzer's reference guide for the exact location of any softkey.

Some commands that do not have an associated query syntax can be queried by sending the command (without a value) and then sending the `OUTPACTI` command, as in the following example that queries the segment power value:

```
10 OUTPUT 716;"SEGPOWER;OUTPACTI;"
```

Many of the commands that *do* have a listed query syntax can also be queried in this manner.

Keys to Programming Commands

Table 2-1. Front Panel Equivalents (1 of 11)

Hardkey	Softkey	Command
Avg	AVERAGING FACTOR	AVERFACT
	AVERAGING <ON |OFF>	AVERO <ON|OFF>
	AVERAGING RESTART	AVERREST
	IF BW [ ]	IFBW
	SMOOTHING APERTURE	SMOOAPER
	SMOOTHING ON OFF	SMOOO <ON|OFF>
Cal	ISOLATION	ISOL
Cal	ALTERNATE A and B	ALTAB
Cal	ALTERNATE RFL/TRAN	ALTAB
Cal	CORRECTION ON OFF	CORR
Cal	DEFINE STANDARD	DEFS
Cal	DONE 1-PORT CAL	SAV1
Cal	DONE 2-PORT CAL	SAV2
Cal	DONE RESP ISOL'N CAL	RAID
Cal	DONE:	DONE
Cal	DONE:	RESPDONE
Cal	EXTENSION PORT 1	PORT1
Cal	EXTENSION PORT 2	PORT2
Cal	EXTENSIONS ON OFF	PORE
Cal	FULL 2-PORT	CALIFUL2
Cal	FWD ISOL'N	FWDI
Cal	FWD MATCH	FWDM
Cal	FWD TRANS	FWDT
Cal	INTERPOL ON OFF	CORI
Cal	ISOL'N STD	RAIISOL
Cal	ISOLATION DONE	ISOD
Cal	MAXIMUM FREQUENCY	MAXF
Cal	OMIT ISOLATION	OMII

Table 2-1. Front Panel Equivalents (2 of 11)

Hardkey	Softkey	Command
Cal	REFLECTION	REFL
Cal	RESPONSE	CALIRESP
Cal	RESPONSE	RAIRESP
Cal	RESPONSE & ISOL'N	CALIRAI
Cal	Response & Match (E/O)	CALIEORM
Cal	Response & Match (O/E)	CALIOERM
Cal	Response & Match: Done	RAMD
Cal	RESUME CAL SEQUENCE	RESC
Cal	REV ISOL'N	REVI
Cal	REV MATCH	REVM
Cal	REV TRANS	REVT
Cal	S11 1-PORT	CALIS111
Cal	S11A	CLASS11A
Cal	S11B	CLASS11B
Cal	S11C	CLASS11C
Cal	S22 1-PORT	CALIS221
Cal	S22A	CLASS22A
Cal	S22A	CLASS22B
Cal	S22A	CLASS22C
Cal	SET Z0	SETZ
Cal	SLIDING LOAD DONE	SLID
Cal	SPECIFY CLASS DONE	CLAD
Cal	standard listed under softkey 1	STANA
Cal	standard listed under softkey 2	STANB
Cal	standard listed under softkey 3	STANC
Cal	standard listed under softkey 4	STAND
Cal	standard listed under softkey 5	STANE
Cal	standard listed under softkey 6	STANF
Cal	standard listed under softkey 7	STANG
Cal	STANDARDS DONE	REFD
Cal	STANDARDS DONE	TRAD

Table 2-1. Front Panel Equivalents (3 of 11)

Hardkey	Softkey	Command
Cal	TESTSET SW n Sweeps	TSSWIn
Cal	TRANSMISSN	TRAN
Cal	VELOCITY FACTOR	VELOFACT
Center		CENT
Chan 1	N/A	CHAN1
Chan 2	N/A	CHAN2
Chan 3	N/A	CHAN3
Chan 4	N/A	CHAN4
Copy	LINE TYPE DATA	LINTDATA
	LINE TYPE MEMORY	LINTMEMO
	LIST VALUES	LISV
	NEXT PAGE	NEXP
	OP PARAMS	OPEP
	PLOT	PLOT
	RESTORE DISPLAY	RESD

Table 2-1. Front Panel Equivalents (4 of 11)

Hardkey	Softkey	Command
Display	2x:[1&2][3&4]	D2XUPCH2
	2x:[1&3][2&4]	D2XUPCH3
	4x:[1][2][3][4]	D4XUPCH2
	4x:[1][3][2][4]	D4XUPCH3
	AUX CHAN ON OFF	AUXC <ON|OFF>
	BACKGROUND INTENSITY	BACI
	BEEP DONE ON OFF	BEEPDONE <ON|OFF>
	BEEP FAIL ON OFF	BEEPFAIL <ON|OFF>
	BEEP WARN ON OFF	BEEPWARN <ON|OFF>
	BRIGHTNESS	CBRI
	DATA ->MEMORY	DATI
	DATA and MEMORY	DISPDATM
	DATA/MEM	DISPDDM
	DATA-MEM	DISPDMM
	DATA+MEM	DISPDPM
	DATA*MEM	DISPDTM
	MEM1/MEM2	DISPM1DM
	MEM1-MEM2	DISPM1MM
	MEM1+MEM2	DISPM1PM
	MEM1*MEM2	DISPM1TM
	MEM2/MEM1	DISPM2DM
	MEM2-MEM1	DISPM2MM
	MEM/DATA	DISPMDD
	MEM-DATA	DISPMMD
	MATH->MEM	MATI
	MEM1->MEM2	MEM1I
	MEM2->MEM1	MEM2I
	DEFAULT COLORS	DEFC
	DISPLAY: DATA	DISPDATA
	DUAL CHAN ON OFF	DUAC <ON|OFF>
	FREQUENCY BLANK	FREO
	INTENSITY	INTE

Table 2-1. Front Panel Equivalents (5 of 11)

Hardkey	Softkey	Command
down	N/A	DOWN
Entry Off	N/A	ENTO
Format	DELAY	DELA
	IMAGINARY	IMAG
	LIN MAG	LINM
	LOG MAG	LOGM
	PHASE	PHAS
	POLAR	POLA
	REAL	REAL
	SMITH CHART	SMIC
	SWR	SWR
Local	ADDRESS: CONTROLLER	ADDRCONT
	ADDRESS: DISK	ADDRDISC
	ADDRESS: P MTR/GPIB	ADDRPOWM
	DISK UNIT NUMBER	DISCUNIT
	GPIB DIAG ON OFF	DEBU <ON|OFF>
	PLTR PORT GPIB	ADDRPLOT
	PRNTR PORT GPIB	ADDRPRIN
	USE PASS CONTROL	USEPASC
Marker	all OFF	MARKOFF
	D MODE OFF	DELO
	D REF = D FIXED MKR	DELRFIXM
	D REF = n	DELR
	FIXED MKR AUX VALUE	MARKFAUV
	FIXED MKR STIMULUS	MARKFSTI
	FIXED MKR VALUE	MARKFVAL
	MARKER n	MARKn
	MKR ZERO	MARKZERO

Table 2-1. Front Panel Equivalents (6 of 11)

Hardkey	Softkey	Command
Marker Fctn	DISP MKRS ON OFF	DISM
	G + jB MKR	SMIMGB
	POLAR LIN MKR	POLMLIN
	SMITH LIN MKR	SMIMLIN
	POLAR LOG MKR	POLMLOG
	SMITH LOG MKR	SMIMLOG
	MARKER -> CENTER	MARKCENT
	MARKER -> DELAY	MARKDELA
	MARKER -> SPAN	MARKSPAN
	MARKER -> START	MARKSTAR
	MARKER -> STOP	MARKSTOP
	MARKERS: CONTINUOUS	MARKCONT
	MARKERS: COUPLED	MARKCOUP
	MARKERS: DISCRETE	MARKDISC
	MARKERS: UNCOUPLED	MARKUNCO
	MEASURE: STATS	MEASTAT
	SMITH R + jX MKR	SMIMRX
	POLAR Re/Im MKR	POLMRI
	SMITH Re/Im MKR	SMIMRI
Marker Search	BANDWIDTH	MARK3DB
	SEARCH LEFT	SEAL
	SEARCH RIGHT	SEAR
	SEARCH: MAX	MARKMAXI
	SEARCH: MAX	SEAMAX
	SEARCH: MIN	MARKMINI
	SEARCH: MIN	SEAMIN
	SEARCH: OFF	SEAOFF
	SEARCH: TARGET	SEATARG
	TRACKING ON OFF	TRACK
	WIDTH VALUE	WIDV
	WIDTHS ON OFF	WIDT

Table 2-1. Front Panel Equivalents (7 of 11)

Hardkey	Softkey	Command
Meas	B	MEASB
	A	MEASA
	A/B	AB
	A/R	AR
	B/R	BR
	CONVERSION 1/S	CONV1DS
	CONVERSION OFF	CONVOFF
	CONVERSION Y:Refl	CONVYREF
	CONVERSION Y:Trans	CONVYTRA
	CONVERSION Z:Refl	CONVZREF
	CONVERSION Z:Trans	CONVZTRA
	E/O Trans	MEASEO1
	O/E Trans (Port 1)	MEASOE1
	O/E Trans (Port 2)	MEASOE2
	O Refl	MEASO1
	O Trans	MEASO01
	R	MEASR
	Refl:FWD S11 (A/R)	S11
	Refl:REV S22 (B/R)	S22
	Trans:FWD S21 (B/R)	S21
	Trans:REV S12 (A/R)	S12
Power	N/A	POWE
	PORT POWER	PORTP
	PWR RANGE AUTO MAN	PWRR <PMAN|PAUTO>
	RANGE n	PRAN
	SOURCE PWR ON OFF	SOUP <ON|OFF>
Preset	N/A	PRES
	N/A	RST

Table 2-1. Front Panel Equivalents (8 of 11)

Hardkey	Softkey	Command
Save/Recall	CLEAR	CLEAREG <01-31>
	CLEAR ALL	CLEARALL
	DATA ARRAY ON OFF	EXTMDATA
	DATA ONLY	EXTMDATO
	FILE NAME	TITF
	FORMAT ARY ON OFF	EXTMFORM
	FORMAT INT DISK	INID
	GRAHPICS ON OFF	EXTMGRAP
	INTERNAL DISK	INTD
	LOAD	LOAD
	RAW ARRAY ON OFF	EXTMRAW
	READ FILE TITLES	REFT
	SAVE	SAVEREG
	SAVE FILE when GRAPH FMT [ ] is set to CSV and FILETYPE: GRAPHIC is selected.	SAVECSV
	SAVE FILE when GRAPH FMT [ ] is set to JPG and FILETYPE: GRAPHIC is selected.	SAVEJPG
	SAVE USING ASCII	SAVUASCI
	SAVE USING BINARY	SAVUBINA
	TITLE	TITREG
Scale Ref	AUTOSCALE	AUTO
	ELECTRICAL DELAY	ELED
	MARKER -> REFERENCE	MARKREF
	PHASE OFFSET	PHAO
	REFERENCE POSITION	REFP
	REFERENCE VALUE	REFV
	SCALE / DIV	SCAL
Span	N/A	SPAN
Start	N/A	STAR
Stop	N/A	STOP

Table 2-1. Front Panel Equivalents (9 of 11)

Hardkey	Softkey	Command
Sweep Setup	ALL SEGS SWEEP	ASEG
	CONTINUOUS	CONT
	CONTINUOUS	FRER
	COUPLED CH ON OFF	COUC
	CW FREQ	CWFREQ
	CW TIME	CWTIME
	EXT TRIG ON SWEEP	EXTTON
	HOLD	HOLD
	LIN FREQ	LINFREQ
	LIST FREQ	LISFREQ
	LIST IF BW ON OFF	LISIFBWM
	LIST POWER ON OFF	LISPWRM
	LIST TYPE: STEPPED	LISTTYPELSTP
	LIST TYPE: SWEPT	LISTTYPELSWP
	LOG FREQ	LOGFREQ
	MANUAL TRG ON POINT	MANTRIG
	MEASURE RESTART	REST
	NUMBER of GROUPS	NUMG
	NUMBER of POINTS	POIN <NUM>
	POWER SWEEP	POWS
	SEGMENT IF BW	SEGIFBW
	SEGMENT POWER	SEGPOWER
	SINGLE	SING
	SINGLE SEG SWEEP	SSEG
	STEP SIZE	STPSIZE
	SWEEP TIME AUTO	SWEA
	SWEEP TIME MANUAL	SWET <NUM>
	TRIGGER: TRIG OFF	EXTTOFF
	EDIT LIST	EDITLIST
Sweep Setup or Cal	CLEAR LIST YES	CLEL
Sweep Setup or System	STEP SWP ON OFF	STEPSWP ON|OFF>

Table 2-1. Front Panel Equivalents (10 of 11)

Hardkey	Softkey	Command
System	AMPLITUDE OFFSET	LIMIAMPO
	BEEP FAIL ON OFF	BEEPFAIL ON|OFF>
	BW DISPLAY on OFF	BWLIMDISP ON|OFF>
	BW TEST on OFF	BWLIMTEST ON|OFF>
	CLEAR LIST	CLER
	CLEAR LIST YES	CLEAL
	DELTA LIMITS	LIMD
	DEMOD: AMPLITUDE	DEMOAMPL
	DEMOD: OFF	DEMOOFF
	DEMOD: PHASE	DEMOPHAS
	EDIT LIMIT LINE	EDITLIML
	EDIT RIPL LIMIT	EDITRLIM
	FIRMWARE REVISION	SOFR
	FLAT LINE	LIMTFL
	FREQUENCY BAND	SELBND
	HARMONIC OFF	HARMOFF
	HARMONIC SECOND	HARMSEC
	HARMONIC THIRD	HARMTHIR
	LIMIT LINE ON OFF	LIMILINE
	LIMIT TEST ON OFF	LIMITEST
	LOWER LIMIT	LIML
	MARKER -> AMP. OFS.	LIMIMAOF
	MARKER -> CW	MARKCW
	MARKER -> MIDDLE	MARKMIDD
	MARKER -> STIMULUS	MARKSTIM
	MAXIMUM BANDWIDTH	BWLIMMAX
	MAXIMUM FREQUENCY	RLIMSTP
	MAXIMUM RIPPLE	RLIMM
	MIDDLE VALUE	LIMM
	MINIMUM BANDWIDTH	BWLIMMIN
	MINIMUM FREQUENCY	RLIMSTR
	N DB POINTS	BWLIMDB

Table 2-1. Front Panel Equivalents (11 of 11)

Hardkey	Softkey	Command
System or Sweep Setup or Cal	SEGMENT DELETE	SDEL
	DONE	EDITDONE
	DONE	SDON
	EDIT SEGMENT	SEDI
	SEGMENT ADD	SADD
up	N/A	UP

Programming Commands

AB

AB; *or* AB?;

Command	Description	Range	Query Response
AB	Measures and displays A/B on the active channel.	N/A	<0|1><L_F>

ADDR

ADDR<CONT|DISC|PLOT|POWM|PRIN><num>; *or* ADDR<CONT|DISC|PLOT|POWM|PRIN>?;

Sets the GPIB address for the following peripherals.

Command	Description	Range	Query Response
ADDRCONT	Controller GPIB address. The address where control is returned after a pass control.	integers 0–30	<num><L_F>
ADDRDISC	External disk drive GPIB address.	integers 0–30	<num><L_F>
ADDRPLOT	Plotter GPIB address.	integers 0–30	<num><L_F>
ADDRPOWM	Power meter GPIB address.	integers 0–30	<num><L_F>
ADDRPRIN	Printer GPIB address.	integers 0–30	<num><L_F>

ADJB

ADJB

Executes autobiasing of optical modulator. No query response

ALTAB

ALTAB; *or* ALTAB?;

Command	Description	Range	Query Response
ALTAB	Places the analyzer in the alternate inputs measurement mode, where A and B measurements are made on alternate sweeps. See also “CHOPAB.”	N/A	<0|1><L_F>

AR

AR; *or* AR?;

Command	Description	Range	Query Response
AR	Measures and displays A/R on the active channel.	N/A	<0|1><L_F>

ASEG

ASEG; *or* ASEG?;

Command	Description	Range	Query Response
ASEG	Uses all segments for list frequency sweep. See also "SSEG"	N/A	<0|1><L_F>

AUTB;

AUTB<ON|OFF>;

Enable or disable autobiasing of optical modulator.

AUTO

AUTO;

Command	Description	Range	Query Response
AUTO	Auto scale the active channel.	N/A	N/A

AUXC

AUXC<ON|OFF>; *or* AUXC?;

Command	Description	Range	Query Response
AUXC	Enables and disables auxiliary channels 3 and 4. OPC-compatible.	N/A	<0|1><L_F>

Example

```
10   OUTPUT 716;"CHAN1;AUXCON;"        Turns on channel 3
20   OUTPUT 716;"CHAN2;AUXCON;"        Turns on channel 4
```

AVER

AVERFACT<num>; *or* AVERFACT?;
AVERO<ON|OFF>; *or* AVERO?;
AVERREST;

Command	Description	Range	Query Response
AVERFACT	Sets the averaging factor on the active channel.	integers 0–999	<num><L_F>
AVERO	Turns averaging on and off on the active channel.	N/A	<0|1><L_F>
AVERREST	Restarts the averaging on the active channel.	N/A	N/A

BACI

BACI<num>; *or* BACI?;

Command	Description	Range	Query Response
BACI	Sets the background intensity of the display.	integers 0–100	<num><L_F>

BEEP

BEEP<DONE|WARN|FAIL><ON|OFF>; *or* BEEP<DONE|WARN|FAIL>?;

Command	Description	Range	Query Response
BEEPDONE	Causes the analyzer's warning beeper to sound at the completion of functions such as save, done with calibration standard, and data trace saved.	N/A	<0|1><L_F>
BEEPFAIL	Causes the analyzer's warning beeper to sound in the event of a limit test failure.	N/A	<0|1><L_F>
BEEPWARN	Causes the analyzer's warning beeper to sound when a warning message is generated.	N/A	<0|1><L_F>

BR

BR; *or* BR?;

Command	Description	Range	Query Response
BR	Measures and displays B/R on the active channel.	N/A	<0|1><L_F>

BSAMP

BSAMP<ON|OFF>;

Switch B, sampler to : ON = LW, OFF = RF.

BWLIMDB

BWLIMDB<num>; *or* BWLIMDB?;

Command	Description	Range	Query Response
BWLIMDB	Enters the *N dB Point*, the amplitude below the peak that is used to measure the filter's bandwidth.	-500 to 500 dB	<num><$^{L}_{F}$>

BWLIMDISP

BWLIMDISP<ON|OFF>; *or* BWLIMDISP?;

Command	Description	Range	Query Response
BWLIMDISP	Turns the measured bandwidth value in the upper left corner of the display on and off. The measured bandwidth value is displayed near the bandwidth Pass/Wide/Narrow message.	N/A	<0|1><$^{L}_{F}$>

BWLIMMKR

BWLIMMKR<ON|OFF>; *or* BWLIMMKR?;

Command	Description	Range	Query Response
BWLIMMKR	Turns the limit bandwidth marker on and off.	N/A	<0|1><$^{L}_{F}$>

BWLIMMAX

BWLIMMAX<num>[HZ|KHZ|MHZ|GHZ]; *or* BWLIMMAX?;

Command	Description	Range	Query Response
BWLIMMAX	Enters the maximum bandwidth value. If the measured bandwidth is greater than this value, the filter fails the bandwidth test.	stimulus range[a]	<num><$^{L}_{F}$>

a. Refer to "Preset State and Memory Allocation" in your analyzer's reference guide.

BWLIMMIN

BWLIMMIN<num>[HZ|KHZ|MHZ|GHZ]; *or* BWLIMMIN?;

Command	Description	Range	Query Response
BWLIMMIN	Enters the minimum bandwidth value. If the measured bandwidth is less than this value, the filter fails the bandwidth test.	stimulus range[a]	<num><$^{L}_{F}$>

a. Refer to “Preset State and Memory Allocation” in your analyzer’s reference guide.

BWLIMSTAT

BWLIMSTAT;

Command	Description	Range	Response
BWLIMSTAT	Returns the results of the bandwidth test. A returned value of 0 indicates that the filter passed the bandwidth test. A returned value of –1 indicates that the filter failed the bandwidth test because it is narrower than the bandwidth limit. A returned value of 1 indicates that the filter failed the bandwidth test because it is wider than the bandwidth limit.	N/A	<–1|0|1><$^{L}_{F}$>

BWLIMTEST

BWLIMTEST<ON|OFF>; *or* BWLIMTEST?;

Command	Description	Range	Query Response
BWLIMTEST	Turns the bandwidth test on and off.	N/A	<0|1><$^{L}_{F}$>

BWLIMVAL

BWLIMVAL;

Command	Description	Range	Response
BWLIMVAL	Returns the measured bandwidth value.	N/A	<num><$^{L}_{F}$>

CALI

CALI<EORM|OERM|FUL2|RAI|RESP|S111|S221>;

CALI<EORM|OERM|FUL2|RAI|RESP|S111|S221>?;

Command	Description	Range	Query Response
CALIEORM	Select E/O response and match calibration.		<011>
CALIOERM	Select O/E response and match calibration.		<011>
CALIFUL2[a]	Begins the sequence for a short, load, open, thru (SLOT) 2-port calibration.	N/A	<0|1><L_F>
CALIRAI	Begins the sequence for a response and isolation calibration.	N/A	<0|1><L_F>
CALIRESP	Begins the sequence for a response calibration.	N/A	<0|1><L_F>
CALIS111	Begins the sequence for an S11 1-port calibration (ES models), or a reflection 1-port calibration (ET models).	N/A	<0|1><L_F>
CALIS221[a]	Begins the sequence for an S22 1-port calibration.	N/A	<0|1><L_F>

a. The result of the query command only tells if the particular type of calibration is currently active. It does not provide information on the status of the cal sequence.

CALK35MM

CALK35MM;

Command	Description	Range	Query Response
CALK35MM	Selects the 3.5mm calibration kit coefficients.	N/A	N/A

CBRI

CBRI<num>; *or* CBRI?;

Command	Description	Range	Query Response
CBRI	Adjusts the color brightness of the selected display feature.	integers 0–100	<num><L_F>

CENT

CENT<num>[HZ|DB]; *or* CENT?;

Command	Description	Range	Query Response
CENT	Sets the center stimulus value. If a list frequency segment is being edited, sets the center of the list segment.	stimulus range[a]	<num><$^{L}_{F}$>

a. For frequency or power sweeps, refer to “Preset State and Memory Allocation,” in your analyzer’s reference guide. For CW time: 0 to 24 hours. For frequency sweep, transform on: ± 1/frequency step. For CW time sweep, transform on: ±1/time step.

CHAN

CHAN<1|2|3|4>;

Makes channel 1, 2, 3, or 4 the active channel. OPC-compatible. No query response.

NOTE These commands should use OPC? to prevent timing errors with subsequent commands. Example code written in BASIC:

```
10 OUTPUT 716;"OPC?;CHAN2;"
20 ENTER 716;OPC
```

CLAD

CLAD;

Command	Description	Range	Query Response
CLAD	Class done (modify cal kit, specify class).	N/A	N/A

CLASS

CLASS<11A|11B|11C|22A|22B|22C>;

These commands call reflection standard classes during a calibration sequence. If only one standard is in the class, it is measured. If there is more than one, the standard being used must be selected with STAN<A|B|C|D|E|F|G>. If there is only one standard in the class, these commands are OPC-compatible.

Command	Description	Range	Query Response
CLASS11A	S11A: S11 (forward reflection) 1-port, open	N/A	N/A
CLASS11B	S11B: S11 (forward reflection) 1-port, short	N/A	N/A
CLASS11C	S11C: S11 (forward reflection) 1-port, load	N/A	N/A
CLASS22A	S22A: S22 (reverse reflection) 1-port, open	N/A	N/A
CLASS22B	S22B: S22 (reverse reflection) 1-port, short	N/A	N/A
CLASS22C	S22C: S22 (reverse reflection) 1-port, load	N/A	N/A

EXAMPLE To measure the female open of a type-N cal kit:

```
OUTPUT 716;"CLASS11A;OPC?;STANB;"
ENTER 716 OPC;
```

CLEAREG

CLEAREG<num>;

CLEARALL;

Command	Description	Range	Query Response
CLEAREG	Clears save/recall registers 01 through 31. CLEAREG01 through CLEAREG05 are the same as CLEA1 through CLEA5. OPC-compatible.	two-digit integers 01–31	N/A
CLEARALL	Clears all the save/recall registers. OPC-compatible.	N/A	N/A

CLEAL

CLEAL;

Command	Description	Range	Query Response
CLEAL	Clears the limit line list. Should be preceded by EDITLIML.	N/A	N/A

CLEL

CLEL;

Clears the currently selected list. This could be a frequency list, power loss list, or limit test list. Must be preceded by an "EDIT" command. No query response.

CLES

CL[E]S;

Command	Description	Range	Query Response
CLES	Clears the status byte register, the event-status registers, and the enable registers.	N/A	N/A
CLS	Same as CLES.	N/A	N/A

CLER

CLER;

Clears (or deletes) the all of existing ripple test limits. No query response.

COEFA

COEFA<1-4> [value]

Set numerator coefficients of response model.

COEFB

COEFB<1-4> [value]

Set denominator coefficients of response model.

COEFDELA

COEFDELA [value];

Set delay coefficient of response model.

COEFK;

COEFK;

Sets constant coefficient of response model.

CONS

CONS;

Continues the sequence that was paused.

CONT

CONT; *or* CONT?;

Command	Description	Range	Query Response
CONT	Places the analyzer in continuous sweep trigger mode.	N/A	<0|1><L_F>

CONV

CONV<1DS|OFF|YREF|YTRA|ZREF|ZTRA>; *or* CONV<1DS|OFF|YREF|YTRA|ZREF|ZTRA>?;

These 6 commands convert the S-parameter data to:

Command	Description	Range	Query Response
CONV1DS	Inverted S-parameters.	N/A	<0|1><L_F>
CONVOFF	Turns S-parameter conversion off.	N/A	<0|1><L_F>
CONVYREF	Y:reflection (admittance).	N/A	<0|1><L_F>
CONVYTRA	Y:transmission (transmission).	N/A	<0|1><L_F>
CONVZREF	Z:reflection (impedance).	N/A	<0|1><L_F>
CONVZTRA	Z:transmission (transmission).	N/A	<0|1><L_F>

CORI

CORI<ON|OFF>; *or* CORI?;

Command	Description	Range	Query Response
CORI	Turns interpolative error correction on and off.	N/A	<0|1><L_F>

CORR

CORR<ON|OFF>; *or* CORR?;

Command	Description	Range	Query Response
CORR	Turns error correction on and off.	N/A	<0|1><L_F>

COU

COU<C><ON|OFF>; *or* COU<C>?;

Command	Description	Range	Query Response
COUC	Couples and uncouples the stimulus between the channels.	N/A	<0|1><L_F>

COUS

COUS<ON|OFF>;

Switch coupling to measurement parameter ON or OFF.

CWFREQ

CWFREQ<num>[HZ|DB]; *or* CWFREQ?;

Command	Description	Range	Query Response
CWFREQ	Sets the CW frequency for power sweep and CW frequency modes. While the list frequency table segment is being edited, it sets the center frequency of the current segment. See also "MARKCENT."	stimulus range[a]	<num><L_F>

a. For frequency or power sweeps, refer to "Preset State and Memory Allocation," in the analyzer's reference guide. For CW time: 0 to 24 hours. For frequency sweep, transform on: ±1/frequency step. For CW time sweep, transform on: ±1/time step.

CWTIME

CWTIME; *or* CWTIME?;

Command	Description	Range	Query Response
CWTIME	Selects CW time as the sweep type.	N/A	<0|1><L_F>

D2XUPCH

D2XUPCH<2|3>; *or* D2XUPCH<2|3>?;

Command	Description	Range	Query Response
D2XUPCH2	Sets up a two-graticule display with channel 2 on top.	N/A	<0|1><L_F>
D2XUPCH3	Sets up a two-graticule display with channel 3 on top.	N/A	<0|1><L_F>

D4XUPCH

D4XUPCH<2|3>; *or* D4XUPCH<2|3>?;

Command	Description	Range	Query Response
D4XUPCH2	Sets up a four-graticule display with channel 2 in the upper right quadrant of the display.	N/A	<0|1><L_F>
D4XUPCH3	Sets up a four-graticule display with channel 3 in the upper right quadrant of the display.	N/A	<0|1><L_F>

DATI

DATI;

Stores the data trace in channel memory. OPC-compatible. No query response.

DEBU

DEBU<ON|OFF>; *or* DEBU?;

Command	Description	Range	Query Response
DEBU	Turns the GPIB debug mode on and off. When on, the analyzer scrolls incoming GPIB commands across the display.	N/A	<0|1><L_F>

DEFC

DEFC;

Sets the default colors for all display features. No query response.

DEFS

DEFS<num>;

Command	Description	Range	Query Response
DEFS	Begins standard definition during cal kit modification. "<num>" is the standard number.	integers 1–8	N/A

DEL

DEL<O|RFIXM>; *or* DEL<O|RFIXM>?;

DELR<num>; *or* DELR<num>?;

Command	Description	Range	Query Response
DELO	Turns delta marker mode off.	N/A	<0|1><L_F>
DELR	Makes the indicated marker the delta reference.	integers 1–5	<0|1><L_F>
DELRFIXM	Makes the fixed marker the delta reference.	N/A	<0|1><L_F>

DELA

DELA; *or* DELA?;

Command	Description	Range	Query Response
DELA	Displays the data formatted as group delay.	N/A	<0|1><L_F>

DEMO

DEMO<AMPL|OFF|PHAS>; *or* DEMO<AMPL|OFF|PHAS>?;

Command	Description	Range	Query Response
DEMOAMPL	Turns on transform demodulation and sets the transform demodulation to amplitude demodulation. Only has a meaningful effect with a CW time transform.	N/A	<0|1><L_F>
DEMOOFF	Turns the transform demodulation function off.	N/A	<0|1><L_F>
DEMOPHAS	Sets the transform demodulation to phase demodulation. Only has a meaningful effect with a CW time transform.	N/A	<0|1><L_F>

DISC

DISC<UNIT><num>; *or* DISC<UNIT>?;

Command	Description	Range	Query Response
DISCUNIT	Specifies which disk in an external multiple-disk drive to be used for save/recall.	integers 0–30	<num><L_F>

DISM

DISM<ON|OFF>; *or* DISM?;

Command	Description	Range	Query Response
DISM	When on, displays the response and stimulus values for all markers that are turned on; when off, only the active marker's value is displayed.	N/A	<0|1><L_F>

DISP

DISP<DATA|DATM|DDM|DMM|MEMO|DPM|DTM|M1DM|M1MM|M1PM|M1TM|M2DM|M2MM|MATH|MMD

|MDD>;

DISP<DATA|DATM|DDM|DMM|MEMO|DPM|DTM|M1DM|M1MM|M1PM|M1TM|M2DM|M2MM|MATH|MMD|MDD>?;

These commands display the indicated combinations of data and trace memory on the active channel.

Command	Description	Range	Query Response
DISPDATA	Data only.	N/A	<0|1><L_F>
DISPDATM	Data and memory.	N/A	<0|1><L_F>
DISPDDM	Data divided by memory (linear division, log subtraction). See also "DIVI."	N/A	<0|1><L_F>
DISPDMM	Data minus memory (linear subtraction). See also "MINU."	N/A	<0|1><L_F>
DISPMEMO	Memory only.	N/A	<0|1><L_F>
DISPDPM	Display data plus memory.	N/A	<0|1><L_F>
DISPDTM	Display data times memory.	N/A	<0|1><L_F>
DISPM1DM	Display memory 1 divided by memory 2.	N/A	<0|1><L_F>
DISPM1MM	Display memory 1 minus by memory 2	N/A	<0|1><L_F>
DISPM1PM	Display memory 1 plus memory 2.	N/A	<0|1><L_F>
DISPM1TM	Display memory 1 times memory 2.	N/A	<0|1><L_F>
DISPM2DM	Display memory 2 divided by memory 1.	N/A	<0|1><L_F>
DISPM2MM	Display memory 2 minus memory 1.	N/A	<0|1><L_F>
DISPMATH	Display math results	N/A	<0|1><L_F>
DISPMMD	Display memory minus data	N/A	<0|1><L_F>
DISPMDD	Display memory divided by data	N/A	<0|1><L_F>

DONE

DONE;

Done with a class of standards, during a calibration. Only needed when multiple standards are measured to complete the class. OPC-compatible. No query response.

DONM

DONM;

Done modifying a test sequence. No query response.

DOSEQ

```
DOSEQ<1-6>;
```

Start sequence 1-6. No query response.

DOWN

```
DOWN;
```

Decrements the value displayed in the active entry area (emulates pressing the down-arrow key). No query response.

DRIVPORT

DRIVPORT<ON|OFF>;

Drive port; ON = LW, OFF = RF.

DUAC

DUAC<ON|OFF>; *or* DUAC?;

Command	Description	Range	Query Response
DUAC	Turns dual channel display on and off.	N/A	<0|1><L_F>

EDIT

EDIT<DONE|LIML|LIST>;

Command	Description	Range	Query Response
EDITDONE	Done editing list frequency, limit table, cal sensor table, or power loss list. OPC-compatible.	N/A	N/A
EDITLIML	Begins editing limit table.	N/A	N/A
EDITLIST	Begins editing list frequency table.	N/A	N/A

EDITRLIM

EDITRLIM;

Begins the editing of the ripple limit list. No query response.

ELED

ELED<num>[S]; *or* ELED?;

Command	Description	Range	Query Response
ELED	Sets the electrical delay offset.	±10 seconds	<num><L_F>

ENTO

ENTO;

Command	Description	Range	Query Response
ENTO	Removes displayed information from the active entry area on the screen.	N/A	N/A

EOCAL

EOCAL;

Internal E/O service calibration parameter.

ESE

ESE<num>; *or* ESE?;

Command	Description	Range	Query Response
ESE	Enables the selected event-status register bits to be summarized by bit 5 in the status byte. An event-status register bit is enabled when the corresponding bit in the operand <num> is set.	integers 0–255	<num><L_F>

ESNL

ESNL<num>; *or* ESNL?;

Command	Description	Range	Query Response
ESNL	Enables the selected event-status register L bits to be summarized by bit 2 in the status byte. An event-status register bit is enabled when the corresponding bit in the operand <num> is set.	integers 0–4095	<num><L_F>

ESL?

ESL?;

Command	Description	Range	Query Response
ESL?	Query only. Outputs event-status register.	N/A	<num><L_F>

EXTM

EXTM<DATA|FORM|GRAP|RAW><ON|OFF>; *or* EXTM<DATA|FORM|GRAP|RAW>?;

These commands include the indicated information when an instrument state is stored to the internal floppy disk drive or an external disk.

Command	Description	Range	Query Response
EXTMDATA	Adds error corrected data (real and imaginary pairs) along with the other files.[a]	N/A	<0|1><L_F>
EXTMFORM	Formatted trace data. Uses currently selected format for data.	N/A	<0|1><L_F>
EXTMGRAP	User graphics.	N/A	<0|1><L_F>
EXTMRAW	Raw data arrays (real and imaginary pairs).	N/A	<0|1><L_F>

a. See Figure 6-1 on page 6-3. This error corrected data is the same as that output by the OUTPDATA command.

EXTT

EXTT<ON|OFF>; *or* EXTT?;

Command	Description	Range	Query Response
EXTT	Activates or deactivates the external trigger mode. OPC-compatible.	N/A	<0|1><L_F>

FORM

FORM<1|2|3|4|5>;

These 5 commands set the data format for array transfers in and out of the instrument:

Command	Description	Range	Query Response
FORM1	The analyzer's internal binary format, 6 bytes-per-data point. The array is preceded by a four-byte header. The first two bytes represent the string "#A", the standard block header. The second two bytes are an integer representing the number of bytes in the block to follow. FORM1 is best applied when rapid data transfers, not to be modified by the computer nor interpreted by the user, are required.	N/A	N/A
FORM2	IEEE 32-bit floating-point format, 4 bytes-per-number, 8 bytes-per-data point. The data is preceded by the same header as in FORM1. Each number consists of a 1-bit sign, an 8-bit biased exponent, and a 23-bit mantissa. FORM2 is the format of choice if your computer is not a PC, but supports single-precision floating-point numbers.	N/A	N/A

Command	Description	Range	Query Response
FORM3	IEEE 64-bit floating-point format, 8 bytes-per-number, 16 bytes-per-data point. The data is preceded by the same header as in FORM1. Each number consists of a 1-bit sign, an 11-bit biased exponent, and a 52-bit mantissa. This format may be used with double-precision floating-point numbers. No additional precision is available in the analyzer data, but FORM3 may be a convenient form for transferring data to your computer.	N/A	N/A
FORM4	ASCII floating-point format. The data is transmitted as ASCII numbers, as described in "Output Syntax" on page 5-4. There is no header. The analyzer always uses FORM4 to transfer data that is not related to array transfers (i.e. marker responses and instrument settings). Data is comma delimited.	N/A	N/A
FORM5	PC-DOS 32-bit floating-point format with 4 bytes-per-number, 8 bytes-per-data point. The data is preceded by the same header as in FORM1. The byte order is reversed with respect to FORM2 to comply with PC-DOS formats. If you are using a PC-based controller, FORM5 is the most effective format to use.	N/A	

FREO

```
FREO;
```

Frequency blank. Turns frequency notation off. Once the frequency notation has been turned off (blanked), it cannot be turned back on until a preset or recall is initiated. No query response.

FRER

```
FRER; or FRER?;
```

Command	Description	Range	Query Response
FRER	Places the analyzer in GPIB free run mode. (Same as continuous sweep trigger mode.) See "CONT."	N/A	<0|1><L_F>

FWD

```
FWD<I|M|T>;
```

These commands are OPC-compatible if there is only one standard in the class. If there is just one standard, that standard is measured automatically. If there is more than one standard in the class,

the standard being used must be selected with the STAN command.

Command	Description	Range	Query Response
FWDI	Selects the forward isolation calibration class during a 2-port calibration sequence.	N/A	N/A
FWDM	Selects the forward match calibration class during a 2-port calibration sequence.	N/A	N/A
FWDT	Selects the forward transmission calibration class during a 2-port calibration sequence.	N/A	N/A

HOLD

`HOLD;` *or* `HOLD?;`

Command	Description	Range	Query Response
HOLD	Puts the sweep trigger into hold mode.	N/A	<0|1><L_F>

IDN?

`IDN?;`

Command	Description	Range	Query Response
IDN?	Query only. Outputs the identification string: where 87NNEX is the model number of the instrument, xxxxxxxxxx is the serial number of the instrument, and X.XX is the firmware revision of the instrument.	N/A	See command description

IFBW

`IFBW<num>[HZ];` *or* `IFBW?;`

Command	Description	Range	Query Response
IFBW	Sets the IF bandwidth.	Choose from 10, 30, 100, 300, 1000, 3000	<num><L_F>

IMAG

`IMAG;` *or* `IMAG?;`

Command	Description	Range	Query Response
IMAG	Selects the imaginary display format.	N/A	<0|1><L_F>

INI

INI<D>;

Initializes the internal disk. All previous information on the disk will be destroyed. No query response.

INPU

INPUCALC<num><array>;

INPU<CALK|DATA|FORM><array>;

INPULEAS<learnstring>; *or* INPULEAS?;

INPURAW<1|2|3|4><array>;

All of these commands (with a few noted exceptions) input an array and require that you set the format for data transfers with the FORM command. All of these commands have an associated OUTPut command that is used to transfer data from the analyzer. See "OUTP," later in this chapter.

Command	Description	Range	Query Response
INPUCALC	Error coefficient array[a] <num>.	two-digit integers 01–12	N/A
INPUCALK[b]	Inputs a cal kit array in FORM1 only. Can be read out with the OUTCALK command. After the transfer, the data should be saved into the user cal kit area with the SAVEUSEK command.	N/A	N/A
INPUDATA	Inputs an error corrected data array, using the current setting of the FORM command.	N/A	N/A
INPUFORM	Inputs a formatted data array, using the current setting of the FORM command.	N/A	N/A
INPULEAS[b]	Inputs a learn string in FORM1 only. Can be read out with the OUTPLEAS command, or with INPULEAS?.	N/A	<data><L_F>
INPURAW1	Inputs raw data array 1 (S_{11} data). After the data is received, the analyzer stops sweeping, error-corrects the data, then formats and displays the data.	N/A	N/A
INPURAW2	Inputs raw data array 2 (S_{21} data). After the data is received, the analyzer stops sweeping, error-corrects the data, then formats and displays the data.	N/A	N/A
INPURAW3	Inputs raw data array 3 (S_{12} data). After the data is received, the analyzer stops sweeping, error-corrects the data, then formats and displays the data.	N/A	N/A
INPURAW4	Inputs raw data array 4 (S_{22} data). After the data is received, the analyzer stops sweeping, error-corrects the data, then formats and displays the data.	N/A	N/A

a. These commands input an individual error coefficient array. Before sending an array, issue a CALIXXXX; command, where XXXX specifies the calibration type. (See "CALI" earlier in this book.) Then input the array or arrays. Lastly store the data with the SAVC command. The instrument goes into hold, displaying uncorrected data. Complete the process by triggering a sweep, with the CONT command (for continuous sweep) or the SING command (for a single sweep). See Table 2-1 on page 2-37 for the contents of the different arrays.
b. Does not require a preceding "FORM" command.

Table 2-1. Error Coefficient Arrays

Array	Response	Response & Isolation	1-port	Enhanced Response	2-port [a]	TRL/ LRM	O/E Response & Match	E/O Response & Match
01	E_R or E_T	E_X (E_D)[b]	E_D	E_D	E_{DF}	E_{DF}	E_{DR}	E_{DI}
02		E_T (E_R)	E_S	E_S	E_{SF}	E_{SF}	E_{SR}	E_S
03			E_R	E_R	E_{RF}	E_{RF}	E_{RR}	E_R
04				E_X	E_{XF}	E_{XF}	E_{XF}	E_X
05				E_L[c]	E_{LF}	E_{LF}	E_{LF}	E_T
06				E_T	E_{TF}	E_{TF}	E_{TF}	
07					E_{DR}	E_{DR}	E_{DF}	
08					E_{SR}	E_{SR}	E_{SF}	
09					E_{RR}	E_{RR}	E_{RF}	
10					E_{XR}	E_{XR}		
11					E_{LR}	E_{LR}		
12					E_{TR}	E_{TR}		

a. One path, 2-port cal duplicates arrays 1 to 6 in arrays 7 to 12.
b. Response and isolation corrects for crosstalk and transmission tracking in transmission measurements, and for directivity and reflection tracking in reflection measurements.
c. This term is used to generate the calibration coefficients, but is not used during measurement error correction.

Meaning of first subscript:
D: directivity
S: source match
R: reflection tracking
X: crosstalk or isolation
L: load match
T: transmission tracking

Meaning of second subscript:
F: forward
R: reverse

INT

```
INT<D>;
```

Selects the internal disk as the active storage device. No query response.

INTE

INTE<num>; *or* INTE?;

Command	Description	Range	Query Response
INTE	Sets the display intensity, 50 to 100 percent.	integers 50–100	<num><$^{L}_{F}$>

ISO

ISO<D|L>;

Command	Description	Range	Query Response
ISOD	Done with isolation subsequence in a 2-port or enhance response calibration. OPC-compatible.	N/A	N/A
ISOL	Begins the isolation subsequence step in a 2-port calibration.	N/A	N/A

LIM

LIM<D|L|M|S|U><num>[DB|HZ];

These commands edit a limit test segment. The limit table editing is begun with EDITLIML;, and a segment is brought up for editing with SEDI<num>; or added using SADD;. The segment is closed with SDON;, the table is closed with EDITDONE;.

Command	Description	Range	Query Response
LIMD	Sets the limit delta value while editing a limit line segment.	amplitude range[a]	see “Note” below
LIML	Sets the lower limit value.	amplitude range[a]	see “Note” below
LIMM	Sets the middle limit value.	amplitude range[a]	see “Note” below
LIMS	Sets the limit stimulus break point.	stimulus range[b]	see “Note” below
LIMU	Sets the upper limit value.	amplitude range[a]	see “Note” below

a. For log mag: ± 500 dB. For phase: ± 500 degrees. For Smith chart and Polar: ± 500 units. For linear magnitude: ± 500 units. For SWR: ± 500 units. The scale is always positive, and has minimum values of 0.001 dB, 10e-12 degrees, 10e-15 seconds, and 10 picounits.

b. For frequency or power sweeps, refer to “Preset State and Memory Allocation,” in your analyzer’s reference guide. For CW time: 0 to 24 hours. For frequency sweep, transform on: ± 1/frequency step. For CW time sweep, transform on: ±1/time step.

NOTE Currently these commands can be queried by sending the command followed by the OUTPACTI command, as in the following example to query the upper limit

value:

```
10 OUTPUT 716;"LIMU;OUTPACTI;"
```

Future revisions of firmware may support the standard query form (which currently always returns a zero) for these commands.

LIMI

`LIMI<AMPO|STIO><num>[HZ|DB];` *or* `LIMI<AMPO|STIO>?;`
`LIMI<LINE|TEST><ON|OFF>;` *or* `LIMI<LINE|TEST>?;`
`LIMIMAOF;`

These commands are used to define and display limit testing.

Command	Description	Range	Query Response
LIMIAMPO	Enters the limit line amplitude offset.	amplitude range[a]	<num><$^{L}_{F}$>
LIMILINE	Turns the display of the limit lines on and off.	N/A	<0|1><$^{L}_{F}$>
LIMIMAOF	Marker to limit offset. Centers the limit lines about the current marker position using the limit amplitude offset function.	N/A	N/A
LIMISTIO	Enters the stimulus offset of the limit lines.	stimulus range[b]	<num><$^{L}_{F}$>
LIMITEST	Turns limit testing on and off.	N/A	<0|1><$^{L}_{F}$>

a. For log mag: ± 500 dB. For phase: ± 500 degrees. For Smith chart and Polar: ± 500 units. For linear magnitude: ± 500 units. For SWR: ± 500 units. The scale is always positive, and has minimum values of 0.001dB, 10e-12 degrees, 10e-15 seconds, and 10 picounits.

b. For frequency or power sweeps, refer to "Preset State and Memory Allocation," in your analyzer's reference guide. For CW time: 0 to 24 hours. For frequency sweep, transform on: ± 1/frequency step. For CW time sweep, transform on: ±1/time step.

LIMT

`LIMT<FL|SL|SP>;` *or* `LIMT<FL|SL|SP>?;`

These commands edit a limit test segment. The limit table editing is begun with `EDITLIML;`, and a segment is brought up for editing with `SEDI N;` or added using `SADD;`. The segment is closed with `SDON;`, the table is closed with `EDITDONE;`.

Command	Description	Range	Query Response
LIMTFL	Makes the segment a flat line.	N/A	<0|1><$^{L}_{F}$>
LIMTSL	Makes the segment a sloping line.	N/A	<0|1><$^{L}_{F}$>
LIMTSP	Makes the segment a single point.	N/A	<0|1><$^{L}_{F}$>

LINFREQ

`LINFREQ;` *or* `LINFREQ?;`

Command	Description	Range	Query Response
LINFREQ	Selects a linear frequency sweep.	N/A	<0|1><L_F>

LINM

`LINM;` *or* `LINM?;`

Command	Description	Range	Query Response
LINM	Selects the linear magnitude display format.	N/A	<0|1><L_F>

LINT

`LINT<DATA|MEMO><num>;`

Command	Description	Range	Query Response
LINTDATA	Enters the line type for plotting data.	integers 0–10	N/A
LINTMEMO	Enters the line type for plotting memory.	integers 0–10	N/A

LIS

`LISFREQ;` *or* `LISFREQ?;`

`LIS<IFBWM|PWRM><ON|OFF>;` *or* `LIS<IFBWM|PWRM>?;`

List frequency functions.

Command	Description	Range	Query Response
LISFREQ	Selects the list frequency sweep mode.	N/A	<0|1><L_F>
LISIFBWM	Enables/disables the IFBW setting for a list-frequency table in swept list mode.	N/A	<0|1><L_F>
LISPWRM	Enables/disables the power setting for a list-frequency table in swept list mode.	N/A	<0|1><L_F>

LISTTYPE

LISTTYPE<LSTP|LSWP>; *or* LISTTYPE?;

Command	Description	Range	Query Response[a]
LISTTYPELSTP	Selects the stepped list mode for use with a list-frequency table.	N/A	<0|1><L_F>
LISTTYPELSWP	Selects the swept list mode for use with a list-frequency table.	N/A	<0|1><L_F>

a. 0 = stepped list mode
1 = swept list mode

LISV

LISV;

Command	Description	Range	Query Response
LISV	Activates the list values function. Requesting a plot (or print) copies only the current page. See also "NEXP," "PREP," "PLOT," "PRINALL," and "PRINTALL."	N/A	N/A

LOAD

LOAD<num>;

Command	Description	Range	Query Response
LOAD	Loads the file from disk using the file name provided by the preceding TITF<num>; command. The actual file loaded depends on the file title in the file position specified by the TITF<num> command. Requires pass control mode when using the GPIB port.	integers 1–5	N/A

LOGFREQ

LOGFREQ; *or* LOGFREQ?;

Command	Description	Range	Query Response
LOGFREQ	Selects a log frequency sweep.	N/A	<0|1><L_F>

LOGM

LOGM; *or* LOGM?;

Command	Description	Range	Query Response
LOGM	Selects the log magnitude display format.	N/A	<0|1><L_F>

MANTRIG

MANTRIG; *or* MANTRIG?;

Command	Description	Range	Query Response
MANTRIG	Sets the trigger mode to manual trigger on point. OPC-compatible.	N/A	<0|1><L_F>

MAN_LASER

MAN_LASER<ON|OFF>; *or* MAN_LASER?;

Toggles laser on and off (with autobias).

MARK

MARK<1|2|3|4|5><num>; *or* MARK<1|2|3|4|5>?;

MARK<3DB|BUCK|FAUV|FSTI|FVAL><num>; *or* MARK<BUCK|FAUV|FSTI|FVAL>?;

MARK<CONT|COUP|DISC|MAXI|MINI|OFF|UNCO>; *or*
MARK<CONT|COUP|DISC|MAXI|MINI|OFF|UNCO>?;

MARK<CENT|CW|DELA|MIDD|REF|SPAN|STAR|STIM|STOP|ZERO>;

Command	Description	Range	Query Response
MARK1	Makes marker 1 active and sets its stimulus value.	stimulus range[a]	<num><L_F>
MARK2	Makes marker 2 active and sets its stimulus value.	stimulus range[a]	<num><L_F>
MARK3	Makes marker 3 active and sets its stimulus value.	stimulus range[a]	<num><L_F>
MARK4	Makes marker 4 active and sets its stimulus value.	stimulus range[a]	<num><L_F>
MARK5	Makes marker 5 active and sets its stimulus value.	stimulus range[a]	<num><L_F>

Command	Description	Range	Query Response
MARK3DB	Searches for 3dB bandwidth on the bandpass high-side, using marker 1 as the reference.	stimulus range	<num><L_F>
MARKBUCK	Places the active marker on a specific sweep point (bucket). <num> is the bucket number.	0 to (number-of-points – 1) See footnote[b].	<num><L_F>
MARKCENT	Sets the center stimulus value to that of the active marker's stimulus value.	N/A	N/A
MARKCONT	Places the markers continuously on the trace, not on discrete points (interpolates the marker values between discrete points).	N/A	<0|1><L_F>
MARKCOUP	Couples the markers between the channels, as opposed to MARKUNCO.	N/A	<0|1><L_F>
MARKCW	Sets the CW frequency to the active marker's frequency.	N/A	N/A
MARKDELA	Sets electrical length so group delay is zero at the active marker's stimulus.	N/A	N/A
MARKDISC	Places the markers on the discrete measurement points.	N/A	<0|1><L_F>
MARKFAUV	Sets the auxiliary value of the fixed marker position. Works in coordination with MARKFVAL and MARKFSTI.	amplitude range[c]	<num><L_F>
MARKFSTI	Sets the stimulus position of the fixed marker.	stimulus range[a]	<num><L_F>
MARKFVAL	Sets the value of the fixed marker position.	amplitude range[c]	<num><L_F>
MARKMAXI	Same as SEAMAX (search for maximum on current channel's trace).	N/A	<0|1><L_F>
MARKMIDD	Makes the marker amplitude the limit segment middle value during a limit segment edit.	N/A	N/A
MARKMINI	Same as SEAMIN (search for minimum on current channel's trace).	N/A	<0|1><L_F>
MARKOFF	Turns all markers and marker functions off.	N/A	<0|1><L_F>
MARKREF	Sets the reference value to that of the active marker's amplitude.	N/A	N/A
MARKSPAN	Sets the span for the entire trace to that of the span between the active marker and the delta reference marker.	N/A	N/A
MARKSTAR	Sets the start stimulus to that of the active marker's.	N/A	N/A

Command	Description	Range	Query Response
MARKSTIM	During a limit segment edit, sets the limit stimulus break point to that of the active marker's.	N/A	N/A
MARKSTOP	Sets the stop stimulus to that of the active marker's.	N/A	N/A
MARKUNCO	Uncouples the markers between channels, as opposed to MARKCOUP.	N/A	<0|1><$^{L}_{F}$>
MARKZERO	Places the fixed marker at the active marker position and makes it the delta reference.	N/A	N/A

a. For frequency or power sweeps, refer to “Preset State and Memory Allocation.” For CW time: 0 to 24 hours. For frequency sweep, transform on: ±1/frequency step. For CW time sweep, transform on: ±1/time step.
b. For example, on a 201 point sweep, <num> can range from 0 to 200.
c. For log mag: ± 500 dB. For phase: ± 500 degrees. For Smith chart and Polar: ± 500 units. For linear magnitude: ± 500 units. For SWR: ± 500 units. The scale is always positive, and has minimum values of 0.001dB, 10e–12 degrees, 10e–15 seconds, and 10 picounits.

MATI

MATI;

MATI to current memory.

MAXF

MAXF[<num>[freq suffix]];

Command	Description	Range	Query Response
MAXF	Sets the maximum valid frequency of a standard being defined during a cal kit modification.	0–1000 GHz	N/A

MEAS

MEAS<A|B|R|E01|O1|OE1|OE2|OFF|MEASO01>; *or*
MEAS<A|B|RE01|O1|OE1|OE2|OFF|MEASO01>?;

Command	Description	Range	Query Response
MEASA	Measures and displays input A on the active channel.	N/A	<0|1><$^{L}_{F}$>
MEASB	Measures and displays input B on the active channel.	N/A	<0|1><$^{L}_{F}$>
MEASR	Measures and displays input R on the active channel.	N/A	<0|1><$^{L}_{F}$>

Command	Description	Range	Query Response
MEASEO1	Measures and displays electrical-to-optical transmission.	N/A	<0|1><L_F>
MEASO1	Measures and displays optical reflection.	N/A	<0|1><L_F>
MEASOE1	Measures and displays optical-to-electrical transmission in port 1	N/A	<0|1><L_F>
MEASOE2	Measures and displays optical-to-electrical transmission in port 2.	N/A	<0|1><L_F>
MEASOFF	Switches off marker function measurements.	N/A	<0|1><L_F>
MEASOO1	Measures and displays optical transmission.	N/A	<0|1><L_F>

MEASTAT

`MEASTAT<ON|OFF>;` *or* `MEASTAT?;`

Command	Description	Range	Query Response
MEASTAT	Turns trace statistics on and off.	N/A	<0|1><L_F>

MEM

MEM<O1|O2|1I|2I>;

Command	Description	Range	Query Response
MEMO1	Activate memory 1.		
MEMO2	Activate memory 2.		
MEM1I	Memory 1 to memory 2.		
MEM2I	Memory 2 to memory 1.		

MENU

MENU<ON|OFF|AVG|CAL|COPY|DISP|FORM|MARK|MEAS|MRKF|POWE|RECA|SAVE|SCAL|SRCH|STIM|SWEE|SYST>;

Command	Description	Range	Query Response
MENUON	Turns the softkey menu on.	N/A	N/A
MENUOFF	Blanks the softkey menu. Use with caution, as this may give unusual results when setting up an instrument state. Recommend setting up states using MENUON (default) and, when setup is complete, using MENUOFF.	N/A	N/A
MENUAVG	Brings up the menu associated with the **Avg** front panel key.	N/A	N/A
MENUCAL	Brings up the menu associated with the **Cal** front panel key.	N/A	N/A
MENUCOPY	Brings up the menu associated with the **Copy** front panel key.	N/A	N/A
MENUDISP	Brings up the menu associated with the **Display** front panel key.	N/A	N/A
MENUFORM	Brings up the menu associated with the **Format** front panel key.	N/A	N/A
MENUMARK	Brings up the menu associated with the **Marker** front panel key.	N/A	N/A
MENUMEAS	Brings up the menu associated with the **Meas** front panel key.	N/A	N/A
MENUMRKF	Brings up the menu associated with the **Marker Fctn** front panel key.	N/A	N/A
MENUSRCH	Brings up the menu associated with the **Marker Fctn** front panel key.	N/A	N/A

Command	Description	Range	Query Response
MENUPOWE	Brings up the menu associated with the **Marker Fctn** front panel key.	N/A	N/A
MENURECA	Brings up the menu associated with the **Save/Recall** front panel key	N/A	N/A
MENUSAVE	Brings up the menu associated with the **Save/Recall** front panel key	N/A	N/A
MENUSCAL	Brings up the menu associated with the **Scale Ref** front panel key.	N/A	N/A
MENUSTIM	Brings up the menu associated with the **Sweep Setup** front panel key.	N/A	N/A
MENUSWEE	Brings up the menu associated with the **Sweep Setup** front panel key.	N/A	N/A
MENUSYST	Brings up the menu associated with the **System** front panel key.	N/A	N/A

Front Panel Equivalents

Press the associated hardkey listed above.

MINMAX

`MINMAX<ON|OFF>;` *or* `MINMAX?;`

Command	Description	Range	Query Response
MINMAX	Enables/disables min/max recording per segment. Min and max values are recorded per limit segment. Limit testing need not be active.	N/A	<0|1><L_F>

For more information refer to "Limit Line and Data Point Special Functions" on page 8-125.

MINU

`MINU;` *or* `MINU?;`

Command	Description	Range	Query Response
MINU	Data minus memory (linear subtraction). See also "DISPDMM."	N/A	<0|1><L_F>

MODEI;

Model to memory.

NEXP

NEXP;

Displays the next page of the operating parameters list. (Use OPEP to display the operating parameters list.) No query response.

NUMG

NUMG<num>;

Command	Description	Range	Query Response
NUMG	Activates the indicated number of groups of sweeps. A group is whatever is needed to update the current parameter once. This function restarts averaging if it is enabled. OPC-compatible.	integers 1–999	N/A

OMII

OMII;

Omits the isolation step of a calibration sequence. No query response.

OPC

OPC; *or* OPC?;

Command	Description	Range	Query Response[a]
OPC	Operation complete. Reports the completion of the next command received by setting bit 0 in the event-status register, or by replying to an interrogation if OPC? is issued.	N/A	<0|1><L_F>

a. 0 = next command not yet completed
1 = next command completed

OPEP

OPEP;

Presents a list of key operating parameters. Requesting a plot (or print) copies only the current page. See also "NEXP," "PREP," "PLOT," "PRINALL," and "PRINTALL." No query response.

OUTP

In the following "Syntax" section, commands are grouped alphabetically by common syntax form, not necessarily due to common functionality.

OUTP<ACTI|AMAX|AMIN|APER>;

OUTP<CALC|ICAL><num>;

OUTP<CALK|CHAN|CNTR|DATA|DATP|DATR|ERRO|FAIP|FORF|FORM>;

OUTPFARPLPT;

OUTP<IDEN|KEY|LEAS>;

OUTP<IPMCL><num>;

OUTP<MARK|MEMO|MSTA|MWID|OPTS|PLOT>;

OUTPLIM<num>;

OUTPLIM<F|L|M>;

OUTP<RAW><num>;

OUTP<PRIN>;

OUTP<RIPL|SEGAF|SEGAM|SEGF|SEGM>;

OUTPRPLBNDALL;

OUTPRPLBNDPF;

OUTPRPLBNDVAL;

OUTPSEQ<num>;

OUTP<SERN|STAT|TESS|TITL>;

NOTE Most commands that output an array require that you set the format for data transfers with the FORM command.

Many of these commands have an associated INPUT command that is used to transfer data to the analyzer. Refer to "Alphabetical Command Reference" on page 2-2 for a list of input commands.

Command	Description	Range	Response
OUTPACTI	Outputs the value of the active function, or the last active function if the active entry area is off. The value is returned in ASCII format.	N/A	<$><$L_F$>
OUTPAMAX[a]	Outputs the max values for all limit line segments. This is an ASCII transfer (FORM4).	N/A	<array><L_F>
OUTPAMIN[a]	Outputs the min values for all limit line segments. This is an ASCII transfer (FORM4).	N/A	<array><L_F>
OUTPAPER	Outputs the smoothing aperture in stimulus units, rather than as a percentage.	N/A	<num><L_F>
OUTPCALC	Outputs the selected error coefficient array for the active cal on the active channel.[b]	two-digit integers 01–12	<array><L_F>
OUTPCALK	Outputs the currently active calibration kit, as a string of less than 1000 bytes. The data is in FORM1.	N/A	<$><$L_F$>
OUTPFARPLPT	Outputs the onscreen failed ripple point information in the following comma-separated value format: the number of failed points followed by pairs of numbers representing the first failed frequency, first failure value, second failed frequency, second failure value, and so on.	N/A	<num,array><L_F>
OUTPCHAN	Outputs the active channel number: 1, 2, 3, or 4.	N/A	<num><L_F>
OUTCNTR	Outputs the ABUS counter.	N/A	
OUTPDATA	Outputs the error-corrected data from the active channel in real/imaginary pairs. See Figure 6-1 on page 6-3.	N/A	<array><L_F>
OUTPDATP	Outputs the trace data indexed by point (see “SELPT”).	N/A	<num,num><L_F>
OUTPDATR	Outputs the trace data for a range of points (see “SELMINPT,” “SELMAXPT”). This is an ASCII (FORM4) transfer.	N/A	<array><L_F>
OUTPERRO	Outputs the oldest error message in the error queue. Sends the error number first, and then the error message itself, as an ASCII (FORM4) string no longer than 50 characters.	N/A	<num,$><$L_F$>

Command	Description	Range	Response
OUTPFAIP	This command is similar to `OUTPLIMF` except that it reports the number of failures first, followed by the stimulus and trace values for each failed point in the test. ASCII format.[a]	N/A	<array><L_F>
OUTPFORM	Outputs the formatted display data array from the active channel, in current display units. See Table 2-3 on page 2-76.	N/A	<array><L_F>
OUTPIDEN	Outputs the identification string for the analyzer in the form: `Agilent,8703B,xxxxxxxxxx,X.XX` where 8703B is the model number of the instrument, xxxxxxxxxx is the serial number of the instrument, and X.XX is the firmware revision of the instrument. (Same as the "IDN?" command.)	N/A	<$><$L_F$>
OUTPLEAS	Outputs the learn string, which contains the entire front panel state, the limit table, and the list frequency table. It is always in binary format not intended for decoding.	N/A	<learnstring><L_F>
OUTPLIM	Outputs the status of the limit test for the channel selected with <num>.[a,c]	integers 1–4	<0|1|−1><L_F>
OUTPLIMF	Outputs the limit test results for each failed point, followed by the number of failed points. This is an ASCII transfer.[a,d]	N/A	<array><L_F>
OUTPLIML	Outputs the limit test results for each point in the sweep. This is an ASCII transfer.[a,c,d]	N/A	<array><L_F>
OUTPLIMM	Outputs the limit test results at the active marker.[a,c,d]	N/A	<num,num,num,num><L_F>
OUTPMARK	Outputs the active marker values. The first two numbers are the marker response values, and the last is the stimulus value. See Table 2-3 on page 2-76 for the meaning of the response values as a function of display format.	N/A	<num,num,num><L_F>
OUTPMAXP	Outputs the maximum point value between selected points.	N/A	<num,num><L_F>
OUTPMEMO	Outputs the memory trace from the active channel. The data is in real/imaginary pairs, and can be treated the same as data read with the OUTPDATA command. See Figure 6-1 on page 6-3.	N/A	<array><L_F>

Command	Description	Range	Response
OUTPMINP	Outputs the minimum point value between selected points.	N/A	<num,num><L_F>
OUTPMRIS	Outputs three values for risetime.	N/A	<num,num,num><L_F>
OUTPMSTA	Outputs the marker statistics in ASCII format: mean, standard deviation, and peak-to-peak variation in that order. If statistics is not on, it is turned on to generate current values and turned off again. See also "MEASTAT."	N/A	<num,num,num><L_F>
OUTPMWID	Outputs the marker bandwidths search results in ASCII format: bandwidth, center, and Q in that order. If widths is not on, it is turned on to generate current values and then turned off again.	N/A	<num,num,num><L_F>
OUTPOPTS	Outputs an ASCII string of the options installed in the analyzer.	N/A	<$><$L_F$>
OUTPPLOT	Outputs the HPGL plot string in ASCII format to the GPIB port. Can be directed to a plotter, or read into the computer.	N/A	<$><$L_F$>
OUTPPRIN	Outputs a PCL raster dump of the display, intended for a graphics printer.	N/A	<$><$L_F$>
OUTPRAW	Outputs the selected raw data array. See Figure 6-1 on page 6-3.	integers 1–4: 1=S_{11} data 2=S_{21} data 3=S_{12} data 4=S_{22} data	<array><L_F>
OUTPRIPL	Outputs the peak-to-peak ripple between selected points.	Frequency range	<num,num><L_F>
OUTPRPLBNDALL	Outputs the measured ripple values for all active frequency bands in the following comma-separated value format: the number of bands followed by pairs of numbers representing the first band number (1), ripple value of first band, second band number (2), ripple value of second band, and so on.	N/A	<num,array><L_F>
OUTPRPLBNDPF	Outputs the pass/fail status for selected frequency band (see "SELBND") as "1" (band passes) or as "0" (band fails).	N/A	<0|1><L_F>
OUTPRPLBNDVAL	Outputs the ripple value for selected frequency band (see "SELBND").	N/A	<num><L_F>

Command	Description	Range	Response
OUTPSEGAF	Outputs the segment number and its limit test status for all active segments. This is an ASCII transfer.[a,c]	N/A	<array><L_F>
OUTPSEGAM	Outputs the limit test min/max for all segments. Outputs the segment number, max stimulus, max value, min stimulus, min value for all active segments. This is an ASCII transfer.[a,c]	N/A	<array><L_F>
OUTPSEGF	Outputs the limit test status for a specified segment. See also "SELSEG."[a,c]	N/A	<0|1|−1><L_F>
OUTPSEGM	Outputs limit test min/max for a specified segment. See also "SELSEG."[a]	N/A	<num,num><L_F>
OUTPSERN	Outputs a string that contains the serial number of the analyzer.	N/A	<$><$L_F$>
OUTPSTAT	Returns the status byte as an ASCII integer (0–255) that can be interpreted as the 8-bit status byte. Refer to "The Status Byte" on page 7-6 for more information about the status byte. This command is the same as "STB?."	N/A	<num><L_F>
OUTPTESS	Outputs the test status	N/A	
OUTPTITL	Outputs the display title in ASCII format.	N/A	<$><$L_F$>

a. Refer to "Limit Line and Data Point Special Functions" on page 8-125.
b. See Table 2-1 on page 2-37 for the contents of the different arrays. Each array is output in the currently set form determined by the FORM command. The data is in real/imaginary pairs, with the same number of pairs as points in the sweep.
c. Values returned for limit test status are: 0 (fail), 1 (pass), or −1 (no limit).
d. This command outputs the limit test results. The results consist of four fields. First is the stimulus value for the point. Second is an integer indicating test status. Third is the upper limit at that point. Fourth is the lower limit at that point. If there are no limits at that point, the third and forth fields are zero.

OUTP Reference Tables

Table 2-2. Error Coefficient Arrays

Array	Response	Response & Isolation	1-port	Enhanced Response	2-port [a]	TRL/ LRM	O/E Response & Match	E/O Response & Match
01	E_R or E_T	E_X (E_D)[b]	E_D	E_D	E_{DF}	E_{DF}	E_{DR}	E_{DI}
02		E_T (E_R)	E_S	E_S	E_{SF}	E_{SF}	E_{SR}	E_S
03			E_R	E_R	E_{RF}	E_{RF}	E_{RR}	E_R
04				E_X	E_{XF}	E_{XF}	E_{XF}	E_X
05				E_L[c]	E_{LF}	E_{LF}	E_{LF}	E_T
06				E_T	E_{TF}	E_{TF}	E_{TF}	
07					E_{DR}	E_{DR}	E_{DF}	
08					E_{SR}	E_{SR}	E_{SF}	
09					E_{RR}	E_{RR}	E_{RF}	
10					E_{XR}	E_{XR}		
11					E_{LR}	E_{LR}		
12					E_{TR}	E_{TR}		

a. One path, 2-port cal duplicates arrays 1 to 6 in arrays 7 to 12.
b. Response and isolation corrects for crosstalk and transmission tracking in transmission measurements, and for directivity and reflection tracking in reflection measurements.
c. This term is used to generate the calibration coefficients, but is not used during measurement error correction.

Meaning of first subscript:
D: directivity
S: source match
R: reflection tracking
X: crosstalk or isolation
L: load match
T: transmission tracking

Meaning of second subscript:
F: forward
R: reverse

PAUS

PAUS;

Pauses the sequence; to be followed by CONS to resume the sequence. No query response.

PHAO

PHAO<num>; *or* PHAO?;

Command	Description	Range	Query Response
PHAO	Sets the phase offset.	0–360 degrees	<num><L_F>

PHAS

PHAS; *or* PHAS?;

Command	Description	Range	Query Response
PHAS	Selects the phase display format.	N/A	<0|1><L_F>

PLOT

PLOT;

Initiates a plot. Requires pass control mode when using the GPIB port. No query response.

POIN

POIN<num>; *or* POIN?;

Command	Description	Range	Query Response
POIN	Sets the number of points in the sweep, or in a sweep segment.	Choose from: 3, 11, 21, 26, 51, 101, 201, 401, 801, 1601	<num><L_F>

NOTE This command should be followed by a wait equal to 2 sweeps. Example wait code written in BASIC:

```
OUTPUT 716;"POIN801;"
OUTPUT 716;”SWET?;”
ENTER 716;T
WAIT 2*T
```

POL

POL<A|MLIN|MLOG|MRI>; *or* POL<A|MLIN|MLOG|MRI>?;

Command	Description	Range	Query Response
POLA	Selects the polar display format.	N/A	<0|1><L_F>
POLMLIN	Selects linear as the marker readout format for polar display.	N/A	<0|1><L_F>
POLMLOG	Selects log as the marker readout format for polar display.	N/A	<0|1><L_F>
POLMRI	Selects real/imaginary as the marker readout format for polar display.	N/A	<0|1><L_F>

PORE

PORE<ON|OFF>; *or* PORE?;

Command	Description	Range	Query Response
PORE	Turns port extensions on and off.	N/A	<0|1><L_F>

PORT

PORT<1|2><num>[S]; *or* PORT<1|2>?;

These commands set the port extension length for the indicated port or input. Ports 1 and 2 refer to the test set ports.

Command	Description	Range	Query Response
PORT1	Port 1	±10 seconds	<num><L_F>
PORT2	Port 2	±10 seconds	<num><L_F>

PORTP

PORTP<CPLD|UNCPLD>; *or* PORTP?;

Command	Description	Range	Query Response[a]
PORTP	Selects either coupled or uncoupled for the port powers of a given channel.	N/A	<0|1><L_F>

a. 0 = uncoupled
1 = coupled

POWE

POWE<num>[DB]; *or* POWE?;

Command	Description	Range	Query Response
POWE	Sets the output power level.	output power range of your analyzer[a]	<num><L_F>

a. The output power range of your analyzer depends upon the model and installed options. Refer to your analyzer's reference guide to determine the power range of your analyzer.

POWS

POWS; *or* POWS?;

Command	Description	Range	Query Response
POWS	Selects power sweep, from the sweep type menu.	N/A	<0|1><L_F>

PRAN

Syntax

PRAN<num>;

Command	Description	Range	Query Response
PRAN	Sets the source power range.	integers[a] 0–7 integers[b] 01–12	N/A

a. PRAN0 through PRAN7 are used for ranges 0 through 7.
b. Use two-digit integers 01 through 12. PRAN01 through PRAN12 are used for ranges 0 through 11.

PRES

PRES;

Presets the analyzer to the factory preset state. OPC-compatible. No query response.

NOTE Pressing the **Preset** key on the analyzer will either invoke the factory preset state, or a user-selected state (if one has been set up). Sending the PRES command will *always* invoke the factory preset state. This is true even if the analyzer is currently set up to recall a user preset state when the **Preset** key is pressed. Refer to "Preset State and Memory Allocation" on page 9-1. For more information on user presets, see your analyzer's user's guide.

NOTE This command should use OPC? to prevent timing errors with subsequent commands. Example code written in BASIC:

```
10 OUTPUT 716;"OPC?;PRES;"
20 ENTER 716;X
```

PWRR

PWRR<PMAN|PAUTO>; *or* PWRR?;

Command	Description	Range	Query Response
PWRR	Selects whether the power range is in auto or manual mode.	N/A	<0|1><L_F>[a]

a. 0 = manual mode
1 = auto mode

PULV [value];

Set pulse width search value.

PULW<ON|OFF>;

Select pulse width search OFF/ON.

RAI

RAI<D|ISOL|RESP>;

Command	Description	Range	Query Response
RAID	Completes the response and isolation cal sequence. OPC-compatible.	N/A	N/A
RAIISOL	Calls the isolation class for the response and isolation calibration.	N/A	N/A
RAIRESP	Calls the response class for the response and isolation calibration.	N/A	N/A

RAMD;

Response and match cal done.

READ

READ<DATE|TIME>;

Command	Description	Range	Query Response
READDATE	Outputs the date in the following string format: DD MMM YYYY.	N/A	N/A
READTIME	Outputs the time in the following string format: HH:MM:SS.	N/A	N/A

REAL

REAL; *or* REAL?;

Command	Description	Range	Query Response
REAL	Sets the display format to real.	N/A	<0|1><L_F>

RECAREG

RECAREG<num>;

Command	Description	Range	Query Response
RECAREG	Recalls from save/recall registers 01–31. OPC-compatible.	two-digit integers 01–31	N/A

RECAREG<1-31>;

Recalls previously saved display colors. No query response.

RECEOUT<ON|OFF>;

Select path to receiver output; ON=CAL, OFF = OPT.

RECO

RECO;

Recalls previously saved display colors. No query response.

REF

REF<D|L>;

Command	Description	Range	Query Response
REFD	Completes the reflection calibration subsequence of a 2-port calibration. OPC-compatible.	N/A	N/A
REFL	Begins the reflection calibration subsequence of a 2-port calibration.	N/A	N/A

REF

REF<P|V><num>; *or* REF<P|V>?;

Command	Description	Range	Query Response
REFP	Enters the reference position. 0 is the bottom, 10 is the top of the graticule.	integers 0–10	<num><L_F>
REFV	Enters the reference line value.	amplitude range[a]	<num><L_F>

a. For log mag: ± 500 dB. For phase: ± 500 degrees. For Smith chart and Polar: ± 500 units. For linear magnitude: ± 500 units. For SWR: ± 500 units. The scale is always positive, and has minimum values of 0.001dB, 10e–12 degrees, 10e–15 seconds, and 10 picounits.

REFT

REFT;

Recalls file titles from disk. Requires pass control if using an external disk drive on GPIB. No query response.

RESC

RESC;

Command	Description	Range	Query Response
RESC	Resume a previously started cal sequence.	N/A	N/A

RESD

RESD;

Command	Description	Range	Query Response
RESD	Restores the measurement display after viewing the operating parameters or list values.	N/A	N/A

RESM;

Reset model 1.

RESPDONE

RESPDONE;

Command	Description	Range	Query Response
RESPDONE	Completes the response calibration sequence. OPC-compatible.	N/A	N/A

REST

REST;

Measurement restart. No query response.

REV

REV<I|M|T>;

These commands are OPC-compatible if there is only one standard in the class. If there is just one standard, that standard is measured automatically. If there is more than one standard in the class, the class command only calls another menu.

Command	Description	Range	Query Response
REVI	Calls the reverse isolation calibration class during a full 2-port calibration.	N/A	N/A
REVM	Calls the reverse match calibration class during a full 2-port calibration.	N/A	N/A
REVT	Calls the reverse transmission calibration class during a full 2-port calibration.	N/A	N/A

RLIMLINE

RLIMLINE<ON|OFF>; *or* RLIMLINE?;

Command	Description	Range	Query Response
RLIMLINE	Turns the lines that represent the ripple test limits on and off.	N/A	<0|1><L_F>

RLIMM

RLIMM<num>[DB]; *or* RLIMM?;

Command	Description	Range	Query Response
RLIMM	Sets the value of the maximum allowable ripple limit for current frequency band.	0.01 to 100 dB	<num><L_F>

RLIMSTP

RLIMSTP<num>[HZ|KHZ|MHZ|GHZ]; *or* RLIMSTP?;

Command	Description	Range	Query Response
RLIMSTP	Sets the stop frequency of the current frequency band.	stimulus range[a]	<num><L_F>

a. Refer to “Preset State and Memory Allocation.”

RLIMSTR

RLIMSTR<num>[HZ|KHZ|MHZ|GHZ]; *or* RLIMSTR?;

Command	Description	Range	Query Response
RLIMSTR	Sets the start frequency of the current ripple limit.	stimulus range[a]	<num><L_F>

a. Refer to “Preset State and Memory Allocation.”

RLIMTEST

RLIMTEST<ON|OFF>; *or* RLIMTEST?;

Command	Description	Range	Query Response
RLIMTEST	Turns the ripple limit test on and off.	N/A	<0|1><L_F>

RLIMVAL

RLIMVAL<OFF|ABS|MAR>;

Command	Description	Range	Query Response
RLIMVAL	Displays the ripple limit value of the selected band (see "SELBND") in absolute format (ABS) or margin format (MAR). OFF turns the displayed ripple limit value off.	N/A	N/A

RST

RST;

Presets the analyzer to the factory preset state. OPC-compatible. See Chapter 9, "Preset State and Memory Allocation" No query response.

NOTE Pressing the **Preset** key on the analyzer will either invoke the factory preset state, or a user-selected state (if one has been set up). Sending the RST command will *always* invoke the factory preset state. This is true even if the analyzer is currently set up to recall a user preset state when the **Preset** key is pressed. For more information on user presets, see your analyzer's user's guide.

S

S<11|12|21|22>; *or* S<11|12|21|22>?;

Command	Description	Range	Query Response
S11	Forward reflection measurement	N/A	<0|1><L_F>
S12	Reverse transmission measurement	N/A	<0|1><L_F>
S21	Forward transmission measurement	N/A	<0|1><L_F>
S22	Reverse reflection measurement	N/A	<0|1><L_F>

SADD

SADD;

Adds a new segment to the table during a list-frequency, limit-table, cal sensor table, or power loss table edit. No query response.

SAV

SAV<1|2|C>;

Command	Description	Range	Query Response
SAV1	Completes the 1-port calibration sequence. OPC-compatible.	N/A	N/A
SAV2	Completes the 2-port calibration sequence. OPC-compatible.	N/A	N/A
SAVC	Completes the transfer of error correction coefficients back into the instrument. OPC-compatible.	N/A	N/A

SAVE

SAVEREG<num>;

Command	Description	Range	Query Response
SAVEREG	Saves to save/recall registers 01–31. SAVEREG01 through SAVEREG05 are the same as SAVE1 through SAVE5. OPC-compatible.	two-digit integers 01–31	N/A

SAVECSV

SAVECSV;

Saves the current measurement to the disk drive in the comma-separated value (CSV) format. No query response.

SAVEJPG

SAVEJPG;

Saves the current display to the disk drive in the JPG format. OPC-compatible. No query response.

SAVU

SAVU<ASCI|BINA>;

Command	Description	Range	Query Response
SAVUASCI	Selects ASCII format for saving to disk. Conforms to CITIFile specifications.	N/A	N/A
SAVUBINA	Selects binary format for saving to disk.	N/A	N/A

SCAL

SCAL<num>; *or* SCAL?;

Command	Description	Range	Query Response
SCAL	Sets the trace scale factor.	amplitude range[a]	<num><L_F>

a. For log mag: ± 500 dB. For phase: ± 500 degrees. For Smith chart and Polar: ± 500 units. For linear magnitude: ± 500 units. For SWR: ± 500 units. The scale is always positive, and has minimum values of 0.001dB, 10e–12 degrees, 10e–15 seconds, and 10 picounits.

SDEL

SDEL;

Deletes the current segment while editing a list frequency, a limit table, or a power loss list. No query response.

SDON

SDON;

Closes a segment after editing a list frequency, a limit table, or a power loss list. No query response.

SEA

SEA<L|R>;

SEA<MAX|MIN|OFF>; *or* SEA<MAX|MIN|OFF>?;

SEATARG<num>; *or* SEATARG?;

These commands control the marker searches. The marker searches place the active marker according to the indicated search criteria. The search is continuously updated if tracking is ON (see "TRACK").

Command	Description	Range	Query Response
SEAL	Search left for next occurrence of the target value.	N/A	N/A
SEAMAX	Search for trace maximum on the current channel.	N/A	<0|1><L_F>
SEAMIN	Search for trace minimum on the current channel.	N/A	<0|1><L_F>
SEAOFF	Turns the marker search off.	N/A	<0|1><L_F>
SEAR	Search right for next occurrence of the target value.	N/A	N/A
SEATARG	Set the search target amplitude.	amplitude range[a]	<num><L_F>

a. For log mag: ± 500 dB. For phase: ± 500 degrees. For Smith chart and Polar: ± 500 units. For linear magnitude: ± 500 units. For SWR: ± 500 units.

SEDI

`SEDI<num>; or SEDI?;`

Command	Description	Range	Query Response
SEDI	During either a frequency, limit, or power loss table edit, selects segment <num> for editing.	state dependent. Range for frequency segment = 1 to 30. Range for limit test segment = 1 to 18 . Range for power loss table segment = 1 to 12	<num><L_F>

SEG

`SEG<IFBW|POWER><num>;`

Command	Description	Range	Query Response
SEGIFBW	Sets the IFBW for the active segment of a list-frequency table in swept list mode.	Choose from 10, 30, 100, 300, 1000, 3000, 3700, 6000	see “Note” below
SEGPOWER	Sets the power for the active segment of a list-frequency table in swept list mode.	output power range of your analyzer[a]	see “Note below

a. The output power range is dependent upon the model and option configuration of your analyzer. Refer to your analyzer’s reference guide to determine the output power range of your analyzer.

NOTE Currently these commands can be queried by sending the command followed by the `OUTPACTI` command, as in the following example to query the upper limit value:

```
10 OUTPUT 716;”SEGIFBW;OUTPACTI;”
```

Future revisions of firmware may support the standard query form (which currently always returns a zero) for these commands.

SEL

SEL<MAXPT|MINPT|PT|SEG><num>; *or* SEL<MAXPT|MINPT|PT|SEG>?;

Command	Description	Range	Query Response
SELMAXPT	Selects the last point number in the range of points that the OUTPDATR command will report.	0 to n–1, where n=number of points	<num><$^{L}_{F}$>
SELMINPT	Selects the first point number in the range of points that the OUTPDATR command will report.	0 to n–1, where n=number of points	<num><$^{L}_{F}$>
SELPT	Selects the point number that the OUTPDATP command will report.	0 to n–1, where n=number of points	<num><$^{L}_{F}$>
SELSEG	Selects the segment number to report on for the OUTPSEGF and OUTPSEGM commands.	integers 1–18	<num><$^{L}_{F}$>

NOTE For the definition of a limit segment, see "Limit Line and Data Point Special Functions" on page 8-125.

SELBND

SELBND<num>; *or* SELBND?;

Command	Description	Range	Query Response
SELBND	Selects the ripple frequency band for the following commands: OUTPRPLBNDPF, OUTPRPLBNDVAL, and RLIMVAL.	integers 1–12	<num><$^{L}_{F}$>

SET

SET<Z><num>; *or* SET<BIT|Z>?;

SET<DATE|TIME><$>;

Command	Description	Range	Query Response
SETDATE	Sets the date in the following format: DD MMM YYYY, where DD is the day and must be 2 digits, MMM is the month and must be three alpha characters (JAN, FEB, MAR, APR, MAY, JUN, JUL, AUG, SEP, OCT, NOV, DEC), and YYYY is the year and must be 4 digits.	See "Description ."	N/A

Command	Description	Range	Query Response
SETTIME	Sets the time in the following format: HH:MM:SS, where HH is the hour, MM is minutes, SS is seconds, and each must be 2 digits.	See “Description .”	N/A
SETZ	Sets the characteristic impedance of the measurement system.	0.1 to 500 Ω	<num><L_F>

SING

SING;

Single sweep. OPC-compatible. No query response.

SLI

SLI<D>;

Sliding load done. No query response.

SMI

SMI<C|MGB|MLIN|MLOG|MRI|MRX>; *or* SMI<C|MGB|MLIN|MLOG|MRI|MRX>?;

Command	Description	Range	Query Response
SMIC	Selects Smith chart display format.	N/A	<0|1><L_F>
SMIMGB	Selects G+jB (conductance and susceptance) marker readout on a Smith chart.	N/A	<0|1><L_F>
SMIMLIN	Selects linear magnitude marker readout on a Smith chart.	N/A	<0|1><L_F>
SMIMLOG	Selects log magnitude marker readout on a Smith chart.	N/A	<0|1><L_F>
SMIMRI	Selects real/imaginary pairs (resistance and reactance) marker readout on a Smith chart.	N/A	<0|1><L_F>
SMIMRX	Selects R + jX marker readout on a Smith chart.	N/A	<0|1><L_F>

SMOO

SMOOAPER<num>; *or* SMOOAPER?;

SMOOO<ON|OFF>; *or* SMOOO?;

Command	Description	Range	Query Response
SMOOAPER	Sets the smoothing aperture as a percent of the trace.	0.05 to 20%	<num><L_F>
SMOOO	Selects whether smoothing is on or off.	N/A	<0|1><L_F>

SOFR

SOFR;

Displays the firmware revision on the screen. No query response.

SOFT

SOFT<num>;

Command	Description	Range	Query Response
SOFT	Acts as though the indicated softkey was pressed.	integers 1–8	N/A

SOUP

SOUP<ON|OFF>; *or* SOUP?;

Command	Description	Range	Query Response
SOUP	Selects whether the source power is on or off.	N/A	<0|1><L_F>

SPAN

SPAN<num>[HZ|DB]; *or* SPAN?;

Command	Description	Range	Query Response
SPAN	Sets the stimulus span value. If a list frequency segment is being edited, sets the span of the list segment.	stimulus range[a]	<num><L_F>

a. For frequency or power sweeps, refer to “Preset State and Memory Allocation,” in your analyzer’s reference guide. For CW time: 0 to 24 hours. For frequency sweep, transform on: ± 1/frequency step. For CW time sweep, transform on: ±1/time step.

SPLD

SPLD<ON|OFF>; *or* SPLD?;

Command	Description	Range	Query Response
SPLD	Turns the split display mode on and off.	N/A	<0|1><L_F>

SPLID

SPLID<1|2|4>; *or* SPLID<1|2|4>?;

Command	Description	Range	Query Response
SPLID1	Puts all displayed channels on one full-size graticule.	N/A	<0|1><L_F>
SPLID2	Puts all displayed channels on two graticules.	N/A	<0|1><L_F>
SPLID4	Puts each displayed channel on a separate graticule.	N/A	<0|1><L_F>

SRE

SRE<num>; *or* SRE?;

Command	Description	Range	Query Response
SRE	Service request enable. A bit set in <num> enables the corresponding bit in the status byte to generate an SRQ.	integers 0–255	<num><L_F>

SSEG

SSEG<num>; *or* SSEG?;

Command	Description	Range	Query Response
SSEG	Selects the desired segment of the frequency list for a list frequency sweep. See also "ASEG".	integers 1–30	<num><L_F>

STAN

STAN<A|B|C|D|E|F|G>;

Standards A through G are associated with softkeys 1 through 7, respectively.

Command	Description	Range	Query Response
STANA	These 7 commands (OPC-compatible) select a standard from a class during a calibration sequence. If a class is requested, as in CLASS11A (S11 1-port cal) the analyzer will do one of two things: • If there is only one standard in the class, it will measure that standard automatically. • If there are several standards in the class, then one of these commands must be used to select one of these standards, causing it to be measured.	N/A	N/A
STANB		N/A	N/A
STANC		N/A	N/A
STAND		N/A	N/A
STANE		N/A	N/A
STANF		N/A	N/A
STANG		N/A	N/A

STAR

STAR<num>[HZ|DB]; *or* STAR?;

Command	Description	Range	Query Response
STAR	Sets the start stimulus value. If a list frequency segment is being edited, sets the start of the list segment.	stimulus range[a]	<num><$^{L}_{F}$>

a. For frequency or power sweeps, refer to “Preset State and Memory Allocation,” in your analyzer’s reference guide. For CW time: 0 to 24 hours. For frequency sweep, transform on: ± 1/frequency step. For CW time sweep, transform on: ±1/time step.

STB?

STB?;

Command	Description	Range	Query Response
STB?	Query only. Outputs the status byte in ASCII format (FORM4). Same as OUTPSTAT.	N/A	<num><$^{L}_{F}$>

STEPSWP

Syntax

STEPSWP<ON|OFF>; *or* STEPSWP?;

Command	Description	Range	Query Response
STEPSWP	Turns step sweep mode on or off.	N/A	<0|1><$^{L}_{F}$>

STOP

STOP<num>[HZ|DB]; *or* STOP?;

Command	Description	Range	Query Response
STOP	Sets the stop stimulus value. If a list frequency segment is being edited, sets the stop of the list segment.	stimulus range[a]	<num><L_F>

a. For frequency or power sweeps, refer to "Preset State and Memory Allocation," in your analyzer's reference guide. For CW time: 0 to 24 hours. For frequency sweep, transform on: ± 1/frequency step. For CW time sweep, transform on: ±1/time step.

STPSIZE

STPSIZE<num>[HZ|DB]; *or* STPSIZE?;

Command	Description	Range	Query Response
STPSIZE	Sets the step size while editing a list frequency segment.	stimulus range[a]	<num><L_F>

a. For frequency or power sweeps, refer to "Preset State and Memory Allocation," in your analyzer's reference guide. For CW time: 0 to 24 hours. For frequency sweep, transform on: ±1/frequency step. For CW time sweep, transform on: ±1/time step.

SWE

SWEA;

SWET<num>[S]; *or* SWET?;

Command	Description	Range	Query Response
SWEA	Automatically selects the fastest sweep time based on the current analyzer settings for number of points, IF bandwidth, sweep mode, averaging condition and frequency span.	N/A	N/A
SWET	Sets the sweep time. (Setting SWET0 is equivalent to sending the SWEA command.)	0–86,400 s	<num><L_F>

NOTE The SWET command should be followed by a wait equal to 2 sweeps. Example wait code written in BASIC:

```
10 OUTPUT 716;"SWET.1;"
20 WAIT 2*.1
```

SWR

SWR; *or* SWR?;

Command	Description	Range	Query Response
SWR	Selects the SWR display format.	N/A	<0|1><L_F>

TESS?

TESS?;

Command	Description	Range	Query Response
TESS?	Query only. Queries whether a test set is connected. Returns a one on the standard analyzer.	N/A	<0|1><L_F>

TINT

TINT<num>; *or* TINT?;

Command	Description	Range	Query Response
TINT	Adjusts the tint for the selected display feature.	integers 0–100	<num><L_F>

TIT

TIT<F|REG><num><$>;

TIT<L><$>;

TITSQ;

Command	Description	Range	Query Response
TITF	Titles the indicated file numbers.	**<num>:** 1–5 **<$>:** 10 char. max.	N/A
TITL	Enters a new display title.	48 characters max	N/A
TITREG	Titles save/recall registers 01 through 31. TITREG01 through TITREG05 are the same as TITR1 through TITR5.	**<num>:** 01–31 **<$>:** 10 char. max.	N/A

TRA

TRA<D|N>;

Command	Description	Range	Query Response
TRAD	Completes the transmission calibration subsequence of a 2-port calibration or enhanced response calibration. OPC-compatible.	N/A	N/A
TRAN	Begins the transmission calibration subsequence of a 2-port calibration or enhanced response calibration.	N/A	N/A

TRACK

TRACK<ON|OFF>; *or* TRACK?;

Command	Description	Range	Query Response
TRACK	Turns marker search tracking on and off.	N/A	<0|1><L_F>

```
20 ENTER 716;X
```

TRIG

TRIG;

Wait for sweep trigger.

TST?

TST?;

Command	Description	Range	Query Response
TST?	Query only. Causes a self test and returns a zero if the test is passed.	N/A	<num><L_F>

TSTP

TSTP<P1|P2>;

Selects test port 1 or 2 for non-S-parameter measurements. No query response.

UP

UP;

Increments the value displayed in the active entry area (emulates pressing the up-arrow key). No query response.

USEPASC

USEPASC; *or* USEPASC?;

Command	Description	Range	Query Response
USEPASC	Puts the analyzer in pass control mode.	N/A	<0|1><L_F>

VELOFACT

VELOFACT<num>; or VELOFACT?;

Command	Description	Range	Query Response
VELOFACT	Enters the velocity factor of the transmission medium.	0 to 10	<num><L_F>

WID

WIDT<ON|OFF>; or WIDT?;
WIDV<num>; or WIDV?;

Command	Description	Range	Query Response
WIDT	Turns the bandwidth search on and off.	N/A	<0|1><L_F>
WIDV	Enters the widths search parameter.	amplitude range[a]	<num><L_F>

a. For log mag: ± 500 dB. For phase: ± 500 degrees. For Smith chart and Polar: ± 500 units. For linear magnitude: ± 500 units. For SWR: ± 500 units.

WINDUSEM

WINDUSEM;

Uses trace memory as window.

WRSK

WRSK<num><$>;

Command	Description	Range	Query Response
WRSK	Enters new softkey labels into the indicated softkey positions. Initial use of these commands requires previous commands MENUFORM; and MENUOFF;.	**<num>:** integers 1–8 **<$>:** 10 char. max.	N/A

8703A Commands Not Supported in the 8703B

Table 2-3.

ALIS	HARMSEC	MAXF
ASAMP	HARMTHIR	MEASEO2
CALCEXEC	INDEREFRxx	MEASO2
CALCEXIT	INPUOEO1	MEASOO2
CALCHECK	INPUOEO2	MODRF
CWEXT	INPUOEO3	OUTKEY
EXTAOPTI	INPUOEO4	OUTMUPL
FRES	INPUOEO5	WAVL1300
GUIS	INPUOEO6	WAVL1550
HARMOFF	KOR?	

3

Command Listings

Alphabetical List of Commands

AB
ADDRCONT
ADDRDISC
ADDRPLOT
ADDRPOWM
ADDRPRIN
ADJB
ALTAB
AR
ASEG
AUTB
AUTO
AUXC
AVERFACT
AVERO
AVERREST
BACI
BEEPDONE
BEEPFAIL
BEEPWARN
BR
BSAMP
BWLIMDB
BWLIMDISP
BWLIMMAX
BWLIMMKR
BWLIMIN
BWLIMSTAT
BWLIMTEST
BWLIMVAL
CALIEORM
CALIOERM
CALIFUL2
CALIRAI
CALIRESP
CALIS111
CALIS221
CALK35MM
CBRI
CENT
CHAN1
CHAN2
CHAN3
CHAN4
CLAD
CLASS11A
CLASS11B
CLASS11C
CLASS22A
CLASS22B
CLASS22C
CLEAL
CLEARALL
CLEAREG<01-31>
CLEL
CLER
CLES
CLS
COEFA<1-4>
COEFB<1-4>
COEFDELA
COEFK
CONS
CONT
CONV1DS
CONVOFF
CONVYREF
CONVYTRA
CONVZREF
CONVZTRA
CORI
CORR
COUC
COUS
CWFREQ
CWTIME
D2XUPCH2
D2XUPCH3
D4XUPCH2
D4XUPCH3
DATI
DEBU
DEFC
DEFS
DELA
DELO
DELR<1-5>
DELRFIXM
DEMOAMPL
DEMOOFF
DEMOPHAS
DISCUNIT
DISM
DISPDATA
DISPDATM
DISPDDM
DISPDMM
DISPMATH
DISPMEMO
DISPDPM
DISPDTM
DISPM1DM
DISPM1MM
DISPM1PM
DISPM1TM
DISPM2DM
DISPM2MM
DISPMMD
DISPMDD
DONE
DONM
DOSEQ<1-6>
DOWN
DRIVPORT
DUAC<ON|OFF>
EDITDONE
EDITLIML
EDITLIST
EDITRLIM
ELED
ENTO
EOCAL
ESE
ESNB
ESR?
EXTMDATA
EXTMFORM
EXTMGRAP
EXTMRAW
EXTT
FORM1
FORM2
FORM3
FORM4
FORM5
FREO
FRER
FWDI
FWDM
FWDT
HOLD
IDN?
IFBW
IMAG
INID
INPUCALC<01-04>
INPUCALK
INPUDATA
INPUFORM
INPULEAS
INPURAW1
INPURAW2
INPURAW3
INPURAW4
INTD
INTE
ISOD
ISOL
LIMD
LIMIAMPO
LIMILINE
LIMIMAOF
LIMISTIO
LIMITEST
LIML
LIMM
LIMS
LIMTFL
LIMTSL
LIMTSP
LIMU
LINFREQ
LINM
LINTDATA
LINTMEMO
LISFREQ
LISIFBWM
LISPWRM
LISTTYPELSTP
LISTTYPELSWP
LISV
LOAD<1-5>
LOGFREQ
LOGM
LWALCI
LWALCV?
LWALCVxx
MAN_LASER
MANTRIG
MARK1
MARK2
MARK3
MARK4
MARK5
MARK3DB
MARKBUCK
MARKCENT
MARKCONT
MARKCOUP
MARKCW
MARKDELA
MARKDISC
MARKFAUV
MARKFSTI
MARKFVAL
MARKMAXI
MARKMIDD
MARKMINI
MARKOFF
MARKREF
MARKSPAN
MARKSTAR
MARKSTIM
MARKSTOP
MARKUNCO
MARKZERO
MATI
MAXF
MEASA
MEASB
MEASR
MEASEO1
MEAS01
MEASOE1
MEASOE2
MEASOFF
MEASOO1
MEASTAT
MENUAVG
MEMO1
MEMO2
MEM1I
MEM2I
MENUCAL
MENUCOPY
MENUDISP
MENUFORM
MENUMARK
MENUMEAS
MENUMRKF
MENUOFF
MENUON
MENUPOWE
MENURECA
MENUSAVE
MENUSCAL
MENUSRCH
MENUSTIM
MENUSWEE
MENUSYST
MINMAX
MINU
MODEI
NEXP
NUMG
OMII
OPC
OPEP
OUTPACTI
OUTPAMAX
OUTPAMIN
OUTPAPER
OUTPCALC<01-12>
OUTPCALK
OUTPCHAN

OUTCNTR
OUTPDATA
OUTPDATP
OUTPDATR
OUTPERRO
OUTPFAIP
OUTPFARPLPT
OUTPFORM
OUTPIDEN
OUTPLEAS
OUTPLIM<1-4>
OUTPLIMF
OUTPLIML
OUTPLIMM
OUTPMARK
OUTMAXP
OUTPMEMO
OUTPMINP
OUTPMRIS
OUTPMSTA
OUTMUPL
OUTPMWID
OUTPOPTS
OUTPPLOT
OUTPPRIN
OUTPRAW<1-4>
OUTPRIPL
OUTPRPLBNDALL
OUTPRPLBNDPF
OUTPRPLBNDVAL
OUTPSEGAF
OUTPSEGAM
OUTPSEGF
OUTPSEGM
OUTPSERN
OUTPSTAT
OUTPTESS
OUTPTITL
PAUS
PHAO
PHAS
PLOT
POIN
POLA
POLMLIN
POLMLOG
POLMRI
PORE
PORT1
PORT2
PORTP
POWE
POWS
PRAN<01-12>
PRES
PWRR
PULV
PULW
RAID
RAIISOL
RAIRESP
RAMD
READDATE
READTIME
REAL
RECAREG <01-31>
RECEOUTON
RECEOUTOFF
RECO
REFD
REFL
REFP
REFT
REFV
RESC
RESD
RESM
RESPDONE
REST
REVI
REVM
REVT
RISTON
RISTOFF
RLIMLINE
RLIMM
RLIMSTP
RLIMSTR
RLIMTEST
RLIMVAL<ABS | MAR | OFF>
RST
S11
S12
S21
S22
SADD
SAV1
SAV2
SAVC
SAVECSV
SAVEJPG
SAVEREG<01-31>
SAVUASCI
SAVUBINA
SCAL
SDEL
SDON
SEAL
SEAMAX
SEAMIN
SEAOFF
SEAR
SEATARG
SEDI
SEGIFBW
SEGPOWER
SELBND
SELMAXPT
SELMINPT
SELPT
SELSEG
SETDATE
SETTIME
SETZ
SING
SLID
SMIC
SMIMGB
SMIMLIN
SMIMLOG
SMIMRI
SMIMRX
SMOOAPER
SMOOO<ON | OFF>
SOFR
SOFT<1-8>
SOUP
SPAN
SPLD
SPLID1
SPLID2
SPLID4
SRE
SSEG
STANA
STANB
STANC
STAND
STANE
STANF
STANG
STAR
STB?
STEPSWP
STOP
STPSIZE
SWEA
SWET
SWR
TESS?
TINT
TITF<1-5>
TITL
TITREG<01-31>
TRACK
TRAD
TRAN
TRIG
TST?
TSTP
UP
USEPASC
VELOFACT
WIDT
WIDV
WINDUSEM
WRSK<1-8>

OPC-Compatible List of Commands

AUXC	CLASS22A	FWDI	RECAREG	SAV2
CHAN1	CLASS22B	FWDM	REFD	SAVC
CHAN2	CLASS22C	FWDT	RESPDONE	SAVEJPG
CHAN3	CLEARALL	ISOD	REVI	SAVEREG
CHAN4	CLEAREG	MANTRIG	REVM	SING
CLASS11A	DATI	NUMG	REVT	STAN
CLASS11B	DONE	PRES	RST	TRAD
CLASS11C	EDITDONE	RAID	SAV1	

4

GPIB Programming

GPIB Programming

In this chapter, you can find an explanation of how various commands are derived from the analyzer front panel keys. There is also information on specific commands that can help you coordinate the timing of analyzer operations and command processing. You can find a general overview of GPIB operation, requirements, and capabilities. This chapter ends with information on remotely controlling a measurement calibration, display graphics, and assigning disk filenames.

Analyzer Command Syntax

Code Naming Convention

The analyzer GPIB commands are derived from their front-panel key titles (where possible), according to this naming convention:

Simple commands are the first four letters of the function they control, as in POWE, the command name for power. If the function label contains two words, the first three mnemonic letters are the first three letters of the first word, and the fourth mnemonic letter is the first letter of the second word. For example, ELED is derived from electrical delay.

If there are many commands grouped together in a category, as in markers, the command is increased to 8 letters. The first four letters are the category label and the last four letters are the function specifier. As an example, category markers are represented by the command MARK, which is used in combination with several functions such as MARKBUCK, MARKCENT.

The code naming guidelines, listed in the following table, are used in order to:

- make commands more meaningful and easier to remember
- maintain compatibility with other products (including the 8510 series analyzers)

NOTE There are times when these guidelines are not followed due to technical considerations.

Table 4-1. Code Naming Convention

Convention	Key Title	For GPIB Code Use	Example
One word	**Power** **Start**	First four letters	POWE STAR
Two words	**ELECTRICAL DELAY** **SEARCH RIGHT**	First three letters of first word, first letter of second word	ELED SEAR
Two words in a group	**MARKER → CENTER**	Four letters of both	MARKCENT
Three words	**FIXED MKR VALUE**	First four Letters of first word, first letter of second word, first three letters of third word	MARKFVAL

Some codes require appendages (ON, OFF, 1, 2, etc.). Codes that do not have a front-panel equivalent are GPIB only commands. They use a similar convention based on the common name of the function.

Valid Characters

The analyzer accepts the following ASCII characters:

- letters
- numbers
- decimal points
- +/–
- semicolons (;)
- quotation marks (“)
- carriage returns (CR)
- linefeeds (LF)

Both upper- and lower-case letters are acceptable. Carriage returns, leading zeros, spaces, and unnecessary terminators are ignored, except for those within a command or appendage. If the analyzer does not recognize a character as appropriate, it generates a syntax error message and recovers at the next terminator.

Units

The analyzer can input and output data in basic units such as Hz, dB, seconds, etc.

S	Seconds	HZ	Hertz
V	Volts	DB	dB or dBm

Input data is assumed to be in basic units (see above) unless one of the following units is used (upper and lower case are equivalent):

MS	Milliseconds	KHZ	Kilohertz
US	Microseconds	MHZ	Megahertz
NS	Nanoseconds	GHZ	Gigahertz
PS	Picoseconds	FS	Femtoseconds

Command Formats

The GPIB commands accepted by the analyzer can be grouped into five input-syntax types. The analyzer does not distinguish between upper- and lower-case letters.

General Structure:

The general syntax structure is: [code][appendage][data][unit][terminator]

The individual sections of the syntax code are explained below.

[code] The root mnemonic

[appendage] A qualifier attached to the root mnemonic. Possible appendages are `ON` or `OFF` (toggle a function on or off), or integers, which specify one capability out of several. There must be no spaces or symbols between the code and the appendage.

[data] A single operand used by the root mnemonic, usually to set the value of a function. The data can be a number or a character string. Numbers are accepted as integers or decimals, with power of ten specified by E (for example, `STAR 0.2E+10;` sets the start frequency to 2 GHz). Note the space between the root mnemonic and the operand. Character strings must be enclosed by double quotation marks.

For example:

A title string using RMB BASIC would look like:
`OUTPUT 716;"TITL""Unit1""";"`
where the first two "" are an escape so that RMB BASIC will interpret the third " properly.

[unit] The units of the operand, if applicable. If no units are specified, the analyzer assumes the basic units as described previously. The data is entered into the function when either units or a terminator are received.

[terminator] Indicates the end of the command, enters the data, and switches the active-entry area off. A semicolon (;) is the recommended terminator.

CAUTION Terminators are not necessary for the analyzer to interpret commands correctly. But in the event of a syntax error, the analyzer will attempt to recover at the next terminator. Therefore, it is recommended that each command include a terminator. The analyzer also interprets line feeds and GPIB end or identify (EOI) messages as terminators.

Syntax

Each command entry listed in Chapter 2, "Alphabetical Command Reference" includes the proper syntax for the command. The conventions used are as follows:

<num>	Required numerical data.
<choice1|choice2|...|choice*n*>	An appendage that is part of the command. For example, EDIT<DONE|LIML|LIST> indicates that the actual commands are EDIT DONE, EDITLIML, and `EDITLIST`.
<$>	Indicates a character string operand which must be enclosed by double quotes.
|	An either/or choice in appendages or optional data.
[ ]	Optional data.

Syntax Example For example, the following is provided as the syntax for the CHAN command:

```
CHAN<1|2|3|4>;
```

Analyzer Operation

Held Commands

The analyzer cannot process GPIB commands while executing certain key commands known as “held” commands. For example, `SING;` is a held command because it requires the analyzer to take one sweep of data before executing any other commands.

Once a held command is received, the analyzer will read new commands into the input buffer, but it will not begin the execution of any commands until the completion of the held command. When the 15-character input buffer is full, the analyzer will put hold on the bus until it is able to process the commands in the buffer.

NOTE Commands that call a calibration class are held if there is just one standard in the class, since such commands trigger a measurement.

Operation Complete

Occasionally, there is a need to know when certain analyzer operations have been completed. There is an operation-complete function (`OPC`) that allows a synchronization of programs with the execution of certain key commands. This mechanism is activated by issuing `OPC;` or `OPC?;` prior to an OPC-compatible command. The status byte or `ESR` operation-complete bit will then be set after the execution of the OPC-compatible command. For example, issuing `OPC;SING;` causes the OPC bit to be set when the single sweep is finished. Issuing `OPC?;` in place of `OPC;` causes the analyzer to output a one (1) when the command execution is complete. The analyzer will halt the computer by not transmitting the one (1) until the command has completed. For example, executing `OPC?;PRES;`, and then immediately querying the analyzer causes the bus to halt until the instrument preset is complete and the analyzer outputs a one (1).

As another example, consider the timing of sweep completion. Send the command string `SWET 3 S;OPC?;SING;` to the analyzer. This string sets the analyzer sweep time to 3 seconds, and then waits for completion of a single sweep to respond with a one (1). The computer should be programmed to read the number one (1) response from the analyzer indicating completion of the single sweep. At this point a valid trace exists and the trace data could be read into the computer.

For a list of OPC-compatible commands, refer to Chapter 3, “Command Listings”.

GPIB Operation

The general purpose interface bus (GPIB) is Agilent Technologies' hardware, software, documentation, and support for IEEE 488.2 and IEC-625 worldwide standards for interfacing instruments. This interface allows you to operate the analyzer and peripherals via two methods:

- with an external system controller
- with the analyzer in system-controller mode

Device Types

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

Every GPIB device must be capable of performing one or more of the following interface functions:

Talker

A “talker” is a device capable of transmitting device-dependent data when addressed to talk. There can be only one active talker at any given time. Examples of this type of device include:

- power meters
- disk drives
- voltmeters
- counters
- tape readers

The analyzer performs as a talker when it sends trace data or marker information over the bus.

Listener

A listener is a device capable of receiving device-dependent data over the interface when addressed to listen. There can be as many as 14 listeners connected to the interface at any given time. Examples of this type of device include:

- printers
- power supplies
- signal generators

The analyzer performs as a listener when it is controlled over the bus by a system controller.

Controller

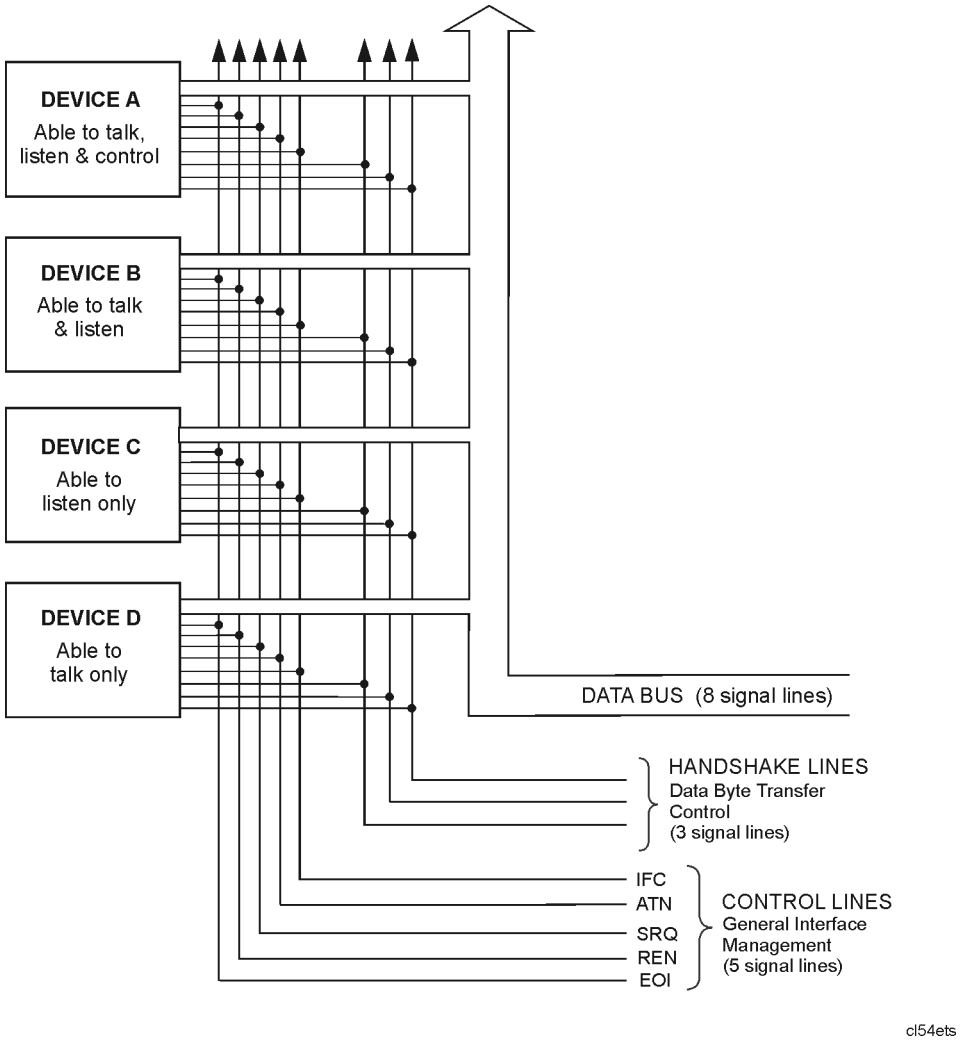
A controller is defined as a device capable of:

- managing the operation of the bus
- addressing talkers and listeners

There can only be one active controller on the interface at any time. Examples of controllers include desktop computers, minicomputers, workstations, and the analyzer. In a multiple-controller system, active control can be passed between controllers, but there can only be one *system* controller connected to the interface. The system controller acts as the master and can regain active control at any time. The analyzer is an active controller when it plots, prints, or stores to an external disk drive in the pass-control mode. The analyzer is also a system controller when it is operating in the system controller mode.

GPIB Bus Structure

Figure 4-1. GPIB Bus Structure



Data Bus

The data bus consists of 8 bidirectional lines that are used to transfer data from one device to another. Programming commands and data transmitted on these lines are typically encoded in ASCII, although binary encoding is often used to speed up the transfer of large arrays. Both ASCII and binary data formats are available to the analyzer. In addition, every byte transferred over GPIB undergoes a handshake to insure valid data.

Handshake Lines

A three-line handshake scheme coordinates the transfer of data between talkers and listeners. To ensure data integrity in multiple-listener transfers, this technique forces data transfers to occur at the transfer rate of the slowest device connected to the interface. With most computing controllers and instruments, the handshake is performed automatically, making it transparent to the programmer.

Control Lines

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

IFC (Interface Clear) This line is used exclusively by the system controller. When this line is true (low), all devices (whether addressed or not) unaddress and revert to an idle state.

ATN (Attention) The active controller uses this line to define whether the information on the data bus is command-oriented or data-oriented. 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 and the active controller services the requesting device. The analyzer can be enabled to pull the SRQ line for a variety of reasons such as requesting control of the interface, for the purposes of printing, plotting, or accessing a disk.

REN (Remote Enable) This line is used exclusively by the system controller. When this line is set true (low), the bus is in the remote mode, and devices are addressed by the controller to either listen or talk. When the bus is in remote mode and a device is addressed, it receives instructions from the system controller via GPIB rather than from its front panel (pressing **Local** returns the device to front-panel operation). When this line is set false (high), the bus and all of the connected 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.

GPIB 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 the specified distances with tri-state drivers. Actual data rate depends on the transfer rate of the slowest device connected to the bus.

Address Capability:

Primary addresses: 31 talk, 31 listen. A maximum of 1 talker and 14 listeners can be connected to the interface at given time.

Multiple-Controller Capability:

In systems with more than one controller (such as this instrument), only one controller can be active at any given 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.

GPIB Operational Capabilities

On the analyzer's rear panel, next to the GPIB connector, there is a list of GPIB device subsets as defined by the IEEE 488.2 standard. The analyzer has the following capabilities:

SH1	Full-source handshake.
AH1	Full-acceptor handshake.
T6	Basic talker, answers serial poll, unaddresses if MLA is issued. No talk-only mode.
L4	Basic listener, unaddresses if MTA is issued. No listen-only mode.
SR1	Complete service request (SRQ) capabilities.
RL1	Complete remote/local capability including local lockout.
PP0	Does not respond to parallel poll.
DC1	Complete device clear.
DT1	Responds to a Group Execute Trigger (GET) in the hold-trigger mode.
C1,C2,C3	System controller capabilities in system-controller mode.
C10	Pass control capabilities in pass-control mode.
E2	Tri-state drivers.
LE0	No extended listener capabilities.
TE0	No extended talker capabilities.

These codes are completely explained in the IEEE Std 488 documents, published by:

The Institute of Electrical and Electronic Engineers, Inc.
345 East 47th Street
New York, New York 11017

Or, visit their Web site at http://standards.ieee.org

GPIB Status Indicators

When the analyzer is connected to other instruments over the GPIB, the GPIB status indicators illuminate to display the current status of the analyzer. The GPIB status indicators are located in the instrument-state function block on the front panel of the analyzer.

R = Remote Operation

L = Listen mode

T = Talk mode

S = Service request (SRQ) asserted by the analyzer

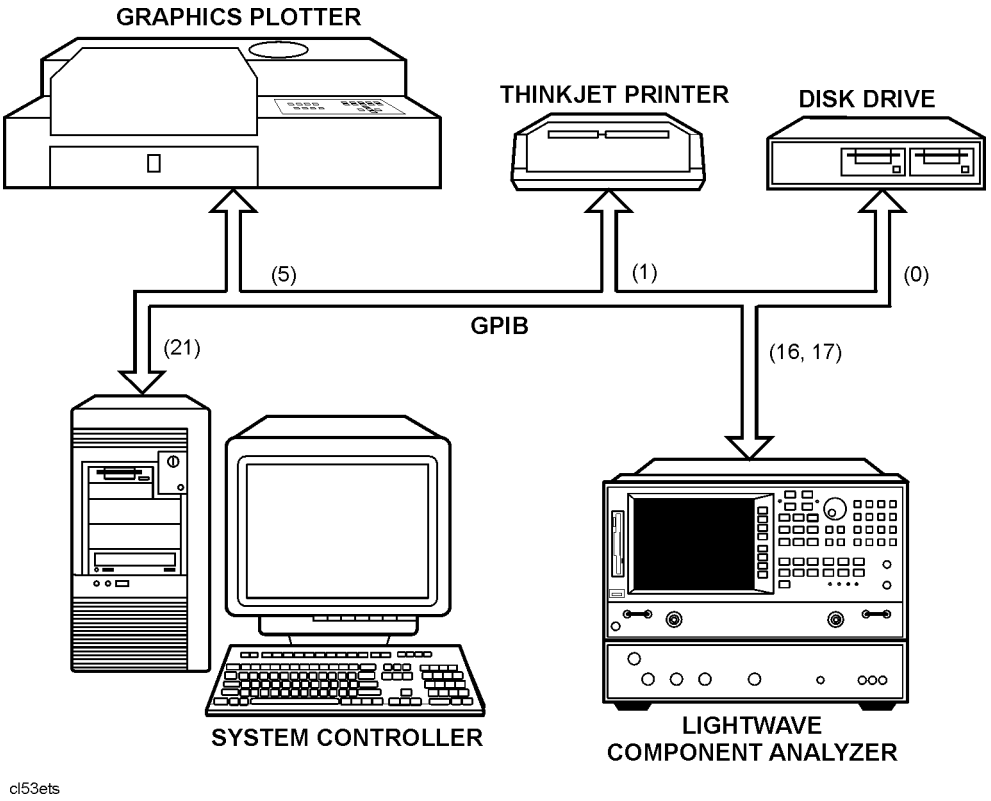
Bus Device Modes

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

Three different controller modes are possible in an GPIB system:

- system-controller mode
- talker/listener mode
- pass-control mode

Figure 4-2. Analyzer Single Bus Concept



System-Controller Mode

This mode allows the analyzer to control peripherals directly in a stand-alone environment (without an external controller). This mode must be selected manually from the analyzer's front panel. It can only be used if no active computer or instrument controller is connected to the system via GPIB. If an attempt is made to set the analyzer to the system-controller mode when another controller is connected to the interface, the following message is displayed on the analyzer's display screen:

"ANOTHER SYSTEM CONTROLLER ON GPIB BUS"

The analyzer must be set to the system-controller mode in order to access peripherals from the front panel. In this mode, the analyzer can directly control peripherals (plotters, printers, disk drives, power meters, etc.) and the analyzer may plot, print, store on external disk or perform power meter functions.

NOTE Do not attempt to use this mode for programming. Agilent recommends using an external instrument controller when programming. See the following section, “Talker/Listener Mode.”

Talker/Listener Mode

This is the mode that is normally used for remote programming of the analyzer. In talker/listener mode, the analyzer and all peripheral devices are controlled from an external instrument controller. The controller can command the analyzer to talk and other devices to listen. The analyzer and peripheral devices cannot talk directly to each other unless the computer sets up a data path between them. This mode allows the analyzer to act as either a talker or a listener, as required by the controlling computer for the particular operation in progress.

Pass-Control Mode

This mode allows the computer to control the analyzer via GPIB (as with the talker/listener mode), but also allows the analyzer to take control of the interface in order to plot, print, or access a disk. During an analyzer-controlled peripheral operation, the host computer is free to perform other internal tasks (i.e. data or display manipulation) while the analyzer is controlling the bus. After the analyzer-controlled task is completed, the analyzer returns control to the system controller.

NOTE Performing an instrument preset does not affect the selected bus mode, although the bus mode will return to talker/listener mode if the line power is cycled. To set the bus mode from the front panel, use the **Local** key.

Analyzer Bus Modes

As discussed earlier, under GPIB control, the analyzer can operate in one of three modes: talker/listener, pass-control, or system-controller mode.

In talker/listener mode, the analyzer behaves as a simple device on the bus. While in this mode, the analyzer can make a plot or print using the `OUTPPLOT;` or `OUTPPRIN;` commands. The analyzer will wait until it is addressed to talk by the system controller and then dump the display to a plotter/printer that the system controller has addressed to listen. Use of the commands `PLOT;` and `PRINTALL;` require control to be passed to another controller.

In pass-control mode, the analyzer can request control from the system controller and take control of the bus if the controller addresses it to take control. This allows the analyzer to take control of printers, plotters, and disk drives on an as-needed basis. The analyzer sets event-status register bit 1 when it needs control of the interface, and the analyzer will transfer control back to the system controller at the completion of the operation. It will pass control back to its controller address, specified by `ADDRCONT`.

The analyzer can also operate in the system-controller mode. This mode is only used when there is no remote controller on the bus. In this mode, the analyzer takes control of the bus, and uses it whenever it needs to access a peripheral. While the analyzer is in this mode, no other devices on the bus can attempt to take control. Specifically, the REN, ATN, and IFC lines must remain unasserted, and the data lines must be freed by all but the addressed talker.

Setting GPIB Addresses

In systems interfaced using GPIB, each instrument on the bus is identified by an GPIB address. This address code must be different for each instrument on the bus. These addresses are stored in non-volatile memory and are not affected when you press **Preset** or cycle the power. The analyzer occupies two GPIB addresses: the instrument itself and the display. The display address is derived from the instrument address by complementing the instrument's least-significant bit. Hence, if the instrument is at an even address, the display occupies the next higher address. If the instrument is at an odd address, the display occupies the next lower address. See “Display Graphics” on page 4-22.

The analyzer addresses are set by pressing **Local SET ADDRESSES**. In system-controller mode, the addresses must be set for the plotter, printer, disk drive, and power meter.

The default address for the analyzer is device 16, and the display address is device 17.

NOTE There is also an address for the system controller. This address refers to the controller when the analyzer is being used in pass-control mode. This is the address that control is passed back to when the analyzer-controlled operation is complete.

Response to GPIB Meta-Messages (IEEE-488 Universal Commands)

Abort

The analyzer responds to the abort message (IFC) by halting all listener, talker, and controller functions.

Device Clear

The analyzer responds to the device clear commands (DCL, SDC) by clearing the input and output queues, and clearing any GPIB errors. The status registers and the error queue are unaffected.

Local

The analyzer will go into local mode if the local command (GTL) is received, the remote line is unasserted, or the front-panel local key is pressed. Changing the analyzer's GPIB status from remote to local does not affect any of the front-panel functions or values.

Local Lockout

If the analyzer receives the local-lockout command (LLO) while it is in remote mode, it will disable the entire front panel except for the line power switch. A local-lockout condition can only be cleared by releasing the remote line, although the local command (GTL) will place the instrument temporarily in local mode.

Parallel Poll

The analyzer does not respond to parallel-poll configure (PPC) or parallel-poll unconfigure (PPU) messages.

Pass Control

If the analyzer is in pass-control mode, is addressed to talk, and receives the take-control command (TCT), from the system control it will take active control of the bus. If the analyzer is not requesting control, it will immediately pass control to the system controller's address. Otherwise, the analyzer will execute the function for which it sought control of the bus and then pass control back to the system controller.

Remote

The analyzer will go into remote mode when the remote line is asserted and the analyzer is addressed to listen. While the analyzer is held in remote mode, all front-panel keys (with the exception of **Local**) are disabled. Changing the analyzer's GPIB status from remote to local does not affect any front-panel settings or values.

Serial Poll

The analyzer will respond to a serial poll with its status byte, as defined in “Error Reporting” on page 7-2. To initiate the serial-poll sequence, address the analyzer to talk and issue a serial-poll enable command (SPE). Upon receiving this command, the analyzer will return its status byte. End the sequence by issuing a serial-poll disable command (SPD). A serial poll does not affect the value of the status byte, and it does not set the instrument to remote mode.

Trigger

In hold mode, the analyzer responds to device trigger by taking a single sweep. The analyzer responds only to selected-device trigger (SDT). This means that it will not respond to group execute-trigger (GET) unless it is addressed to listen. The analyzer will not respond to GET if it is not in hold mode.

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: `AGILENT,8703B,0,X.XX` where X.XX is the firmware revision of the instrument.
`*LRN?`	This returns a binary 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 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).
`*OPC?`	Operation complete query. The analyzer will return an ASCII "1" when all pending overlapped operations have been completed.
`*PCB <num>`	Sets the pass-control-back address (the address of the controller before a pass control is executed).
`*RST`	Executes a device reset and cancels any pending `*OPC` command or query.
`*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.
`*WAI`	Prohibits the instrument from executing any new commands until all pending overlapped commands have been completed.

Calibration

Measurement calibration over GPIB follows the same command sequence as a calibration from the front-panel. For detailed information on measurement calibration, refer to your analyzer's user's guide.

1. Start by selecting a calibration kit, such as 3.5mm(`CALK35MM;`).
2. Select a calibration type, such as S11 1-port (`CALIS111;`).
3. Call each class used by the calibration type, such as FORWARD: OPEN (`CLASS11A;`) During a 2-port calibration, the reflection, transmission, and isolation subsequences must be opened before the classes in the subsequence are called, and then closed at the end of each subsequence.
4. If a class has more than one standard in it, select a standard from the menu presented (`STANA` to `STANG`).
5. If, during a calibration, two standards are measured to satisfy one class, the class must be closed with `DONE;`.
6. Declare the calibration done, such as with DONE 1-PORT CAL (`SAV1;`).

The `STANA` to `STANG` commands will hold off the GPIB until completion because they trigger a sweep. If a class has only one standard in it, which means that it will trigger a sweep when called, the class command will also hold off the GPIB.

NOTE Since different cal kits can have a different number of standards in a given class, any automated calibration sequence is valid only for a specific cal kit.

Table 4-2. Relationship between Calibrations and Classes

Class	Response	Response & Isolation	S11 1-port	S22 1-port	One path 2-port	Full 2-port	TRL/ LRM	Fwd Enh Resp	Rev Enh Resp	E/O Resp & Match	O/E Resp & Match
Reflection:[a]					•	•	•	•	•		•
S11A, RE FW MTCH			•		•	•	•	•		•	•
S11B, LN FW MTCH			•		•	•	•	•		•	•
S11C, LN FW TRAN			•		•	•	•	•		•	•
S22A, LN RV MTCH				•		•	•		•		•
S22B, LN RV TRAN				•		•	•		•		•
S22C, LN RV TRAN				•		•	•		•		•
Transmission:[a]					•	•	•	•	•	•	•
Forward match					•	•	•	•			•
Forward trans					•	•	•	•		•	•
Reverse match						•	•		•		
Reverse trans						•	•		•		
Isolation:[a]					•	•	•	•	•	•	•
Forward					•	•	•	•		•	•
Reverse						•	•		•		
Response	•										
Response and isolation:											
Response		•									
Isolation		•									
TRL thru:[b]							•				
TRL reflect:[b]							•				
TRL line or match[b]:							•				

a. These subheadings must be called when doing full 2-port calibrations.
b. These subheadings must be called when doing TRL 2-port calibrations

Table 4-3. Error Coefficient Arrays

Array	Response	Response & Isolation	1-port	Enhanced Response	2-port [a]	TRL/ LRM	O/E Response & Match	E/O Response & Match
01	E_R or E_T	E_X (E_D)[b]	E_D	E_D	E_{DF}	E_{DF}	E_{DR}	E_{DI}
02		E_T (E_R)	E_S	E_S	E_{SF}	E_{SF}	E_{SR}	E_S
03			E_R	E_R	E_{RF}	E_{RF}	E_{RR}	E_R
04				E_X	E_{XF}	E_{XF}	E_{XF}	E_X
05				E_L[c]	E_{LF}	E_{LF}	E_{LF}	E_T
06				E_T	E_{TF}	E_{TF}	E_{TF}	
07					E_{DR}	E_{DR}	E_{DF}	
08					E_{SR}	E_{SR}	E_{SF}	
09					E_{RR}	E_{RR}	E_{RF}	
10					E_{XR}	E_{XR}		
11					E_{LR}	E_{LR}		
12					E_{TR}	E_{TR}		

a. One path, 2-port cal duplicates arrays 1 to 6 in arrays 7 to 12.
b. Response and isolation corrects for crosstalk and transmission tracking in transmission measurements, and for directivity and reflection tracking in reflection measurements.
c. This term is used to generate the calibration coefficients, but is not used during measurement error correction.

Meaning of first subscript:
D: directivity
S: source match
R: reflection tracking
X: crosstalk or isolation
L: load match
T: transmission tracking

Meaning of second subscript:
F: forward
R: reverse

Display Graphics

User Graphics Units

Size of graticule only:

- length = 350 to 4915
- height = 150 to 3950

Size of complete display (graticule plus annotation and softkey labels)

- length = 0 to 5850
- height = 0 to 4095

HPGL Commands

AF

Erases the user graphics display.

CS

Turns off the measurement display.

DIrun,rise

Specifies the direction in which characters are lettered.

run,rise	Direction
1,0	0 degrees
0,1	90 degrees
–1,0	180 degrees
0,–1	270 degrees

DF

Sets the default values.

LB<text><etx>

Labels the display, placing the symbols starting at the current pen position. All incoming characters are printed until the etx symbol is received. The default etx symbol is the ASCII value 3 (not the character 3).

LTa

Specifies line type:

a	line type
0	solid
1	solid
2	short dashes
3	long dashes

OP

Outputs P1 and P2, the scaling limits: 0,0,5850,4095.

PAx,y

Draws from the current pen position to x,y. There can be many pairs of x,y coordinates within one command. They are separated by commas, and the entire sequence is terminated with a semicolon.

PD

Pen down. A line is drawn only if the pen is down.

PG

Erases the user graphics display.

PRx,y

Plot relative: draws a line from the current pen position to a position y up and x over.

PU

Pen up. Stops anything from being drawn.

RS

Turns on the measurement display.

SIh,w

Sets the character size, for height h and width w in centimeters:

h	w	char size
0.16	0.20	smallest
0.25	0.30	
0.33	0.39	
0.41	0.49	largest

SPn

Selects pen n:

n	brightness
0	blank
1	yellow
2	green
3	light blue
4	light red
5	white
6	red
7	blue

Accepted but ignored HPGL commands

Command	Description
IM	Input service request mask
IP	Input Pl,P2 scaling points
IW	Input window
OC	Output current pen position
OE	Output error
OI	Output identity
OS	Output status
SL	Character slant
SR	Relative character size

Disk File Names

Disk files created by the analyzer consist of a state name of up to eight characters, such as "FILTER," appended with up to two characters. In LIF format, the file name is "FILTERXX." In DOS format, the filename is "FILTER.XX." The first appended character is the file type, telling the kind of information in the file. The second appended character is a data index, used to distinguish files of the same type.

Error-corrected data, raw data, formatted data, memory traces, and calibration files are FORM 3 data files (IEEE 64-bit floating point format). The other files are not meant to be decoded. The following table lists the appended characters and their meanings.

Table 4-4. File Suffix Character Meaning

Char 1	Meaning	Char 2	Meaning
I, P	Instrument state		
W	Four-channel instrument state		
G	Graphics	1	Display graphics
D	Error-corrected data	1 2 3 4	Channel 1 Channel 2 Channel 3 Channel 4
		2	Channel 2
R	Raw data	1 to 4	Channel 1/Channel 3, raw arrays 1 to 4
		5 to 8	Channel 2/Channel 4, raw arrays 1 to 4
F	Formatted data	1 2 3 4	Channel 1 Channel 2 Channel 3 Channel 4
M	Memory trace	1 2 3 4	Channel 1 Channel 2 Channel 3 Channel 4
C	Cal kit	K	
1	Cal data, channel 1	O 1 to 9 A B C	Stimulus state Coefficients 1 to 9 Coefficient 10 Coefficient 11 Coefficient 12
2	Cal data, channel 2	0 to C	Same as channel 1

Table 4-4. File Suffix Character Meaning

Char 1	Meaning	Char 2	Meaning
F	Full page (HPGL plot)	P	
L	Left (HPGL plot)	L U	Lower Upper
R	Right (HPGL plot)	L U	Lower Upper
S	Error-corrected data (S2P)	1 2	Channel 1 Channel 2

5

Reading Analyzer Data

Reading Analyzer Data

In this chapter, you can find information on how to remotely query the analyzer for measurement data, extract the data, and then transfer the data to a medium for later analysis.

Output Queue

Whenever an output-data command is received, the analyzer puts the data into the output queue (or buffer) where it is held until the system controller outputs the next read command. The queue, however, is only one event long: the next output-data command will overwrite the data already in the queue. Therefore, it is important to read the output queue immediately after every query or data request from the analyzer.

Command Query

All instrument functions can be queried to find the current on/off state or value. For instrument state commands, append the question mark character (?) to the command to query the state of the functions. Suppose the operator has changed the power level from the analyzer's front panel. The computer can ascertain the new power level using the analyzer's command-query function. If a question mark is appended to the root of a command, the analyzer will output the value of that function. For instance, `POWE 7DB;` sets the source power to 7 dB, and `POWE?;` outputs the current RF source power at the test port. When the analyzer receives `POWE?;`, it prepares to transmit the current RF source power level. This condition illuminates the analyzer front-panel talk light (T). In this case, the analyzer transmits the output power level setting to the controller.

On/off commands can also be queried. The reply is a one (1) if the function is on, or a zero (0) if it is off. For example, if a command controls an active function that is underlined on the analyzer display, querying that command yields a one (1) if the command is underlined or a zero (0) if it is not. As another example, there are nine options on the format menu and only one option is underlined at a time. Only the underlined option will return a one when queried.

For instance, send the command string `DUAC?;` to the analyzer. If dual-channel display is switched on, the analyzer will return a one (1) to the instrument controller.

Similarly, to determine if phase is being measured and displayed, send the command string `PHAS?;` to the analyzer. In this case, the analyzer will return a one (1) if phase is currently being displayed. Since the command only applies to the active channel, the response to the `PHAS?;` query depends on which channel is active.

Identification

The analyzer's response to `IDN?;` is “`AGILENT,8703B,xxxxxxxxxx,X.XX`” where xxxxxxxxxx is the serial number, and X.XX is the firmware revision of the instrument.

The analyzer also has the capability to output its serial number with the command `OUTPSERN;`, and to output its installed options with the command `OUTPOPTS;`.

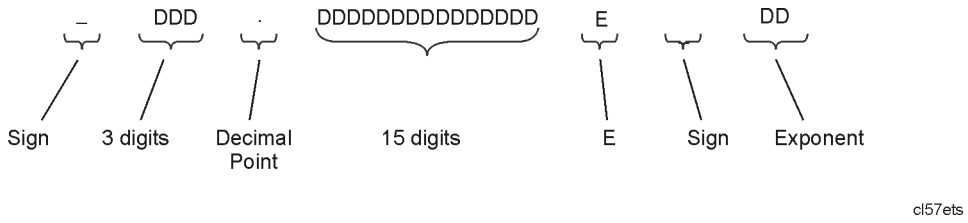
Output Syntax

The following three types of data are transmitted by the analyzer in ASCII format:

- response to query
- certain output commands
- ASCII floating-point (FORM4) array transfers

Marker-output commands and queried commands are output in ASCII format only, meaning that each character and each digit is transmitted as a separate byte, leaving the receiving computer to reconstruct the numbers and strings. Numbers are transmitted as 24-character strings, consisting of:

Figure 5-1. FORM4 (ASCII) Data-Transfer Character String



Sign	'–' for negative, blank for positive.
3 digits	Digits to the left of the decimal point.
Decimal Point	Standard decimal point.
15 digits	Digits to the right of the decimal point.
E	Exponent notation.
Sign (in exponent)	'–' for negative, '+' for positive.
Exponent	Two digits for the exponent.

When multiple numbers are sent, the numbers are separated by commas. When number pairs are sent, the numbers are separated by a comma and terminated with a line feed, $<^{L}_{F}>$.

Marker Data

The analyzer offers several options for outputting trace-related data. Data can be selectively read from the trace using the markers, or the entire trace can be read by the controller. If only specific information is required (such as a single point on the trace or the result of a marker search), the marker output command can be used to read the information. Specific data points can be read using the OUTPDATP or OUTPDATR commands. These commands allow a much faster data transfer than when using markers to output specific data points.

A marker must first be assigned to the desired frequency before it can be used to read the trace data. This is accomplished using the marker commands. The controller sends a marker command followed by a frequency within the trace-data range. If the actual desired frequency was not sampled, the markers can be set to continuous mode and the desired marker value will be linearly interpolated from the two nearest points. This interpolation can be prevented by putting the markers into discrete mode. Discrete mode allows the marker to only be positioned on a measured trace-data point.

As an alternative, the analyzer can be programmed to choose the stimulus value by using the **Marker Search** functions. Maximum, minimum, target value, or bandwidth search can be automatically determined with **Marker Search** functions. To continually update the search, switch the marker tracking on. The trace-value search will remain activated until one of the following occurs:

- The search is switched off.
- The tracking is switched off.
- All markers are switched off.

Marker data can be output to a controller by using analyzer commands. These commands cause the analyzer to transmit three numbers: marker value 1, marker value 2, and marker stimulus value. For example, in log-magnitude display mode we get the log magnitude at the marker (value 1), zero (value 2), and the marker frequency. Table 5-1, “Units as a Function of Display Format,” on page 5-6 for a complete listing of all the possibilities for values 1 and 2. The possibilities for the marker stimulus value are:

- frequency
- CW time
- power (in power sweep mode)

Table 5-1. Units as a Function of Display Format

Display Format	Marker Mode	OUTPMARK		OUTPFORM		Marker Readout[a]	
		Value 1	Value 2	Value 1	Value 2	Value	Aux Value
LOG MAG		dB	N/S[b]	dB	N/S[b]	dB	N/S
PHASE		degrees	N/S[b]	degrees	N/S[b]	degrees	N/S
DELAY		seconds	N/S[b]	seconds	N/S[b]	seconds	N/S
SMITH CHART	LIN MKR	lin mag	degrees	real	imag	lin mag	degrees
	LOG MKR	dB	degrees	real	imag	dB	degrees
	Re/Im	real	imag	real	imag	real	imag
	R + jX	real ohms	imag ohms	real	imag	real ohms	imag ohms
	G + jB	real Siemens	imag Siemens	real	imag	real Siemens	imag Siemens
POLAR	LIN MKR	lin mag	degrees	real	imag	lin mag	degrees
	LOG MKR	dB	degrees	real	imag	dB	degrees
	Re/Im	real	imag	real	imag	real	imag
LIN MAG		lin mag	N/S[b]	lin mag	N/S[b]	lin mag	N/S
SWR		SWR	N/S[b]	SWR	N/S[b]	SWR	N/S
REAL		real	N/S[b]	real	N/S[b]	real	N/S
IMAGINARY		imag	N/S[b]	imag	N/S[b]	imag	N/S

a. The marker readout values are the marker values displayed in the upper right-hand corner of the display. They also correspond to the value and auxiliary value associated with the fixed marker.
b. Value 2 is not significant in this format, though it is included in data transfers. .

Array-Data Formats

The analyzer can transmit and receive arrays in the analyzer's internal binary format as well as four different numeric formats. The current format is set with the FORM1, FORM2, FORM3, FORM4, and FORM5 commands. These commands do not affect learnstring transfers, calibration-kit string transfers, or non-array transfers, such as command query, or output marker values.

A transmitted array will be output in the current format, and the analyzer will attempt to read incoming arrays according to the current format. Each data point in an array is a pair of numbers, usually a real/imaginary pair. The number of data points in each array is the same as the number of points in the current sweep.

The five formats are described below:

FORM1 The analyzer's internal binary format, 6 bytes-per-data point. The array is preceded by a four-byte header. The first two bytes represent the string "#A", the standard block header. The second two bytes are an integer representing the number of bytes in the block to follow. FORM1 is best applied when rapid data transfers, not to be modified by the computer nor interpreted by the user, are required.

FORM2 IEEE 32-bit floating-point format, 4 bytes per number, 8 bytes-per-data point. The data is preceded by the same header as in FORM1. Each number consists of a 1-bit sign, an 8-bit biased exponent, and a 23-bit mantissa. FORM2 is the format of choice if your computer is not a PC, but supports single-precision floating-point numbers.

FORM3 IEEE 64-bit floating-point format, 8 bytes per number, 16 bytes-per-data point. The data is preceded by the same header as in FORM1. Each number consists of a 1-bit sign, an 11-bit biased exponent, and a 52-bit mantissa. This format may be used with double-precision floating-point numbers. No additional precision is available in the analyzer data, but FORM3 may be a convenient form for transferring data to your computer.

FORM4 ASCII floating-point format. There is no header. The analyzer always uses FORM4 to transfer data that is not related to array transfers (i.e. marker responses and instrument settings).

FORM5 PC-DOS 32-bit floating-point format with 4 bytes-per-number, 8 bytes-per-data point. The data is preceded by the same header as in FORM1. The byte order is reversed with respect to FORM2 to comply with PC-DOS formats. If you are using a PC-based controller, FORM5 is the most effective format to use.

The analyzer terminates each transmission by asserting the EOI interface line with the last byte transmitted. Table 5-2 offers a comparative overview of the five array-data formats.

Table 5-2. Analyzer Array-Data Formats

Format Type	Type of Data	Bytes per Data Value	Bytes per point 2 data values	(201 pts) Bytes per trace	Total Bytes with header
FORM 1	Internal Binary	N/A	6	1206	1210
FORM 2	IEEE 32-bit Floating-Point	4	8	1608	1612
FORM 3	IEEE 64-bit Floating-Point	8	16	3216	3220
FORM 4	ASCII Numbers	24 (Typical)	50 (Typical)	10,050 (Typical)	10,050[a] (Typical)
FORM 5	PC-DOS 32-bit Floating-Point	4	8	1608	1612

a. FORM4 does not use a header.

Trace-Data Transfers

Transferring trace-data from the analyzer using an instrument controller can be divided into three steps:

1. Allocating an array to receive and store the data
2. Commanding the analyzer to transmit the data
3. Accepting the transferred data

Data residing in the analyzer is always stored in pairs for each data point (to accommodate real/imaginary pairs). Hence, the receiving array must be two elements wide, and as deep as the number of points in the array being transferred. Memory space for the array must be declared before any data can be transferred from the analyzer to the computer.

As mentioned earlier, the analyzer can transmit data over GPIB in five different formats. The type of format affects what kind of data array is declared (real or integer), because the format determines what type of data is transferred.

Stimulus-Related Values

Frequency-related values are calculated for the analyzer display. The start and stop frequencies or center and span frequencies of the selected frequency range are available to the programmer.

In a linear frequency range, the frequency values can be easily calculated because the trace data points are equally spaced across the trace. Relating the data from a linear frequency sweep to frequency can be done by querying the start frequency, the frequency span, and the number of points in the trace.

Given that information, the frequency of point n in a linear-frequency sweep is represented by the equation:

$$F = StartFrequency + (n-1) \times \frac{Span}{(Points-1)}$$

In most cases, this is an easy solution for determining the related frequency value that corresponds with a data point. This technique is illustrated in "Example 3B: Data Transfer Using FORM 4 (ASCII Transfer)" on page 8-66.

When using log sweep or a list-frequency sweep, the points are not evenly spaced over the frequency range of the sweep. In these cases, an effective way of determining the frequencies of the current sweep is to use the OUTPLIML command. Although this command is normally used for limit lines, it can also be used to identify all of the frequency points in a sweep. Limit lines do not need to be on in order to read the frequencies directly out of the instrument with the OUTPLIML command. Refer to "Example 3D: Data Transfer Using Frequency-Array Information" on page 8-70.

NOTE Another method of identifying all of the frequency points in a sweep is to use the marker commands MARKBUCK<num> and OUTPMARK in a "FOR NEXT" programming loop that corresponds to the number of points in the sweep. MARKBUCK<num> places a marker at a point in the sweep, where <num> is the number of the point in a sweep, and OUTPMARK outputs the stimulus value as part of the marker data.

6

Data Processing Chain

Data Processing Chain

This chapter contains descriptions of how the analyzer processes measurement data.

NOTE Refer to "OUTP" on page 2-49 for detailed information (such as proper syntax) for the output commands discussed in this chapter.

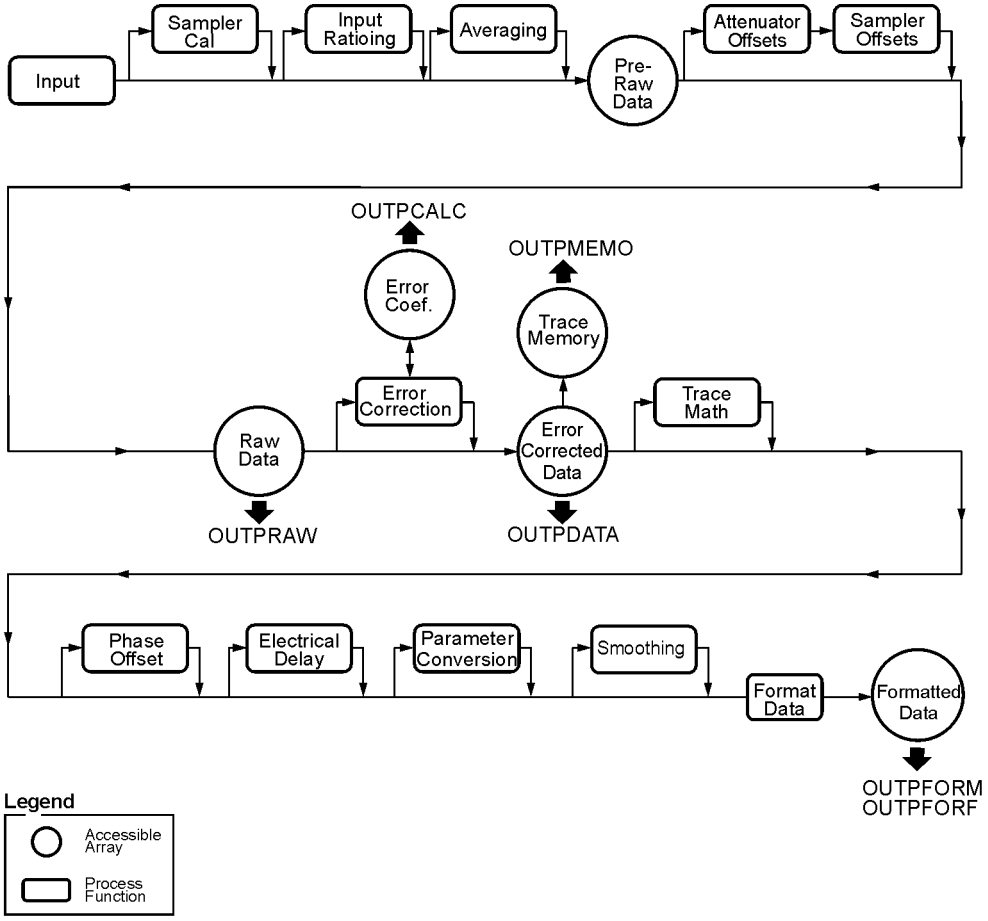
Data Arrays

The following figure shows the different kinds of data available within the instrument:

- pre-raw measured data
- raw measured data
- error-corrected data
- formatted data
- trace memory
- calibration coefficients

Trace memory can be directly output to a controller with `OUTPMEMO;`, but it cannot be directly transmitted back.

Figure 6-1. The Data-Processing Chain For Measurement Outputs



Common Output Commands

All the data-output commands are designed to insure that the data transmitted reflects the current state of the instrument:

- OUTPDATA, OUTPRAW, and OUTPFORM will not transmit data until all formatting functions have completed.
- OUTPLIML, OUTPLIMM, and OUTPLIMF will not transmit data until the limit test has occurred (if activated).
- OUTPMARK will activate a marker if a marker is not already selected. It will also insure that any current marker searches have been completed before transmitting data.
- OUTPMSTA ensures that the statistics have been calculated for the current trace before transmitting data. If the statistics are not activated, it will activate the statistics long enough to update the current values before deactivating the statistics.

- OUTPMWID ensures that a bandwidth search has been executed for the current trace before transmitting data. If the bandwidth-search function is not activated, it will activate the bandwidth-search function long enough to update the current values before switching off the bandwidth-search functions.

Fast Data Transfer Commands

The analyzer has a distinct fast data transfer command. These command circumvents the internal “byte handler” routine and output trace dumps as block data. In other words, the analyzer outputs the entire array without allowing any process swapping to occur. FORM4, ASCII data transfer times are not affected by these routines. However, there are speed improvements with binary data formats. The following is a description of the fast data transfer command:

- OUTPFORF outputs the formatted display trace array from the active channel in the current output format. Only the first number in each of the OUTPFORM data pairs is actually transferred for the display formats LOG MAG, PHASE, group DELAY, LIN MAG, SWR, REAL and IMAGinary. Because the data array does not contain the second value for these display formats, the INPUFORM command may not be used to re-input the data back into the analyzer. The second value may not be significant in some display formats thus reducing the number of bytes transferred.

Data Levels

Different levels of data can be read out of the instrument. The following sections describe the different types of data that are available from the analyzer.

Raw Data

The basic measurement data, reflecting the stimulus parameters, IF averaging, and IF bandwidth. If a full 2-port measurement calibration is activated, there are actually four raw arrays kept: one for each raw S-parameter. The data can be output to a controller with the OUTPRAW commands. Normally, only raw 1 is available, and it holds the current parameter. If a 2-port measurement calibration is active, the four arrays refer to S_{11}, S_{21}, S_{12}, and S_{22} respectively. This data is represented in real/imaginary pairs.

Error Coefficients

The results of a measurement calibration are arrays containing error coefficients. These error coefficients are then used in the error-correction routines. Each array corresponds to a specific error term in the error model. Your analyzer's user's guide details which error coefficients are used for specific calibration types, as well as the arrays those coefficients can be found in. Not all calibration types use all 12 arrays. The data is stored as real/imaginary pairs.

Error-Corrected Data

This is the raw data with error-correction applied. The array represents the currently measured parameter, and is stored in real/imaginary pairs. The error-corrected data can be output to a controller with the `OUTPDATA;` command. The `OUTPMEMO;` command reads the trace memory, if available. The trace memory also contains error-corrected data. Note that neither raw nor error-corrected data reflect such post-processing functions as electrical-delay offset, or trace math.

Formatted Data

This is the array of data actually being displayed. It reflects all post-processing functions such as electrical delay. The units of the array output depend on the current display format.

Generally, formatted data is the most useful of the five data levels, because it is the same information the operator sees on the display. However if post-processing is unnecessary (such as in some cases involving smoothing), error-corrected data may be more desirable. Error-corrected data also affords the user the opportunity to input the data to the analyzer and apply post-processing at another time.

Learnstring and Calibration-Kit String

The learn string is a summary of the instrument state. It includes all the front-panel settings, the limit-test tables, and the list-frequency table for the current instrument state. It does not include calibration data or the information stored in the save/recall registers.

The learn string can be output to a controller with the `OUTPLEAS;` command, which commands the analyzer to start transmitting the binary string. The string has a fixed length for a given firmware revision. The array has the same header as in FORM 1. Refer to "Example 5A: Using the Learn String" on page 8-84.

The calibration kit includes a set of key characteristics of the calibration standards used to determine the calibration accuracy. There are default kits for several different connector types. There is also space for a user-defined calibration kit. The command `OUTPCALK` outputs the currently active calibration kit as a binary string in FORM 1. As with the learn string, the calibration-kit string has a fixed length for a given firmware revision.

7

Error Reporting

Error Reporting

This chapter contains descriptions of analyzer error reporting process. The descriptions include how the analyzer reports errors to a status register and how you can cause the register to output the errors messages.

Status Reporting

The analyzer status reporting structure is depicted in the following figure and tables.

Figure 7-1. Status Reporting Structure

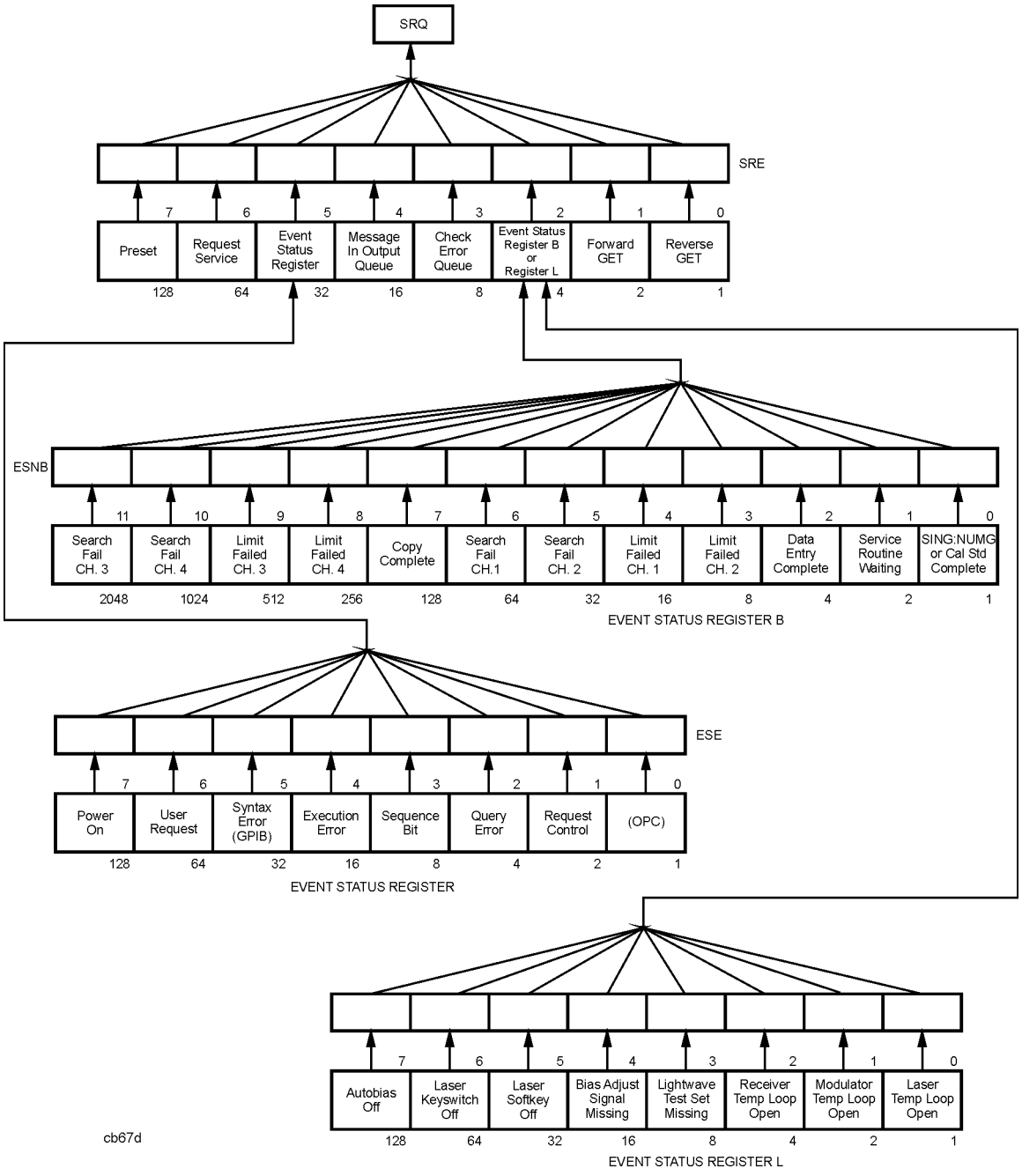


Table 7-1. Status Byte: Status Bit Definitions

Status Byte		
Bit	**Name**	**Definition**
0	Waiting for reverse GET	Not available.
1	Waiting for forward GET	Not available.
2	Check event-status register B	One of the enabled bits in event status register B has been set.
3	Check error queue	An error has occurred and the message has been placed in the error queue, but has not been read yet.
4	Message in output queue	A command has prepared information to be output, but it has not been read yet.
5	Check event-status register	One of the enabled bits in the event-status register has been set.
6	Request service	One of the enabled status-byte bits is causing an SRQ.
7	Preset	An instrument preset has been executed.

Table 7-2. Event Status Register: Status Bit Definitions

Bit	Name	Definition
0	Operation complete	A command for which OPC has been enabled has completed operation.
1	Request control	The analyzer has been commanded to perform an operation that requires control of a peripheral, and needs control of GPIB. Requires pass-control mode.
2	Query error	The analyzer has been addressed to talk but there is nothing in the output queue to transmit.
3	Sequence Bit	A sequence has executed the assert SRQ command.
4	Execution error	A command was received that could not be executed.
5	Syntax error	The incoming GPIB commands contained a syntax error. The syntax error can only be cleared by a device clear or an instrument preset.
6	User request	The operator has pressed a front-panel key or turned the RPG.
7	Power on	A power-on sequence has occurred since the last read of the register.

Table 7-3. Event Status Register B: Status Bit Definitions

Bit	Name	Definition
0	Single sweep, number of groups, or calibration step complete	A single sweep, group, or calibration step has been completed since the last read of the register.
1	Service routine waiting or done	An internal service routine has completed operation, or is waiting for an operator response.
2	Data entry complete	A terminator key has been pressed or a value entered over GPIB since the last read of the register.
3	Limit failed, Channel 2	Limit test failed on Channel 2.
4	Limit failed, Channel 1	Limit test failed on Channel 1.
5	Search failed, Channel 2	A marker search was executed on Channel 2, but the target value was not found.
6	Search failed, Channel 1	A marker search was executed on Channel 1, but the target value was not found.
7	Copy Complete	A copy has been completed since the last read of the register.

Table 7-4. Event Status Register L: Status Bit Definitions

Bit	Name	Definition
0	Laser Temp Loop Open	1 = Laser temperature control is not functioning.
1	Modulator Temp Loop Open	1 = Modulator temperature control is not functioning.
2	Receiver Temp Loop Open	1 = Reciever Temperature control is not functioning.
3	Lightwave Test Set Missing	1 = No lightwave test set installed or cable to test set not connected.
4	Bias Adjust Signal Missing	1 = Laser bias signal missing.
5	Laser Softkey Off	1 = Laser is off via softkey.
6	Laser Keyswitch Off	1 = Laser is off via keyswitch.
7	Autobias Off	1 = Autobias is turned on.

The Status Byte

The analyzer has a status-reporting mechanism that reports information about specific analyzer functions and events. The status byte (consisting of summary bits) is the top-level register. Each bit reflects the condition of another register or queue. If a summary bit is set (equals 1), the corresponding register or queue should be read to obtain the status information and clear the condition. Reading the status byte does not affect the state of the summary bits. The summary bits always reflect the condition of the summarized queue or register.

The status byte can be read by a serial poll or by using the command OUTPSTAT. OUTPSTAT does not automatically put the instrument in remote mode, thus giving the operator access to the analyzer front-panel functions. OUTPSTAT will return an ASCII (text) integer (0–255) that can be interpreted as the 8-bit status byte. Using the OUTPSTAT command will not necessarily return the same status byte value as when using a serial poll because the “Message in Output Queue” bit is always set when using OUTPSTAT.

The status byte:

- summarizes the error queue
- summarizes two event-status registers that monitor specific conditions inside the instrument
- contains a bit that is set when the instrument is issuing a service request (SRQ) over GPIB
- contains a bit that is set when the analyzer has data to transmit over GPIB

Any bit in the status byte can be selectively enabled to generate a service request (SRQ) when set. Setting a bit in the service-request-enable register with the SRE*nn* command enables the corresponding bit in the status byte. The units variable *nn* represents the binary equivalent of the bit in the status byte. For example, SRE24; enables status-byte bits 3 and 4 (since 2^3 + 2^4 = 24) and disables all the other bits. SRE will not affect the state of the status-register bits.

The sequencing bit can be set during the execution of a test sequence to assert an SRQ.

The status byte also summarizes two queues: the output queue and the error queue. (The error queue is described in the next section.) When the analyzer outputs information, it puts the information in the output queue where it resides until the controller reads it. The output queue is only one event long. Therefore, the next output request will clear the current data. The summary bit is set whenever there is data in the output queue.

The Event-Status Register and Event-Status Registers B and L

The event-status register, the event-status register B, and the event-status register L are the other three registers in the status-reporting structure. They are selectively summarized by bits in the status byte via enable registers. The event-status registers consist of latched bits. A latched bit is set at the beginning of a specific trigger condition in the instrument. It can only be cleared by reading the register. The bit will not be reactivated until the condition occurs again. If a bit in one of these three registers is enabled, it is summarized by the summary bit in the status byte. The registers are enabled using the commands ESE*nn*; and ESNB*nn*;, both of which work in the same manner as SRE*nn*. The units variable *nn* represents the binary equivalent of the bit in the status byte. Event-status register L is primarily a hardware status register for information specific to the lightwave portion of the analyzer.

If a bit in one of the event-status registers is enabled, and therefore, the summary bit in the status byte is enabled, an SRQ will be generated. The SRQ will not be cleared until one of the five following conditions transpire:

- The event-status register is read, clearing the latched bit.
- The summary bit in the status byte is disabled.
- The event-status register bit is disabled.
- The status registers are cleared with the CLES; command.
- An instrument preset is performed.

Service requests generated when there are error messages or when the instrument is waiting for the group execute trigger (GET) command are cleared by:

- reading the errors
- issuing GET (disabling the bits)
- clearing the status registers

Error Output

When an error condition is detected in the analyzer, a message is generated, displayed on the analyzer's display screen, and placed in the error queue. Error messages consist of an error number followed by an ASCII string no more than 50 characters long. The string contains the same message that appears on the analyzer's display. The error queue holds up to 20 error messages in the order in which they occur. The error messages remain in the error queue until the errors are read by the system controller using the command OUTPERRO. The OUTPERRO command outputs one error message.

NOTE The error queue can only be cleared by performing an instrument preset or by cycling the line power. In order to keep the queue up-to-date, it is important to read all of the messages out of the queue each time errors are detected.

Error Messages in Numerical Order

Error Number	Error
2	INVALID KEY
4	PHASE LOCK CAL FAILED
5	NO IF FOUND: CHECK R INPUT LEVEL
6	POSSIBLE FALSE LOCK
7	PHASE LOCK FAILURE
8	PHASE LOCK LOST
9	LIST TABLE EMPTY
10	CONTINUOUS SWITCHING NOT ALLOWED
11	SWEEP TIME INCREASED
12	SWEEP TIME TOO FAST
13	AVERAGING INVALID ON NON-RATIO MEASURE
14	FUNCTION NOT VALID
15	NO MARKER DELTA - SPAN NOT SET
17	DEMODULATION NOT VALID
21	POWER SUPPLY HOT!
22	POWER SUPPLY SHUT DOWN!
24	PRINTER: not on, not connect, wrong addrs
25	PRINT ABORTED
26	PLOTTER: not on, not connect, wrong addrs
27	PLOT ABORTED
28	PLOTTER NOT READY-PINCH WHEELS UP
30	REQUESTED DATA NOT CURRENTLY AVAILABLE
31	ADDRESSED TO TALK WITH NOTHING TO SAY
32	WRITE ATTEMPTED WITHOUT SELECTING INPUT TYPE
33	SYNTAX ERROR
34	BLOCK INPUT ERROR
35	BLOCK INPUT LENGTH ERROR

Error Number	Error
36	SYST CTRL OR PASS CTRL IN LOCAL MENU
37	ANOTHER SYSTEM CONTROLLER ON GPIB
38	DISK: not on, not connected, wrong addrs
39	DISK HARDWARE PROBLEM
40	DISK MEDIUM NOT INITIALIZED
41	NO DISK MEDIUM IN DRIVE
42	FIRST CHARACTER MUST BE A LETTER
43	ONLY LETTERS AND NUMBERS ARE ALLOWED
44	NOT ENOUGH SPACE ON DISK FOR STORE
45	NO FILE(S) FOUND ON DISK
46	ILLEGAL UNIT OR VOLUME NUMBER
47	INITIALIZATION FAILED
48	DISK IS WRITE PROTECTED
49	DISK WEAR-REPLACE DISK SOON
50	TOO MANY SEGMENTS OR POINTS
51	INSUFFICIENT MEMORY
54	NO VALID MEMORY TRACE
55	NO VALID STATE IN REGISTER
56	INSTRUMENT STATE MEMORY CLEARED
57	OVERLOAD ON INPUT R, POWER REDUCED
58	OVERLOAD ON INPUT A, POWER REDUCED
59	OVERLOAD ON INPUT B, POWER REDUCED
60	ANALOG INPUT OVERLOAD
61	8703 SOURCE PARAMETERS CHANGED
62	NOT VALID FOR PRESENT TEST SET
63	CALIBRATION REQUIRED
64	CURRENT PARAMETER NOT IN CAL SET
66	CORRECTION TURNED OFF
68	ADDITIONAL STANDARDS NEEDED
69	NO CALIBRATION CURRENTLY IN PROGRESS

Error Number	Error
70	NO SPACE FOR NEW CAL. CLEAR REGISTERS
71	MORE SLIDES NEEDED
72	EXCEEDED 7 STANDARDS PER CLASS
73	SLIDES ABORTED (MEMORY REALLOCATION)
74	CALIBRATION ABORTED
75	FORMAT NOT VALID FOR MEASUREMENT
77	WRONG DISK FORMAT, INITIALIZE DISK
111	DEADLOCK
112	SELF TEST #n FAILED
114	NO FAIL FOUND
116	POWER METER INVALID
117	PWR MTR: NOT ON/CONNECTED OR WRONG ADDRS
118	POWER METER NOT SETTLED
119	DEVICE: not on, not connect, wrong addrs
123	NO MEMORY AVAILABLE FOR INTERPOLATION
124	SELECTED SEQUENCE IS EMPTY
125	DUPLICATING TO THIS SEQUENCE NOT ALLOWED
126	NO MEMORY AVAILABLE FOR SEQUENCING
127	CAN'T STORE/LOAD SEQUENCE, INSUFFICIENT MEMORY
130	D2/D1 INVALID WITH SINGLE CHANNEL
131	FUNCTION NOT VALID DURING MOD SEQUENCE
132	MEMORY FOR CURRENT SEQUENCE IS FULL
133	THIS LIST FREQ INVALID
144	NO LIMIT LINES DISPLAYED
145	SWEEP TYPE CHANGED TO LINEAR SWEEP
150	LOG SWEEP REQUIRES 2 OCTAVE MINIMUM SPAN
151	SAVE FAILED / INSUFFICIENT MEMORY
152	D2/D1 INVALID: CH1 CH2 NUM PTS DIFFERENT
153	SEQUENCE MAY HAVE CHANGED, CAN'T CONTINUE
154	INSUFFICIENT MEMORY, PWR MTR CAL OFF

Error Number	Error
157	SEQUENCE ABORTED
159	CH1 (CH2) TARGET VALUE NOT FOUND
163	FUNCTION ONLY VALID DURING MOD SEQUENCE
164	TOO MANY NESTED SEQUENCES. SEQ ABORTED
165	PARALLEL PORT NOT AVAILABLE FOR GPIO
166	PRINT/PLOT IN PROGRESS, ABORT WITH LOCAL
167	PARALLEL PORT NOT AVAILABLE FOR COPY
168	INSUFFICIENT MEMORY FOR PRINT/PLOT
169	GPIB COPY IN PROGRESS, ABORT WITH LOCAL
170	COPY: device not responding; copy aborted
171	PRINTER: paper error
172	PRINTER: not on line
173	PRINTER: not connected
174	PRINTER: power off
175	PRINTER: error
176	PRINTER: busy
177	PRINTER: not handshaking
178	print color not supported with EPSON
179	POWER OUT MAY BE UNLEVELED
180	DOS NAME LIMITED TO 8 CHARS + 3 CHAR EXTENSION
181	BAD FREQ FOR HARMONIC
183	BATTERY FAILED. STATE MEMORY CLEARED
184	BATTERY LOW! STORE SAVE REGS TO DISK
185	CANNOT FORMAT DOS DISKS ON THIS DRIVE
187	SWEEP MODE CHANGED TO CW TIME SWEEP
188	DIRECTORY FULL
189	DISK READ/WRITE ERROR
190	DISK MESSAGE LENGTH ERROR
192	FILE NOT FOUND
193	ASCII: MISSING 'BEGIN' statement

Error Number	Error
194	ASCII: MISSING 'CITIFILE' statement
195	ASCII: MISSING 'DATA' statement
196	ASCII: MISSING 'VAR' statement
197	FILE NOT FOUND OR WRONG TYPE
198	NOT ALLOWED DURING POWER METER CAL
199	CANNOT MODIFY FACTORY PRESET
200	ALL REGISTERS HAVE BEEN USED
201	FUNCTION NOT VALID FOR INTERNAL MEMORY
202	FUNCTION NOT AVAILABLE
203	CANNOT READ/WRITE HFS FILE SYSTEM
205	LIMIT TABLE EMPTY
206	ARGUMENT OUT OF RANGE
207	POWER OUT MAY BE UNLEVELED
208	EXT R CHAN MUST BE ON FOR FREQUENCY OFFSET MODE
209	SWEEP MUST BE STEPPED FOR FREQUENCY OFFSET MODE?
211	OVERLAP!LIST TYPE CHANGED TO STEPPED
212	ANALOG BUS DISABLED IN 6 KHZ IF BW
213	RANGE CAUSED POWER LVL CHANGE IN LIST
214	CORRECTION ON: AUX CHANNEL(S) RESTORED
215	CAUTION: CORRECTION OFF: AUX CHANNEL(S) DISABLED
218	CAUTION: FLOPPY DISK IS FULL
219	LASER TEMPERATURE LOOP OPEN
220	MODULATOR TEMPERATURE LOOP OPEN
221	RECEIVER TEMPERATURE LOOP OPEN
223	ISOL AVGS < SWP AVGS

8

Programming Examples

Example Programs

This chapter provides explanations and listings for the example programs that are included on the CD-ROM (part number 08703-10202) that is shipped with the analyzer. Refer to "Measurement Process" on page 8-3 for a description of the typical measurement process.

All of the examples on the CD-ROM are provided in BASIC. See Chapter 1, "Introduction to Instrument Control" for information on using BASIC, and installing and using the VXI*plug&play* driver.

NOTE "Example 1A: Setting Parameters for Electrical Devices" on page 8-9 includes program listings for BASIC, Visual C++, and Visual BASIC. These listings are provided for you to see a comparison of the different programming languages. The rest of the examples in this manual only include the BASIC listing. You can readily view *all* of the programs by accessing the CD-ROM that was shipped with this manual.

Measurement Process

This section explains how to organize instrument commands into a measurement sequence. A typical measurement sequence consists of the following steps:

1. setting up the instrument
2. calibrating the test setup
3. connecting the device under test
4. taking the measurement data
5. post-processing the measurement data
6. transferring the measurement data

Step 1. Setting Up the Instrument

Define the measurement by setting all of the basic measurement parameters. These include:

- the sweep type
- the frequency span
- the sweep time
- the number of points (in the data trace)
- the RF power level
- the type of measurement
- the IF averaging
- the IF bandwidth

You can quickly set up an entire instrument state, using the save/recall registers and the learn string. The learn string is a summary of the instrument state compacted into a string that the computer reads and retransmits to the analyzer. See “Example 5A: Using the Learn String” on page 8-84.

Step 2. Calibrating the Test Setup

After you have defined an instrument state, you should perform a measurement calibration. Although it is not required, a measurement calibration improves the accuracy of your measurement data.

The following list describes several methods to calibrate the analyzer:

- Stop the program and perform a calibration from the analyzer's front panel.
- Use the computer to guide you through the calibration, as discussed in “Measurement Calibration Examples” on page 8-26.
- Transfer the calibration data from a previous calibration back into the analyzer, as discussed in “Example 5C: Saving and Restoring the Analyzer Instrument State” on page 8-88.

Step 3. Connecting the Device under Test

After you connect your test device, you can use the computer to speed up any necessary device adjustments such as limit testing, bandwidth searches, and trace statistics.

Step 4. Taking the Measurement Data

Measure the device response and set the analyzer to hold the data. This captures the data on the analyzer display.

By using the single-sweep command (SING), you can ensure a valid sweep. When you use this command, the analyzer completes all stimulus changes before starting the sweep, and does not release the GPIB hold state until it has displayed the formatted trace. Then when the analyzer completes the sweep, the instrument is put into hold mode, freezing the data. Because single sweep is OPC-compatible, it is easy to determine when the sweep has been completed.

The number-of-groups command (NUMGn) triggers multiple sweeps. It is designed to work the same as single-sweep command. NUMGn is useful for making a measurement with an averaging factor *n* (*n* can be 1 to 999). Both the single-sweep and number-of-groups commands restart averaging.

Step 5. Post-Processing the Measurement Data

Figure 6-1 on page 6-3 shows the process functions used to affect the data after you have made an error-corrected measurement. These process functions have parameters that can be adjusted to manipulate the error-corrected data prior to formatting. They do not affect the analyzer's data gathering. The most useful functions are trace statistics, marker searches, and electrical-delay offset.

After performing and activating a full 2-port measurement calibration, any of the four S-parameters may be viewed without taking a new sweep.

Step 6. Transferring the Measurement Data

Read your measurement results. All the data-output commands are designed to ensure that the data transmitted reflects the current state of the instrument.

Programming Examples

The following example programs provide you with factory-tested solutions for several remotely-controlled analyzer processes. The programs can be used in their present state or modified to suit specific needs. The programs discussed in this section can be found on the "Programming Examples" CD-ROM that was shipped with this manual.

Table 8-1. Measurement Setup Example Programs

Example Number	Description	File Name(s) on CD-ROM
1A	Setting Parameters for Electrical Devices	EXAMP1A EXAMP1A.CPP EXAMP1A.FRM
1B	Setting Parameters on Two Channels for Optical Devices	LEXAMP1B
1C	Setting Parameters on Four Channels for Optical Devices	LEXAMP1C
1D	Verifying Parameters	EXAMP1D

Table 8-2. Measurement Calibration Example Programs

Example Number	Description	File Name(s) on CD-ROM
2A	Response Calibration for E/E Devices	EXAMP2A
2B	Response Calibration for O/O, E/O, O/E Devices	LEXAMP2B
2C	1-Port Measurement Calibration for E/E Devices	EXAMP2C
2D	Enhanced Response Calibration for E/E Devices	EXAMP2D
2E	Full 2-Port Measurement Calibration for E/E Devices	EXAMP2E
2F	Response and Match Calibration for O/E Devices	LEXAMP2F
2G	Adapter Removal Calibration for E/E Devices	EXAMP2G
2H	Using Raw Data to Create a Calibration (Simmcal) for E/E Devices	EXAMP2H
2I	Response and Match Calibration for E/O Devices	LEXAMP2I
2J	Take4 — Error Correction Processed on an External PC	EXAMP2J

Table 8-3. Measurement Data Transfer Example Programs

Example Number	Description	File Name(s) on CD-ROM
3A	Data Transfer Using Markers	EXAMP3A
3B	Data Transfer Using FORM 4 (ASCII Transfer)	EXAMP3B
3C	Data Transfer Using Floating-Point Numbers	EXAMP3C

Programming Examples

Table 8-3. Measurement Data Transfer Example Programs

Example Number	Description	File Name(s) on CD-ROM
3D	Data Transfer Using Frequency–Array Information	EXAMP3D
3E	Data Transfer Using FORM 1 (Internal Binary Format)	EXAMP3E

Table 8-4. Measurement Process Synchronization Example Programs

Example Number	Description	File Name(s) on CD-ROM
4A	Using the Error Queue	EXAMP4A
4B	Generating Interrupts	EXAMP4B
4C	Power Meter Calibration[a]	EXAMP4C

a. This example program will not work with BASIC for Windows.

Table 8-5. Measurement Process Synchronization Example Programs

Example Number	Description	File Name(s) on CD-ROM
5A	Using the Learn String	EXAMP5A
5B	Reading Calibration Data	EXAMP5B
5C	Saving and Restoring the Analyzer Instrument State	EXAMP5C

Table 8-6. Limit-Line Testing Example Programs

Example Number	Description	File Name(s) on CD-ROM
6A	Setting Up a List-Frequency Table in Stepped List Mode	EXAMP6A
6B	Setting Up a List-Frequency Table in Swept List Mode	EXAMP6B
6C	Selecting a Single Segment from a Table of Segments	EXAMP6C
6D	Setting Up a Limit Test Table	EXAMP6D
6E	Performing PASS/FAIL Tests while Tuning	EXAMP6E

Table 8-7. Report Generation Example Programs

Example Number	Description	File Name(s) on CD-ROM
7A	Operation Using Talker/Listener Mode	EXAMP7A
7B	Controlling Peripherals Using Pass-Control Mode[a]	EXAMP7B
7C	Printing with the Parallel Port	EXAMP7C
7D	Plotting to a File and Transferring the File Data to a Plotter	EXAMP7D
7E	Reading Plot Files From a Disk[a]	EXAMP7E
7F	Reading ASCII Disk Files to the Instrument Controller's Disk File	EXAMP7F

a. This example program will not work with BASIC for Windows.

Program Information

The following information is provided for every BASIC example program included on the CD-ROM that is shipped with the analyzer:

- A program description
- An outline of the program's processing sequence
- A step-by-step instrument-command-level tutorial explanation of the program including:

  The command mnemonic and command name for the GPIB instrument command used in the program.

  An explanation of the operations and affects of the GPIB instrument commands used in the program.

Analyzer Features Helpful in Developing Programming Routines

Analyzer-Debug Mode

The analyzer-debug mode aids you in developing programming routines. The analyzer displays the commands being received. If a syntax error occurs, the analyzer displays the last buffer and points to the first character in the command line that it could not understand.

You can enable this mode from the front panel by pressing **Local, GPIB DIAG ON.** The debug mode remains activated until you preset the analyzer or deactivate the mode. You can also enable this mode over the GPIB using the `DEBUON;` command and disable the debug mode using the `DEBUOFF;` command.

User-Controllable Sweep

There are three important advantages to using the single-sweep mode:

1. The user can initiate the sweep.
2. The user can determine when the sweep has completed.
3. The user can be confident that the trace data has be derived from a valid sweep.

Execute the command string `OPC?;SING;` to place the analyzer in single-sweep mode and trigger a sweep. Once the sweep is complete, the analyzer returns an ASCII character one (1) to indicate the completion of the sweep.

NOTE The measurement cycle and the data acquisition cycle must always be synchronized. The analyzer must complete a measurement sweep for the data to be valid.

Measurement Setup Examples

The programs included in this section provide the option to perform instrument-setup functions for the analyzer from a remote controller. Examples 1A, 1B, and 1C are programs designed to set up the analyzer's measurement parameters. Example 1D is a program designed to verify the measurement parameters.

Example 1A: Setting Parameters for Electrical Devices

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

This example sets the following parameters:

- data display formats
- number of channels and graticules displayed
- frequency range

The following is an outline of the program's processing sequence:

- An I/O path is assigned for the analyzer.
- The system is initialized.
- The analyzer is adjusted to measure return loss (S_{11}) on channel 1 and display as log magnitude.
- The analyzer is adjusted to measure return loss (S_{11}) on channel 2 and display the phase.
- The dual-channel display mode is activated.
- The system operator is prompted to enter the frequency range of the measurement.
- The displays are autoscaled.
- The system operator is prompted to press “Enter” on the computer keyboard to view four types of measurements simultaneously.
- Channels 3 and 4 are turned on.
- Channel 2 is adjusted to measure transmission (S_{21}) displayed in log magnitude, and then the display is autoscaled.
- Channel 3 is adjusted to measure reflected power (A) displayed in log magnitude, and then the display is autoscaled.
- Channel 4 is adjusted to measure transmitted power (B) displayed in log magnitude, and then the display is autoscaled.
- The four channels are each displayed in a separate graticule.
- The analyzer is released from remote control and the program ends.

BASIC Program Listing

```
10  ! This program demonstrates setup of various measurement parameters such
20  ! as start frequency, stop frequency, etc.  The program first selects one
30  ! type of measurement to be viewed using dual-channel display format.
40  ! The specified start and stop frequencies are then programmed and the
50  ! analyzer display is autoscaled.  The program concludes by displaying
60  ! four types of measurements simultaneously.
70  !
80  ! EXAMP1A Setting Parameters for Electrical Devices
90  !
100   ASSIGN @Nwa TO 716                ! Assign an I/O path for the analyzer
110 !
120   CLEAR SCREEN
130 ! Initialize the system
140   ABORT 7                           ! Generate an IFC (Interface Clear)
150   CLEAR @Nwa                        ! SDC (Selected Device Clear) analyzer
160   OUTPUT @Nwa;"OPC?;PRES;"          ! Preset the analyzer and wait
170   ENTER @Nwa;Reply                  ! Read in the 1 returned
180 !
190 ! Set up measurement and display
200   OUTPUT @Nwa;"CHAN1;"              ! Channel 1
210   OUTPUT @Nwa;"AUXCOFF;"            ! Turn off auxiliary channel, if it is on
220   OUTPUT @Nwa;"S11;"                ! Return Loss (Reflection) measurement
230   OUTPUT @Nwa;"LOGM;"               ! Log magnitude display
240 !
250   OUTPUT @Nwa;"CHAN2;"              ! Channel 2
260   OUTPUT @Nwa;"AUXCOFF;"            ! Turn off auxiliary channel, if it is on
270   OUTPUT @Nwa;"S11;"                ! Return Loss (Reflection) measurement
280   OUTPUT @Nwa;"PHAS;"               ! Phase display
290 !
300   OUTPUT @Nwa;"DUACON;"             ! Dual channel display
310 !
320 ! Request start and stop frequency
330   INPUT "ENTER START FREQUENCY (MHz):",F_start
340   INPUT "ENTER STOP FREQUENCY (MHz):",F_stop
350 !
360 ! Program the analyzer settings
370   OUTPUT @Nwa;"STAR";F_start;"MHZ;"  ! Set the start frequency
380   OUTPUT @Nwa;"STOP";F_stop;"MHZ;"   ! Set the stop frequency
390 !
400 ! Autoscale the displays
410   OUTPUT @Nwa;"CHAN1;AUTO;"         ! Autoscale channel 1 display
420   OUTPUT @Nwa;"CHAN2;AUTO;"         ! Autoscale channel 2 display
430 !
440   PRINT "The display should now be autoscaled."
450   INPUT "Press RETURN to view four types of measurements simultaneously",X
460 !
470   OUTPUT @Nwa;"CHAN1;AUXCON;"       ! Turn on auxiliary channel (Channel 3)
480   OUTPUT @Nwa;"CHAN2;AUXCON;"       ! Turn on auxiliary channel (Channel 4)
490 !
500 ! Channel 2 Insertion Loss (Transmission) measurement
510   OUTPUT @Nwa;"S21;"
520   OUTPUT @Nwa;"LOGM;AUTO;"          ! Channel 2 log magnitude and autoscale
530 !
540 ! Channel 3 Reflected Power measurement
550   OUTPUT @Nwa;"CHAN3;MEASA;"
560   OUTPUT @Nwa;"LOGM;AUTO;"          ! Channel 3 log magnitude and autoscale
```

```
570 !
580 ! Channel 4 Transmitted Power measurement
590   OUTPUT @Nwa;"CHAN4;MEASB;"
600   OUTPUT @Nwa;"LOGM;AUTO;"          ! Channel 4 log magnitude and autoscale
610 !
620   OUTPUT @Nwa;"SPLID4;"             ! Display as four separate graticules
630 !
640   OUTPUT @Nwa;"OPC?;WAIT;"          ! Wait for the analyzer to finish
650   ENTER @Nwa;Reply                  ! Read the 1 when complete
660   LOCAL @Nwa                        ! Release GPIB control
670   END
```

Visual C++ Program Listing

```
#include <stdlib.h>
#include <stdio.h>
#include <conio.h>

#include "visa.h"
#include "875x_cpp.h"

ViStatus initialize(ViRsrc Nwa, ViBoolean id_query, ViBoolean do_reset, ViPSession
vi_ptr);
ViStatus checkErr(ViSession vi, ViStatus err_status);

/*****************************************************************************/
/* 875x Instrument Driver EXAMPLE #1A                                        */
/*                                                                           */
/* This program demonstrates setup of various measurement parameters such    */
/* as start frequency, stop frequency, etc.  The program first selects one   */
/* type of measurement to be viewed using dual-channel display format.       */
/* The specified start and stop frequencies are then programmed and the      */
/* analyzer display is autoscaled.  The program concludes by displaying      */
/* four types of measurements simultaneously.                                */
/*****************************************************************************/
int main ()
{
ViSessionvi;
ViStatuserr_status;
ViRsrc      nwa;
ViReal64    f_start;
ViReal64    f_stop;
ViBoolean   reply;

printf("Example 1a --\n");
printf("This program demonstrates setup of various measurement parameters such\n");
printf("as start frequency, stop frequency, etc.  The program first selects one\n");
printf("type of measurement to be viewed using dual-channel display format.\n");
printf("The specified start and stop frequencies are then programmed and the\n");
printf("analyzer display is autoscaled.  The program concludes by displaying\n");
printf("four types of measurements simultaneously.\n\n");

nwa = "GPIB0::16::INSTR";

// Initialize the instrument
initialize(nwa, VI_FALSE, VI_TRUE, &vi);
```

```
// Set the timeout to 3000 msec (3 sec)
err_status = 875x_timeOut(vi, 3000);
checkErr(vi, err_status);

// Set up measurement and display

// Channel 1
err_status = 875x_channelSelect(vi, 875x_CH1);
checkErr(vi, err_status);
// Turn auxiliary channel off.
err_status = 875x_auxChannel(vi, 875x_CH1, 875x_OFF);
checkErr(vi, err_status);
// Return Loss measurement, no parameter conversion
// 875x_S_PAR11 is interpreted as 875x_REFL
err_status = 875x_measType(vi, 875x_S_PAR11, 875x_CONV_OFF);
checkErr(vi, err_status);
// Log magnitude display
err_status =  875x_displaySelect(vi,875x_CH1,875x_DISP_DATA,875x_DISP_LOGM);
checkErr(vi, err_status);
// Channel 2
err_status = 875x_channelSelect(vi, 875x_CH2);
checkErr(vi, err_status);
// Turn auxiliary channel off.
err_status = 875x_auxChannel(vi, 875x_CH2, 875x_OFF);
checkErr(vi, err_status);
// Return Loss measurement, no parameter conversion
err_status = 875x_measType(vi, 875x_S_PAR11, 875x_CONV_OFF);
checkErr(vi, err_status);
// Phase display
err_status =  875x_displaySelect(vi,875x_CH2,875x_DISP_DATA,875x_DISP_PHAS);
checkErr(vi, err_status);
// Dual channel display (single graticule)
err_status = 875x_dualSplit(vi, 875x_DUAL_CHAN_ON, 875x_SPLIT_CHAN_OFF);
checkErr(vi, err_status);
// Request start and stop frequency
printf("Enter Start Frequency (MHz)\n");
scanf("%lf", &f_start);
f_start *= 1000000;
printf("Enter Stop Frequency (MHz)\n");
scanf("%lf", &f_stop);
f_stop *= 1000000;
// Program the frequency settings
err_status = 875x_frequency(vi, 875x_FREQ_STRT_STOP, f_start, f_stop);
checkErr(vi, err_status);
// Autoscale the displays
err_status = 875x_channelSelect(vi, 875x_CH1);
checkErr(vi, err_status);
err_status = 875x_autoscale(vi);
checkErr(vi, err_status);
err_status = 875x_channelSelect(vi, 875x_CH2);
checkErr(vi, err_status);
err_status = 875x_autoscale(vi);
checkErr(vi, err_status);
printf("The display should now be autoscaled.\n");
printf("Press any key on the computer keyboard to view four types of measurements\n");
printf("simultaneously on the analyzer display.\n");
// Wait for keyboard input, then remove character from buffer
while (!_kbhit());
```

```
while (_kbhit()) getch();
// Channel 1
err_status = 875x_channelSelect(vi, 875x_CH1);
checkErr(vi, err_status);
// Turn auxiliary channel on.
err_status = 875x_auxChannel(vi, 875x_CH1, 875x_ON);
checkErr(vi, err_status);
// Channel 2
err_status = 875x_channelSelect(vi, 875x_CH2);
checkErr(vi, err_status);
// Turn auxiliary channel on.
err_status = 875x_auxChannel(vi, 875x_CH2, 875x_ON);
checkErr(vi, err_status);
// Channel 2 Insertion Loss measurement, no parameter conversion
// 875x_S_PAR21 is interpreted as 875x_TRANS
err_status = 875x_measType(vi, 875x_S_PAR21, 875x_CONV_OFF);
checkErr(vi, err_status);
// Log magnitude and autoscale
err_status =  875x_displaySelect(vi,875x_CH2,875x_DISP_DATA,875x_DISP_LOGM);
checkErr(vi, err_status);
err_status = 875x_autoscale(vi);
checkErr(vi, err_status);
// Channel 3 Reflected Power measurement, no parameter conversion
err_status = 875x_channelSelect(vi, 875x_CH3);
checkErr(vi, err_status);
err_status = 875x_measType(vi, 875x_IN_MA, 875x_CONV_OFF);
checkErr(vi, err_status);
// Log magnitude and autoscale
err_status =  875x_displaySelect(vi,875x_CH3,875x_DISP_DATA,875x_DISP_LOGM);
checkErr(vi, err_status);
err_status = 875x_autoscale(vi);
checkErr(vi, err_status);
// Channel 4 Transmitted Power measurement, no parameter conversion
err_status = 875x_channelSelect(vi, 875x_CH4);
checkErr(vi, err_status);
err_status = 875x_measType(vi, 875x_IN_MB, 875x_CONV_OFF);
checkErr(vi, err_status);
// Log magnitude and autoscale
err_status =  875x_displaySelect(vi,875x_CH4,875x_DISP_DATA,875x_DISP_LOGM);
checkErr(vi, err_status);
err_status = 875x_autoscale(vi);
checkErr(vi, err_status);
// Display all 4 measurements, each in a separate graticule, Channel 3 in upper right
err_status = 875x_dualSplit4Parm(vi, 875x_DUAL_CHAN_ON, 875x_DISP_4_GRAT,
875x_DISP_2_CHAN3_TOP, 875x_DISP_4_CHAN3_UPR);
checkErr(vi, err_status);
// Wait for analyzer to finish
err_status = 875x_opc_Q(vi, “WAIT”, &reply);
checkErr(vi, err_status);
// Close the session
err_status = 875x_close(vi);
checkErr(vi, err_status);
printf(“Program completed\n”);
return(0);
}

ViStatus initialize(ViRsrc Nwa, ViBoolean id_query, ViBoolean do_reset, ViPSession
vi_ptr)
```

```
{
ViSession   vi;
ViStatuserr_status;
ViChar      err_message[256];

/*  Note that this function can verify that the instrument specified is an  */
/*  875x (id_query=VI_TRUE) and can send a reset to the instrument        */
/*  (do_reset=VI_TRUE).                                                    */

 err_status = 875x_init(Nwa, id_query, do_reset, &vi);

 if (( err_status < VI_SUCCESS ) || ( vi == VI_NULL ))
 {
    printf("\ninit failed with return code %d\n", err_status);
    if ( vi != VI_NULL )
    {
        875x_error_message(vi,err_status,err_message);
        printf("  Error Status:  %d\n", err_status);
        printf("  Error Message: %s\n", err_message);
    }
   exit (err_status);
 }

 *vi_ptr = vi;

 return(VI_SUCCESS);
}

ViStatus checkErr (ViSession vi, ViStatus err_status)
{
    ViInt32     inst_err;
    ViChar      err_message[256];

    if(VI_SUCCESS > err_status)
    {
            /* Send a device clear to ensure communication with */
            /* the instrument.                            */

        875x_dcl(vi);

            /*   If the driver is set to detect instrument errors, and */
            /*   an instrument error is detected, the error code is    */
            /*   875x_INSTR_ERROR_DETECTED (see 875x_cpp.h). In this */
    /*   case, query the instrument for the error and display  */
            /*   it. Otherwise, the error is a driver error. Query the */
            /*   driver for the error and display it.                  */

        if(875x_INSTR_ERROR_DETECTED == err_status)

        {
            875x_error_query(vi, &inst_err, err_message);
            printf("Instrument Error : %ld, %s\n", inst_err, err_message);


        }
        else
        {
            875x_error_message(vi, err_status, err_message);
```

```
            printf("Driver Error : %ld, %s\n", err_status, err_message);
        }

            /* Optionally reset the instrument, close the */
            /* instrument handle, and exit the program.   */

    /* 875x_reset(vi); */
    /* 875x_close(vi); */
    /* exit(err_status);   */
    return VI_TRUE;
    }

return VI_SUCCESS ;
}
```

Visual BASIC Program Listing

```
VERSION 5.00
Begin VB.Form frmExample1a
   Caption         =   "875x Visual Basic Programming Example 1a"
   ClientHeight    =   5610
   ClientLeft      =   1140
   ClientTop       =   1515
   ClientWidth     =   5895
   LinkTopic       =   "Form1"
   PaletteMode     =   1  'UseZOrder
   ScaleHeight     =   5610
   ScaleWidth      =   5895
   Begin VB.ListBox lstText
      Height          =   4350
      ItemData        =   "frmExamp1a.frx":0000
      Left            =   360
      List            =   "frmExamp1a.frx":0002
      TabIndex        =   2
      Top             =   120
      Width           =   5175
   End
   Begin VB.CommandButton cmdQuit
      Caption         =   "Quit"
      Height          =   495
      Left            =   3120
      TabIndex        =   1
      Top             =   4800
      Width           =   1455
   End
   Begin VB.CommandButton cmdExecute
      Caption         =   "Execute Program"
      Height          =   495
      Left            =   1200
      TabIndex        =   0
      Top             =   4800
      Width           =   1455
   End
End
Attribute VB_Name = "frmExample1a"
Attribute VB_GlobalNameSpace = False
Attribute VB_Creatable = False
Attribute VB_PredeclaredId = True
```

```
Attribute VB_Exposed = False
'*************************************************************************
'* 875x Instrument Driver EXAMPLE #1A                                    *
'*                                                                       *
'* This program demonstrates setup of various measurement parameters such  *
'* as start frequency, stop frequency, etc.  The program first selects one *
'* type of measurement to be viewed using dual-channel display format.     *
'* The specified start and stop frequencies are then programmed and the    *
'* analyzer display is autoscaled.  The program concludes by displaying    *
'* four types of measurements simultaneously.                            *
'*************************************************************************

Private Sub cmdExecute_Click()

Dim vi         As Long
Dim err_status As Long
Dim nwa        As String
Dim retString  As String
Dim msg        As String
Dim f_start    As Double
Dim f_stop     As Double
Dim iRetVal    As Integer
Dim reply      As Integer

lstText.Clear
lstText.AddItem "Example 1a --"
lstText.AddItem "This program demonstrates setup of various measurement parameters
such"
lstText.AddItem "as start frequency, stop frequency, etc.  The program first selects
one"
lstText.AddItem "type of measurement to be viewed using dual-channel display format."
lstText.AddItem "The specified start and stop frequencies are then programmed and the"
lstText.AddItem "analyzer display is autoscaled.  The program concludes by displaying"
lstText.AddItem "four types of measurements simultaneously."
lstText.AddItem ""

nwa = "GPIB0::16::INSTR"

' Initialize the instrument
Call initialize(nwa, VI_FALSE, VI_TRUE, vi)

' Set the timeout to 3000 msec (3 sec)
err_status = 875x_timeOut(vi, 3000)
Call checkErr(vi, err_status)

' Set up measurement and display

' Channel 1
err_status = 875x_channelSelect(vi, 875x_CH1)
Call checkErr(vi, err_status)
' Turn auxiliary channel off.
err_status = 875x_auxChannel(vi, 875x_CH1, 875x_OFF)
Call checkErr(vi, err_status)
' Return Loss measurement, no parameter conversion
' 875x_S_PAR11 is interpreted as 875x_REFL
err_status = 875x_measType(vi, 875x_S_PAR11, 875x_CONV_OFF)
Call checkErr(vi, err_status)
' Log magnitude display
```

```
err_status = 875x_displaySelect(vi, 875x_CH1, 875x_DISP_DATA, 875x_DISP_LOGM)
Call checkErr(vi, err_status)
‘ Channel 2
err_status = 875x_channelSelect(vi, 875x_CH2)
Call checkErr(vi, err_status)
‘ Turn auxiliary channel off.
err_status = 875x_auxChannel(vi, 875x_CH2, 875x_OFF)
Call checkErr(vi, err_status)
‘ Return Loss measurement, no parameter conversion
err_status = 875x_measType(vi, 875x_S_PAR11, 875x_CONV_OFF)
Call checkErr(vi, err_status)
‘ Phase display
err_status = 875x_displaySelect(vi, 875x_CH2, 875x_DISP_DATA, 875x_DISP_PHAS)
Call checkErr(vi, err_status)
‘ Dual channel display (single graticule)
err_status = 875x_dualSplit(vi, 875x_DUAL_CHAN_ON, 875x_SPLIT_CHAN_OFF)
Call checkErr(vi, err_status)
‘ Request start and stop frequency
retString = InputBox$(“Enter Start Frequency (MHz)”, frmExample1a.Caption)
If retString = “” Then Exit Sub
f_start = Val(retString) * 1000000# ‘ Convert to real value in Hz
retString = InputBox$(“Enter Stop Frequency (MHz)”, frmExample1a.Caption)
If retString = “” Then Exit Sub
f_stop = Val(retString) * 1000000# ‘ Convert to real value in Hz
‘ Program the frequency settings
err_status = 875x_frequency(vi, 875x_FREQ_STRT_STOP, f_start, f_stop)
Call checkErr(vi, err_status)
‘ Autoscale the displays
err_status = 875x_channelSelect(vi, 875x_CH1)
Call checkErr(vi, err_status)
err_status = 875x_autoscale(vi)
Call checkErr(vi, err_status)
err_status = 875x_channelSelect(vi, 875x_CH2)
Call checkErr(vi, err_status)
err_status = 875x_autoscale(vi)
Call checkErr(vi, err_status)
msg = “The display should now be autoscaled.” & Chr$(13) & Chr$(10)
msg = msg & “Click OK to view four types of measurements “
msg = msg & “simultaneously on the analyzer display.”
iRetVal = MsgBox(msg, vbOKCancel, frmExample1a.Caption)
If iRetVal = vbCancel Then Exit Sub
‘ Channel 1
err_status = 875x_channelSelect(vi, 875x_CH1)
Call checkErr(vi, err_status)
‘ Turn auxiliary channel on.
err_status = 875x_auxChannel(vi, 875x_CH1, 875x_ON)
Call checkErr(vi, err_status)
‘ Channel 2
err_status = 875x_channelSelect(vi, 875x_CH2)
Call checkErr(vi, err_status)
‘ Turn auxiliary channel on.
err_status = 875x_auxChannel(vi, 875x_CH2, 875x_ON)
Call checkErr(vi, err_status)
‘ Channel 2 Insertion Loss measurement, no parameter conversion
‘ 875x_S_PAR21 is interpreted as 875x_TRANS
err_status = 875x_measType(vi, 875x_S_PAR21, 875x_CONV_OFF)
Call checkErr(vi, err_status)
‘ Log magnitude and autoscale
```

```
err_status = 875x_displaySelect(vi, 875x_CH2, 875x_DISP_DATA, 875x_DISP_LOGM)
Call checkErr(vi, err_status)
err_status = 875x_autoscale(vi)
Call checkErr(vi, err_status)
‘ Channel 3 Reflected Power measurement, no parameter conversion
err_status = 875x_channelSelect(vi, 875x_CH3)
Call checkErr(vi, err_status)
err_status = 875x_measType(vi, 875x_IN_MA, 875x_CONV_OFF)
Call checkErr(vi, err_status)
‘ Log magnitude and autoscale
err_status = 875x_displaySelect(vi, 875x_CH3, 875x_DISP_DATA, 875x_DISP_LOGM)
Call checkErr(vi, err_status)
err_status = 875x_autoscale(vi)
Call checkErr(vi, err_status)
‘ Channel 4 Transmitted Power measurement, no parameter conversion
err_status = 875x_channelSelect(vi, 875x_CH4)
Call checkErr(vi, err_status)
err_status = 875x_measType(vi, 875x_IN_MB, 875x_CONV_OFF)
Call checkErr(vi, err_status)
‘ Log magnitude and autoscale
err_status = 875x_displaySelect(vi, 875x_CH4, 875x_DISP_DATA, 875x_DISP_LOGM)
Call checkErr(vi, err_status)
err_status = 875x_autoscale(vi)
Call checkErr(vi, err_status)
‘ Display all 4 measurements, each in a separate graticule, Channel 3 in upper right
err_status = 875x_dualSplit4Parm(vi, 875x_DUAL_CHAN_ON, 875x_DISP_4_GRAT,
875x_DISP_2_CHAN3_TOP, 875x_DISP_4_CHAN3_UPR)
Call checkErr(vi, err_status)
‘ Wait for analyzer to finish
err_status = 875x_opc_Q(vi, “WAIT”, reply)
Call checkErr(vi, err_status)
‘ Close the session
err_status = 875x_close(vi)
Call checkErr(vi, err_status)
‘ Display “Example Completed” message box
iRetVal = MsgBox(“The example has completed”, vbOKOnly, frmExample1a.Caption)
End Sub

Private Sub cmdQuit_Click()
    ‘ Close the application
    End
End Sub

Public Sub initialize(ByVal nwa As String, ByVal id_query As Integer, ByVal do_reset
As Integer, vi As Long)

Dim err_status As Long
Dim err_msg    As String * 256

‘ Note that this function will verify that the instrument
‘ specified is an 875x (id_query=VI_TRUE) and will send
‘ a reset to the instrument (do_reset=VI_TRUE).

err_status = 875x_init(nwa, id_query, do_reset, vi)

If ((err_status < VI_SUCCESS) Or (vi = VI_NULL)) Then

        msg = “init failed with return code “ & err_status
```

```
        If (vi <> VI_NULL) Then
                err_status = 875x_error_message(vi, err_status, err_msg)
                msg = msg & “,  Error Status: “ & err_status

                msg = msg & “,  Error Message: “ & err_msg
        End If
        MsgBox msg, vbInformation, frmExample1a.Caption
        End
End If
End Sub

Sub checkErr(ByVal vi As Long, ByVal err_status As Long)

Dim inst_err            As Long
Dim err_message         As String * 250
Dim retStatus           As Long

Dim nl
nl = Chr(10)

If VI_SUCCESS > err_status Then

    ‘Send a device clear to ensure communication with ‘the instrument.

    retStatus = 875x_dcl(vi)

    If (875x_INSTR_ERROR_DETECTED = err_status) Then

        ‘query the instrument for the error
        retStatus = 875x_error_query(vi, inst_err, err_message)

        msg = “CHECK :Instrument Error :” & inst_err & nl & “Error Message = “ &
err_message

        MsgBox msg, vbOKOnly, frmExample1a.Caption
    Else
        ‘get the driver error message
        retStatus = 875x_error_message(vi, err_status, err_message)
        msg = “CHECK :Driver Error :” & errStatus & nl & “Error Message = “ &
err_message

        MsgBox msg, vbInformation, frmExample1a.Caption
    End If

End If
‘ optionally reset the instrument, close the instrument handle

‘retStatus=875x_reset(vi)
‘retStatus=875x_close(vi)

End Sub
```

Example 1B: Setting Parameters on Two Channels for Optical Devices

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

This example shows how to setup various measurement parameters.

This example sets the following parameters:

- data display formats
- number of channels and graticules displayed
- frequency range

The following is an outline of the program's processing sequence:

- An I/O path is assigned for the analyzer.
- The system is initialized.
- The analyzer is adjusted to measure O/O transmission on channel 1 and display as log magnitude.
- The analyzer is adjusted to measure O/O transmission on channel 2 and display the phase.
- The dual-channel display mode is activated.
- The system operator is prompted to enter the frequency range of the measurement.
- The displays are autoscaled.
- The analyzer is released from remote control and the program ends.

BASIC Program Listing

```
10  ! This program demonstrates setup of various measurement parameters such
20  ! as start frequency, stop frequency, etc.  The program first selects one
30  ! type of measurement to be viewed using dual-channel display format.
40  ! The specified start and stop frequencies are then programmed and the
50  ! analyzer display is autoscaled.
60  ! .
70  !
80  ! LEXAMP1B Setting Parameters for Optical Devices
90  !
100   ASSIGN @Nwa TO 716                ! Assign an I/O path for the analyzer
110 !
120   CLEAR SCREEN
130 ! Initialize the system
140   ABORT 7                           ! Generate an IFC (Interface Clear)
150   CLEAR @Nwa                        ! SDC (Selected Device Clear) analyzer
160   OUTPUT @Nwa;"OPC?;PRES;"          ! Preset the analyzer and wait
```

```
170   ENTER @Nwa;Reply                  ! Read in the 1 returned
180 !
190 ! Set up measurement and display
200   OUTPUT @Nwa;"CHAN1;"              ! Channel 1
210   OUTPUT @Nwa;"AUXCOFF;"            ! Turn off auxiliary channel, if it is on
220   OUTPUT @Nwa;"MEASOO1;"            ! O/O measurement
230   OUTPUT @Nwa;"LOGM;"               ! Log magnitude display
240 !
250   OUTPUT @Nwa;"CHAN2;"              ! Channel 2
260   OUTPUT @Nwa;"AUXCOFF;"            ! Turn off auxiliary channel, if it is on
270   OUTPUT @Nwa;"MEASOO1;"            ! O/O measurement
280   OUTPUT @Nwa;"PHAS;"               ! Phase display
290 !
300   OUTPUT @Nwa;"DUACON;"             ! Dual channel display
310 !
320 ! Request start and stop frequency
330   INPUT "ENTER START FREQUENCY (MHz):",F_start
340   INPUT "ENTER STOP FREQUENCY (MHz):",F_stop
350 !
360 ! Program the analyzer settings
370   OUTPUT @Nwa;"STAR";F_start;"MHZ;"  ! Set the start frequency
380   OUTPUT @Nwa;"STOP";F_stop;"MHZ;"   ! Set the stop frequency
390 !
400 ! Autoscale the displays
410   OUTPUT @Nwa;"CHAN1;AUTO;"          ! Autoscale channel 1 display
420   OUTPUT @Nwa;"CHAN2;AUTO;"          ! Autoscale channel 2 display
430 !
440   PRINT "The display should now be autoscaled."
450 ! Release GPIB control
460   END
```

Example 1C: Setting Parameters on Four Channels for Optical Devices

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

This example sets the following parameters:

- data display formats
- number of channels and graticules displayed
- frequency range

The following is an outline of the program's processing sequence:

- An I/O path is assigned for the analyzer.
- The system is initialized.
- The analyzer is adjusted to measure return loss (S_{11}) on channel 1 and display as log magnitude.
- The analyzer is adjusted to measure return loss (S_{11}) on channel 2 and display the phase.
- The dual-channel display mode is activated.
- The system operator is prompted to enter the frequency range of the measurement.
- The displays are autoscaled.
- The system operator is prompted to press “Enter” on the computer keyboard to view four types of measurements simultaneously.
- Channels 3 and 4 are turned on.
- Channel 2 is adjusted to measure E/O transmission displayed in log magnitude, and then the display is autoscaled.
- Channel 3 is adjusted to measure reflected power (S22) displayed in log magnitude, and then the display is autoscaled.
- Channel 4 is adjusted to measureO/O transmission displayed in log magnitude, and then the display is autoscaled.
- The four channels are each displayed in a separate graticule.
- The analyzer is released from remote control and the program ends.

BASIC Program Listing

```
10  ! This program demonstrates setup of various measurement parameters such
20  ! as start frequency, stop frequency, etc.  The program first selects one
30  ! type of measurement to be viewed using dual-channel display format.
40  ! The specified start and stop frequencies are then programmed and the
50  ! analyzer display is autoscaled.  The program concludes by displaying
60  ! four types of measurements simultaneously.
70  !
80  ! LEXAMP1C Setting Parameters for Optical Devices
90  !
100   ASSIGN @Nwa TO 716                ! Assign an I/O path for the analyzer
110 !
120   CLEAR SCREEN
130 ! Initialize the system
140   ABORT 7                           ! Generate an IFC (Interface Clear)
150   CLEAR @Nwa                        ! SDC (Selected Device Clear) analyzer
160   OUTPUT @Nwa;”OPC?;PRES;”          ! Preset the analyzer and wait
170   ENTER @Nwa;Reply                  ! Read in the 1 returned
```

```
180 !
190 ! Set up measurement and display
200   OUTPUT @Nwa;"CHAN1;"              ! Channel 1
210   OUTPUT @Nwa;"AUXCOFF;"            ! Turn off auxiliary channel, if it is on
220   OUTPUT @Nwa;"S11;"                ! Return Loss (Reflection) measurement
230   OUTPUT @Nwa;"LOGM;"               ! Log magnitude display
240 !
250   OUTPUT @Nwa;"CHAN2;"              ! Channel 2
260   OUTPUT @Nwa;"AUXCOFF;"            ! Turn off auxiliary channel, if it is on
270   OUTPUT @Nwa;"S11;"                ! Return Loss (Reflection) measurement
280   OUTPUT @Nwa;"PHAS;"               ! Phase display
290 !
300   OUTPUT @Nwa;"DUACON;"             ! Dual channel display
310 !
320 ! Request start and stop frequency
330   INPUT "ENTER START FREQUENCY (MHz):",F_start
340   INPUT "ENTER STOP FREQUENCY (MHz):",F_stop
350 !
360 ! Program the analyzer settings
370   OUTPUT @Nwa;"STAR";F_start;"MHZ;"  ! Set the start frequency
380   OUTPUT @Nwa;"STOP";F_stop;"MHZ;"   ! Set the stop frequency
390 !
400 ! Autoscale the displays
410   OUTPUT @Nwa;"CHAN1;AUTO;"          ! Autoscale channel 1 display
420   OUTPUT @Nwa;"CHAN2;AUTO;"          ! Autoscale channel 2 display
430 !
440   PRINT "The display should now be autoscaled."
450   INPUT "Press RETURN to view four types of measurements simultaneously",X
460 !
470   OUTPUT @Nwa;"CHAN1;AUXCON;"        ! Turn on auxiliary channel (Channel 3)
480   OUTPUT @Nwa;"CHAN2;AUXCON;"        ! Turn on auxiliary channel (Channel 4)
490 !
500 ! E/O measurement
510   OUTPUT @Nwa;"MEASEO1;"
520   OUTPUT @Nwa;"LOGM;AUTO;"           ! Channel 2 log magnitude and autoscale
530 !
540 ! Channel 3 Reflected Power measurement
550   OUTPUT @Nwa;"CHAN3;S22;"           ! OUTPUT @NWA;"CHAN3#;#S22;"
560   OUTPUT @Nwa;"LOGM;AUTO;"           ! Channel 3 log magnitude and autoscale
570 !
580 ! Channel 4 O/O measurement
590   OUTPUT @Nwa;"CHAN4;MEASOE2;"       ! OUTPUT @NWA;"CHAN4;MEASOE2;"
600   OUTPUT @Nwa;"LOGM;AUTO;"           ! Channel 4 log magnitude and autoscale
```

```
610 !
620   OUTPUT @Nwa;"SPLID4;"              ! Display as four separate graticules
630 !
640   OUTPUT @Nwa;"OPC?;WAIT;"           ! Wait for the analyzer to finish
650   ENTER @Nwa;Reply                   ! Read the 1 when complete
660   LOCAL @Nwa                         ! Release GPIB control
670   END
```

Example 1D: Verifying Parameters

This example shows how to read analyzer settings into your controller. Appending a "?" to a command that sets an analyzer parameter will return the value of that setting. Parameters that are set as ON or OFF when queried will return a zero (0) if off or a one (1) if active. Parameters are returned in ASCII format, FORM 4. This format varies in length from 1 to 24 characters-per-value. In the case of marker or other multiple responses, the values are separated by commas.

The following is an outline of the program's processing sequence:

- An I/O path is assigned for the analyzer.
- The system is initialized.
- The number of points in the trace is queried and dumped to a printer.
- The start frequency is queried and output to a printer.
- The averaging is queried and output to a printer.
- The analyzer is released from remote control and the program ends.

Running the Program

The analyzer is preset. The preset values are returned and printed out for: the number of points, the start frequency, and the state of the averaging function. The analyzer is released from remote control and the program ends.

BASIC Program Listing

```
10  ! This program performs some example queries of network analyzer
20  ! settings. The number of points in a trace, the start frequency
30  ! and if averaging is turned on, are determined and displayed.
40  !
50  ! EXAMP1D Verifying Parameters
60  !
70  ASSIGN @Nwa TO 716                 ! Assign an I/O path for the analyzer
80  !
90  CLEAR SCREEN
100 ! Initialize the system
110 ABORT 7                            ! Generate an IFC (Interface Clear)
120 CLEAR @Nwa                         ! SDC (Selected Device Clear)
130 OUTPUT @Nwa;"OPC?;PRES;"           ! Preset the analyzer and wait
140 ENTER @Nwa;Reply                   ! Read in the 1 returned
150 !
```

```
160 ! Query network analyzer parameters
170 OUTPUT @Nwa;"POIN?;"                ! Read in the default trace length
180 ENTER @Nwa;Num_points
190 PRINT "Number of points ";Num_points
200 PRINT
210 !
220 OUTPUT @Nwa;"STAR?;"                ! Read in the start frequency
230 ENTER @Nwa;Start_f
240 PRINT "Start Frequency ";Start_f
250 PRINT
260 !
270 OUTPUT @Nwa;"AVERO?;"               ! Averaging on?
280 ENTER @Nwa;Flag
290 PRINT "Flag =";Flag;"   ";
300 IF Flag=1 THEN                      ! Test flag and print analyzer state
310   PRINT "Averaging ON"
320 ELSE
330   PRINT "Averaging OFF"
340 END IF
350 !
360 OUTPUT @Nwa;"OPC?;WAIT;"            ! Wait for the analyzer to finish
370 ENTER @Nwa;Reply                    ! Read the 1 when complete
380 LOCAL @Nwa                          ! Release GPIB control
390 END
```

Measurement Calibration Examples

This section shows you how to coordinate a measurement calibration over GPIB. You can use the following sequence for performing either a manual measurement calibration, or a remote measurement calibration via GPIB:

1. Select the calibration type.
2. Measure the calibration standards.
3. Declare the calibration done.

The actual sequence depends on the calibration kit and changes slightly for 2-port calibrations, which are divided into three calibration sub-sequences. The following examples are included:

- Example 2A is a program designed to perform a response calibration for E/E devices.
- Example 2B is a program designed to perform a response calibration for O/O, O/E, and E/O devices.
- Example 2C is a program designed to perform a 1-port measurement calibration for E/E devices.
- Example 2D is a program designed to perform an enhanced response calibration for E/E devices.
- Example 2E is a program designed to perform a full 2-port measurement calibration for E/E devices.
- Example 2F is a program designed to perform a response and match calibration for O/E devices.
- Example 2G is a program designed to accurately measure a “non-insertable” 2-port E/E device, using adapter removal.
- Example 2H is a program designed to use raw data to create a calibration, sometimes called Simmcal for E/E devices.
- Example 2I is a program designed to perform a response and match calibration for E/O devices.
- Example 2J is a program designed to offload the calculation of the 2-port error corrected data to an external computer.

Example programs for E/E devices illustrate how to perform different types of calibrations using any of the following calibration kits:

Calibration Kits
85052B/D (3.5-mm)
85056A/D (2.4-mm)

If you wish to use a different calibration kit, modify the example program accordingly. These programs simplify the calibration by providing explicit directions on the analyzer display while allowing the user to run the program from the controller keyboard. More

information on selecting calibration standards can be found in your analyzer's user's guide. For type-N connectors, the sex of the connector that the calibration standard will mate to must be observed. These programs assume that the connector on PORT 1 is a female test port and that PORT 2 is a male test port.

Calibration Kits

The calibration kit tells the analyzer what standards to expect at each step of the calibration. The set of standards associated with a given calibration is termed a "class." For example, measuring the short during a 1-port measurement calibration is one calibration step. All of the shorts that can be used for this calibration step make up the class, which is called class S11B. For the 7-mm and the 3.5-mm cal kits, class S11B uses only one standard. For type-N cal kits, class S11B contains two standards: male and female shorts.

When doing a 1-port measurement calibration using a 7- or 3.5-mm calibration kit, selecting **SHORT,** automatically measures the short because the class contains only one standard. When doing the same calibration in type-N, selecting **SHORT,** brings up a second menu, allowing the operator to select which standard in the class is to be measured. The sex listed refers to the test port: if the test port is female, then the operator selects the female short option. Once the standard has been selected and measured, the **DONE,** key must be pressed to exit the class.

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

The `STAN` command is OPC-compatible. A command that calls a class is only OPC-compatible if that class has only one standard in it. If there is more than one standard in a class, the command that calls the class brings up another menu, and there is no need to query it. `DONE;` must be sent to exit the class.

Example 2A: Response Calibration for E/E Devices

The following is an outline of the program's processing sequence:

- An I/O path is assigned for the analyzer.
- The system is initialized.
- The appropriate calibration kit is selected.
- The softkey menu is deactivated.
- The response calibration sequence is run.
- The response calibration data is saved.
- The softkey menu is activated.
- The analyzer is released from remote control and the program ends.

Running the Program

NOTE This program does not modify the instrument state in any way. Before running the program, set up the desired instrument state.

The program assumes that the test ports have a 7-mm, type-N 50 Ω, 3.5-mm, or 2.4-mm interface or an adapter set using a 7-mm, type-N 50 Ω, 3.5-mm, or 2.4-mm interface. The prompts appear just above the message line on the analyzer display. Pressing **Enter,** on the controller keyboard continues the program and measures the standard. The program will display a message when the measurement calibration is complete.

BASIC Program Listing

```
10  ! This program guides the operator through a response calibration
20  ! using a thru (cable).
120 !
130 ! The routine Waitforkey displays a message on the instrument's
140 ! display and the console, to prompt the operator to connect the
150 ! calibration standard. Once the standard is connected, the
160 ! ENTER key on the computer keyboard is pressed to continue.
170 !
180 ! EXAMP2A Response Calibration for E/E Devices
190 !
200   ASSIGN @Nwa TO 716                ! Assign an I/O path for the analyzer
210 !
220   CLEAR SCREEN
230 ! Initialize the system
240   ABORT 7                           ! Generate an IFC (Interface Clear)
250   CLEAR @Nwa                        ! SDC (Selected Device Clear)
390     OUTPUT @Nwa;"CALK35MD;"
430 !
440   OUTPUT @Nwa;"MENUOFF;"            ! Turn softkey menu off.
450 !
460   OUTPUT @Nwa;"CALIRESP;"           ! Response CAL initiated
470 !
480   CALL Waitforkey("CONNECT THRU BETWEEN PORTS")
520     OUTPUT @Nwa;"OPC?;STANC;"       ! Select the third standard, C
540   ENTER @Nwa;Reply                  ! Read in the 1 returned
550 !
560   DISP "COMPUTING CALIBRATION COEFFICIENTS"
570 !
580   OUTPUT @Nwa;"OPC?;RESPDONE;"      ! Finished with the CAL cycle
590   ENTER @Nwa;Reply                  ! Read in the 1 returned
600 !
610   DISP "RESPONSE CAL COMPLETED. CONNECT TEST DEVICE."
620   OUTPUT @Nwa;"MENUON;"             ! Turn on the softkey menu
630 !
640   OUTPUT @Nwa;"OPC?;WAIT;"          ! Wait for the analyzer to finish
650   ENTER @Nwa;Reply                  ! Read the 1 when complete
660   LOCAL @Nwa                        ! Release GPIB control
670 !
680   END
690 !
700 ! ************************** Subroutines *****************************
710 !
```

```
720 Waitforkey:  ! Prompt routine to read a keypress on the controller
730   SUB Waitforkey(Lab$)
740   !   Position and display text on the analyzer display
750     OUTPUT 717;”PG;PU;PA390,3700;PD;LB”;Lab$;”, PRESS ENTER WHEN READY;”&CHR$(3)
760   !
770     DISP Lab$&”  Press ENTER when ready”;! Display prompt on console
780     INPUT A$                          ! Read ENTER key press
790   !
800     OUTPUT 717;”PG;”                  ! Clear analyzer display
810   SUBEND
```

Example 2B: Response Calibration for O/O, E/O, O/E Devices

The following is an outline of the program's processing sequence:

- An I/O path is assigned for the analyzer.
- The system is initialized.
- The appropriate calibration kit is selected.
- The softkey menu is deactivated.
- The response calibration sequence is run.
- The response calibration data is saved.
- The softkey menu is activated.
- The analyzer is released from remote control and the program ends.

```
10  ! This program guides the operator through a response calibration
20  ! using a thru (optical cable for O/O devices or optical and electrical cables for
E/O and O/E devices).
30  !
40 !
50 ! The routine Waitforkey displays a message on the instrument’s
60 ! display and the console, to prompt the operator to connect the
70 ! calibration standard. Once the standard is connected, the
80 ! ENTER key on the computer keyboard is pressed to continue.
90 !
100 ! LEXAMP2B Response Calibration for O/O, E/O, O/E Devices
110 !
120   ASSIGN @Nwa TO 716                ! Assign an I/O path for the analyzer
130 !
140   CLEAR SCREEN
150 ! Initialize the system
160   ABORT 7                           ! Generate an IFC (Interface Clear)
170   CLEAR @Nwa                        ! SDC (Selected Device Clear)
180 ! Select Meas type
190   PRINT “Enter one of the following numbers:”
200   PRINT “1 to do an O/O Response Cal,”
```

```
210   PRINT “2 to do an O/E Response Cal,”
220   PRINT “3 to do an E/O Response Cal,”
230
240   INPUT Kit
250   SELECT Kit
260   CASE 1
270     OUTPUT @Nwa;”MEASOO1;”
280   CASE 2
290     OUTPUT @Nwa;”MEASOE1;”
300   CASE 3
310     OUTPUT @Nwa;”MEASEO1;”
320   END SELECT
330 !
340   OUTPUT @Nwa;”MENUOFF;”             ! Turn softkey menu off.
350 !
360   OUTPUT @Nwa;”CALIRESP;”            ! Response CAL initiated
370 !
380   CALL Waitforkey(“CONNECT THRUs BETWEEN ELECTRICAL AND OPTICAL PORTS”)
390     OUTPUT @Nwa;”OPC?;STANA;”        ! Select the third standard, C
400   ENTER @Nwa;Reply                   ! Read in the 1 returned
410 !
420   DISP “COMPUTING CALIBRATION COEFFICIENTS”
430 !
440   OUTPUT @Nwa;”OPC?;RESPDONE;”       ! Finished with the CAL cycle
450   ENTER @Nwa;Reply                   ! Read in the 1 returned
460 !
470   DISP “RESPONSE CAL COMPLETED. CONNECT TEST DEVICE.”
480   OUTPUT @Nwa;”MENUON;”              ! Turn on the softkey menu
490 !
500   OUTPUT @Nwa;”OPC?;WAIT;”           ! Wait for the analyzer to finish
510   ENTER @Nwa;Reply                   ! Read the 1 when complete
520   LOCAL @Nwa                         ! Release GPIB control
530 !
540   END
550 !
560 ! *************************** Subroutines ******************************
570 !
580 Waitforkey:  ! Prompt routine to read a keypress on the controller
590   SUB Waitforkey(Lab$)
600   !   Position and display text on the analyzer display
610     OUTPUT 717;”PG;PU;PA390,3700;PD;LB”;Lab$;”, PRESS ENTER WHEN READY;”&CHR$(3)
620   !
630     DISP Lab$&”  Press ENTER when ready”;! Display prompt on console
```

```
640     INPUT A$                              ! Read ENTER key press
650   !
660     OUTPUT 717;"PG;"                      ! Clear analyzer display
670   SUBEND
```

Example 2C: 1-Port Calibration for E/E Devices

The following is an outline of the program's processing sequence:

- An I/O path is assigned for the analyzer.
- The system is initialized.
- The appropriate calibration kit is selected.
- The softkey menu is deactivated.
- The 1-port calibration sequence is run.
- The 1-port calibration data is saved.
- The softkey menu is activated.
- The analyzer is released from remote control and the program ends.

Running the Program

NOTE This program does not modify the instrument state in any way. Before running the program, set up the desired instrument state.

The program assumes that the test ports have a 3.5-mm interface. Pressing **Enter,** on the controller keyboard continues the program and measures the standard. The program will display a message when the measurement calibration is complete.

BASIC Program Listing

```
10  ! This program guides the operator through a 1-port calibration.
120 ! The routine Waitforkey displays a message on the instrument's
130 ! display and the console, to prompt the operator to connect the
140 ! calibration standard. Once the standard is connected, the
150 ! ENTER key on the computer keyboard is pressed to continue.
160 !
170 ! EXAMP2C 1-Port Measurement Calibration for E/E Devices
180 !
190   ASSIGN @Nwa TO 716                 ! Assign an I/O path for the analyzer
200 !
210   CLEAR SCREEN
220 ! Initialize the system
230   ABORT 7                            ! Generate an IFC (Interface Clear)
240   CLEAR @Nwa                         ! SDC (Selected Device Clear)
380     OUTPUT @Nwa;"CALK35MD;"
430   OUTPUT @Nwa;"MENUOFF;"              ! Turn softkey menu off.
440 !
450   OUTPUT @Nwa;"CALIS111;"             ! S11 1 port CAL initiated
460 !
```

```
470 ! Open reflection CAL
480   CALL Waitforkey("CONNECT OPEN AT PORT 1 (REFLECTION PORT)")
530     OUTPUT @Nwa;"OPC?;CLASS11A;"      ! Only one standard in class
550   ENTER @Nwa;Reply                    ! Read in the 1 returned
560   OUTPUT @Nwa;"DONE;"                 ! Finished with class standards
570  !
580  ! Short reflection CAL
590   CALL Waitforkey("CONNECT SHORT AT PORT 1 (REFLECTION PORT)")
640     OUTPUT @Nwa;"OPC?;CLASS11B;"      ! Only one standard in class
660   ENTER @Nwa;Reply                    ! Read in the 1 returned
670   OUTPUT @Nwa;"DONE;"                 ! Finished with class standards
680  !
690  ! Reflection load CAL
700   CALL Waitforkey("CONNECT LOAD AT PORT 1 (REFLECTION PORT)")
720     OUTPUT @Nwa;"CLASS11C;"
730     OUTPUT @Nwa;"OPC?;STANA;"         ! Select the first standard, A
740   ELSE
750     OUTPUT @Nwa;"OPC?;CLASS11C;"      ! Only one standard in class
760   END IF
770   ENTER @Nwa;Reply                    ! Read in the 1 returned
780   OUTPUT @Nwa;"DONE;"                 ! Finished with class standards
790  !
800   DISP "COMPUTING CALIBRATION COEFFICIENTS"
810  !
820   OUTPUT @Nwa;"OPC?;SAV1;"            ! Save the ONE PORT CAL
830   ENTER @Nwa;Reply                    ! Read in the 1 returned
840 !
850   DISP "S11 1-PORT CAL COMPLETED. CONNECT TEST DEVICE."
860   OUTPUT @Nwa;"MENUON;"               ! Turn softkey menu on
870 !
880   OUTPUT @Nwa;"OPC?;WAIT;"            ! Wait for the analyzer to finish
890   ENTER @Nwa;Reply                    ! Read the 1 when complete
900   LOCAL @Nwa                          ! Release GPIB control
910 !
920   END
930 !
940 ! ************************** Subroutines *****************************
950 !
960 Waitforkey:   ! Prompt routine to read a keypress on the controller
970   SUB Waitforkey(Lab$)
980 !   Position and display text on the analyzer display
990     OUTPUT 717;"PG;PU;PA390,3700;PD;LB";Lab$;", PRESS ENTER WHEN READY;"&CHR$(3)
1000!
1010    DISP Lab$&"  Press ENTER when ready";! Display prompt on console
1020    INPUT A$                            ! Read ENTER key press
1030!
1040    OUTPUT 717;"PG;"                    ! Clear analyzer display
1050  SUBEND
```

Example 2D: Enhanced Response Calibration for E/E Devices

The following is an outline of the program's processing sequence:

- An I/O path is assigned for the analyzer.
- The system is initialized.

- The appropriate calibration kit is selected.
- The softkey menu is deactivated.
- The enhanced response calibration sequence is run.
- The enhanced response calibration data is saved.
- The softkey menu is activated.
- The analyzer is released from remote control and the program ends.

Running the Program

NOTE This program does not modify the instrument state in any way. Before running the program, set up the desired instrument state.

The program assumes that the test ports have a 3.5-mm interface. Pressing **Enter,** on the controller keyboard continues the program and measures the standard. The program will display a message when the measurement calibration is complete.

BASIC Program Listing

```
10  ! This program guides the operator through an enhanced response
20  ! calibration.
120 !
130 ! The routine Waitforkey displays a message on the instrument's
140 ! display and the console, to prompt the operator to connect the
150 ! calibration standard. Once the standard is connected, the
160 ! ENTER key on the computer keyboard is pressed to continue.
170 !
180 ! EXAMP2D Enhanced Response Calibration for E/E Devices
190 !
200   ASSIGN @Nwa TO 716                ! Assign an I/O path for the analyzer
210 !
220   CLEAR SCREEN
230 ! Initialize the system
240   ABORT 7                           ! Generate an IFC (Interface Clear)
250   CLEAR @Nwa                        ! SDC (Selected Device Clear)
380     OUTPUT @Nwa;"CALK35MD;"
420 !
430   OUTPUT @Nwa;"MENUOFF;"             ! Turn softkey menu off.
440 !
450   OUTPUT @Nwa;"CALIERC;"             ! Enhanced response CAL initiated
460 !
470   OUTPUT @Nwa;"REFL;"                ! Reflection CAL
480 !
490 ! Open reflection CAL
500   CALL Waitforkey("CONNECT OPEN AT PORT 1 (REFLECTION PORT)")
550     OUTPUT @Nwa;"OPC?;CLASS11A;"      ! Only one standard in class
570   ENTER @Nwa;Reply                    ! Read in the 1 returned
580   OUTPUT @Nwa;"DONE;"                 ! Finished with class standards
590  !
600  ! Short reflection CAL
610   CALL Waitforkey("CONNECT SHORT AT PORT 1 (REFLECTION PORT)")
660     OUTPUT @Nwa;"OPC?;CLASS11B;"      ! Only one standard in class
680   ENTER @Nwa;Reply                    ! Read in the 1 returned
```

```
690   OUTPUT @Nwa;"DONE;"                ! Finished with class standards
700  !
710  ! Reflection load CAL
720   CALL Waitforkey("CONNECT LOAD AT PORT 1 (REFLECTION PORT)")
740     OUTPUT @Nwa;"CLASS11C;"
750     OUTPUT @Nwa;"OPC?;STANA;"        ! Select the first standard, A
790   ENTER @Nwa;Reply                   ! Read in the 1 returned
800   OUTPUT @Nwa;"DONE;"                ! Finished with class standards
810  !
820   DISP "COMPUTING REFLECTION CALIBRATION COEFFICIENTS"
830  !
840   OUTPUT @Nwa;"REFD;"                 ! Reflection portion complete
850  !
860   OUTPUT @Nwa;"TRAN;"                 ! Transmission portion begins
870  !
880   CALL Waitforkey("CONNECT THRU BETWEEN PORTS")
890   DISP "MEASURING FORWARD TRANSMISSION"
900   OUTPUT @Nwa;"OPC?;FWDT;"            ! Measure (forward) transmission
910   ENTER @Nwa;Reply                    ! Read in the 1 returned
920  !
930   OUTPUT @Nwa;"OPC?;FWDM;"            ! Measure (forward) load match
940   ENTER @Nwa;Reply                    ! Read in the 1 returned
950  !
960   OUTPUT @Nwa;"TRAD;"                 ! Transmission CAL complete
970  !
980   INPUT "SKIP ISOLATION CAL? Y OR N.",An$
990   IF An$="Y" THEN
1000    OUTPUT @Nwa;"OMII;"               ! Skip isolation cal
1010    GOTO 1150
1020  END IF
1030 !
1040  CALL Waitforkey("ISOLATE TEST PORTS")
1050!
1060  OUTPUT @Nwa;"ISOL;"                ! Isolation CAL
1070  OUTPUT @Nwa;"AVERFACT10;"          ! Average for 10 sweeps
1080  OUTPUT @Nwa;"AVEROON;"             ! Turn on averaging
1090  DISP "MEASURING ISOLATION"
1100  OUTPUT @Nwa;"OPC?;FWDI;"           ! Measure (forward) isolation
1110  ENTER @Nwa;Reply                   ! Read in the 1 returned
1120!
1130  OUTPUT @Nwa;"ISOD;AVEROOFF;"       ! Isolation complete averaging off
1140!
1150  DISP "COMPUTING CALIBRATION COEFFICIENTS"
1160  OUTPUT @Nwa;"OPC?;ERCDONE;"        ! Finished with the CAL cycle
1170  ENTER @Nwa;Reply                   ! Read in the 1 returned
1180!
1190  DISP "ENHANCED RESPONSE CAL COMPLETED. CONNECT TEST DEVICE."
1200  OUTPUT @Nwa;"MENUON;"              ! Turn softkey menu on
1210!
1220  OUTPUT @Nwa;"OPC?;WAIT;"           ! Wait for the analyzer to finish
1230  ENTER @Nwa;Reply                   ! Read the 1 when complete
1240  LOCAL @Nwa                         ! Release GPIB control
1250!
1260  END
1270!
1280! *************************** Subroutines ******************************
1290!
1300 Waitforkey:  ! Prompt routine to read a keypress on the controller
```

```
1310  SUB Waitforkey(Lab$)
1320!   Position and display text on the analyzer display
1330    OUTPUT 717;"PG;PU;PA390,3700;PD;LB";Lab$;", PRESS ENTER WHEN READY;"&CHR$(3)
1340!
1350    DISP Lab$&"  Press ENTER when ready";! Display prompt on console
1360    INPUT A$                            ! Read ENTER key press
1370!
1380    OUTPUT 717;"PG;"                    ! Clear analyzer display
1390  SUBEND
```

Example 2E: Full 2-Port Measurement Calibration for E/E Devices

A full 2-port calibration removes both the forward- and reverse-error terms so that all four S-parameters of the device under test can be measured.

The following is an outline of the program's processing sequence:

- An I/O path is assigned for the analyzer.
- The system is initialized.
- The appropriate calibration kit is selected.
- The softkey menu is deactivated.
- The 2-port calibration sequence is run.
- The operator is prompted to choose or skip the isolation calibration.
- The softkey menu is activated.
- The analyzer is released from remote control and the program ends.

Running the Program

NOTE This program does not modify the instrument state in any way. Before running the program, set up the desired instrument state.

The program assumes that the test ports have either a 3.5-mm interface. After the prompt is displayed, pressing **Enter,** on the computer console continues the program and measures the standard. The operator has the option of omitting the isolation calibration. If the isolation calibration is performed, averaging is automatically employed to ensure a good calibration. The program will display a message when the measurement calibration is complete.

BASIC Program Listing

```
10  ! This program guides the operator through a full 2-port calibration.
      “          ’          “
110 !
120 ! The routine Waitforkey displays a message on the instrument’s
130 ! display and the console to prompt the operator to connect the
140 ! calibration standard. Once the standard is connected, the
150 ! ENTER key on the computer keyboard is pressed to continue.
160 !
170 ! EXAMP2E Full 2-Port Measurement Calibration for E/E Devices
180 !
190   ASSIGN @Nwa TO 716                  ! Assign an I/O path to the analyzer
200 !
210   CLEAR SCREEN
220 ! Initialize the analyzer
230   ABORT 7                             ! Generate an IFC (Interface Clear)
240   CLEAR @Nwa                          ! SDC (Selected Device Clear)
380     OUTPUT @Nwa;”CALK35MD;”
420 !
430   OUTPUT @Nwa;”MENUOFF;”              ! Turn softkey menu off.
440 !
450   OUTPUT @Nwa;”CALIFUL2;”             ! Full 2 port CAL
460 !
470   OUTPUT @Nwa;”REFL;”                 ! Reflection CAL
480 !
490 ! S11 open CAL
500   CALL Waitforkey(“CONNECT OPEN AT PORT 1”)
550     OUTPUT @Nwa;”OPC?;CLASS11A;”      ! Only one standard in class
570   ENTER @Nwa;Reply                    ! Read in the 1 returned
580   OUTPUT @Nwa;”DONE;”                 ! Finished with class standards
590  !
600  ! S11 short CAL
610   CALL Waitforkey(“CONNECT SHORT AT PORT 1”)
660     OUTPUT @Nwa;”OPC?;CLASS11B;”      ! Only one standard in class
680   ENTER @Nwa;Reply                    ! Read in the 1 returned
690   OUTPUT @Nwa;”DONE;”                 ! Finished with class standards
700  !
710  ! S11 load CAL
720   CALL Waitforkey(“CONNECT LOAD AT PORT 1”)
740     OUTPUT @Nwa;”CLASS11C;”
750     OUTPUT @Nwa;”OPC?;STANA;”         ! Select the first standard, A
790   ENTER @Nwa;Reply                    ! Read in the 1 returned
800   OUTPUT @Nwa;”DONE;”                 ! Finished with class standards
810 !
820 ! S22 open CAL
830   CALL Waitforkey(“CONNECT OPEN AT PORT 2”)
880     OUTPUT @Nwa;”OPC?;CLASS22A;”      ! Only one standard in class
900   ENTER @Nwa;Reply                    ! Read in the 1 returned
910   OUTPUT @Nwa;”DONE;”                 ! Finished with class standards
920  !
930  ! S22 short CAL
940   CALL Waitforkey(“CONNECT SHORT AT PORT 2”)
990     OUTPUT @Nwa;”OPC?;CLASS22B;”      ! Only one standard in class
1010  ENTER @Nwa;Reply                    ! Read in the 1 returned
1020  OUTPUT @Nwa;”DONE;”                 ! Finished with class standards
1030 !
1040 ! S22 load CAL
```

```
1050  CALL Waitforkey("CONNECT LOAD AT PORT 2")
1070    OUTPUT @Nwa;"CLASS22C;"
1080    OUTPUT @Nwa;"OPC?;STANA;"         ! Select the first standard, A
1120  ENTER @Nwa;Reply                     ! Read in the 1 returned
1130  OUTPUT @Nwa;"DONE;"                  ! Finished with class standards
1140 !
1150  DISP "COMPUTING REFLECTION CALIBRATION COEFFICIENTS"
1160 !
1170  OUTPUT @Nwa;"REFD;"                   ! Reflection portion complete
1180 !
1190  OUTPUT @Nwa;"TRAN;"                   ! Transmission portion begins
1200 !
1210  CALL Waitforkey("CONNECT THRU [PORT1 TO PORT 2]")
1220  DISP "MEASURING FORWARD TRANSMISSION"
1230  OUTPUT @Nwa;"OPC?;FWDT;"              ! Measure forward transmission
1240  ENTER @Nwa;Reply                      ! Read in the 1 returned
1250 !
1260  OUTPUT @Nwa;"OPC?;FWDM;"              ! Measure forward load match
1270  ENTER @Nwa;Reply                      ! Read in the 1 returned
1280 !
1290  DISP "MEASURING REVERSE TRANSMISSION"
1300  OUTPUT @Nwa;"OPC?;REVT;"              ! Measure reverse transmission
1310  ENTER @Nwa;Reply                      ! Read in the 1 returned
1320 !
1330  OUTPUT @Nwa;"OPC?;REVM;"              ! Measure reverse load match
1340  ENTER @Nwa;Reply                      ! Read in the 1 returned
1350 !
1360  OUTPUT @Nwa;"TRAD;"                   ! Transmission CAL complete
1370 !
1380  INPUT "SKIP ISOLATION CAL? Y OR N.",An$
1390  IF An$="Y" THEN
1400    OUTPUT @Nwa;"OMII;"                 ! Skip isolation cal
1410    GOTO 1600
1420  END IF
1430  !
1440  CALL Waitforkey("ISOLATE TEST PORTS")
1450 !
1460  OUTPUT @Nwa;"ISOL;"                   ! Isolation CAL
1470  OUTPUT @Nwa;"AVERFACT10;"             ! Average for 10 sweeps
1480  OUTPUT @Nwa;"AVEROON;"                ! Turn on averaging
1490  DISP "MEASURING REVERSE ISOLATION"
1500  OUTPUT @Nwa;"OPC?;REVI;"              ! Measure reverse isolation
1510  ENTER @Nwa;Reply                      ! Read in the 1 returned
1520 !
1530  DISP "MEASURING FORWARD ISOLATION"
1540  OUTPUT @Nwa;"OPC?;FWDI;"              ! Measure forward isolation
1550  ENTER @Nwa;Reply                      ! Read in the 1 returned
1560 !
1570  OUTPUT @Nwa;"ISOD;AVEROOFF;"          ! Isolation complete averaging off
1580  OUTPUT 717;"PG;"                      ! Clear analyzer display prompt
1590 !
1600  DISP "COMPUTING CALIBRATION COEFFICIENTS"
1610  OUTPUT @Nwa;"OPC?;SAV2;"              ! Save THE TWO PORT CAL
1620  ENTER @Nwa;Reply                      ! Read in the 1 returned
1630 !
1640  DISP "DONE WITH FULL 2-PORT CAL. CONNECT TEST DEVICE."
1650  OUTPUT @Nwa;"MENUON;"                 ! Turn softkey menu on
1660 !
```

```
1670  OUTPUT @Nwa;"OPC?;WAIT;"             ! Wait for the analyzer to finish
1680  ENTER @Nwa;Reply                     ! Read the 1 when complete
1690  LOCAL @Nwa                           ! Release GPIB control
1700 !
1710  END
1720 !
1730 ! ************************ Subroutines ******************************
1740 !
1750  SUB Waitforkey(Lab$)
1760 ! Position and display prompt on the analyzer display
1770    OUTPUT 717;"PG;PU;PA390,3700;PD;LB";Lab$;", PRESS ENTER WHEN READY;"&CHR$(3)
1780 !
1790    DISP Lab$&"   Press ENTER when ready";   ! Display prompt on console
1800    INPUT A$                              ! Read ENTER keypress on controller
1810    OUTPUT 717;"PG;"                      ! Clear analyzer display
1820  SUBEND
```

Example 2F: Response and Match Calibration for O/E Devices

A response and match calibration effectively removes the frequency response errors of the test setup for transmission measurements of electrical-to-optical devices.

The following is an outline of the program's processing sequence:

- An I/O path is assigned for the analyzer.
- The system is initialized.
- The appropriate calibration kit is selected.
- The softkey menu is deactivated.
- The response and match calibration sequence is run.
- The operator is prompted to choose or skip the isolation calibration.
- The softkey menu is activated.
- The analyzer is released from remote control and the program ends.

BASIC Program Listing

```
10  ! This program guides the operator through a response and match calibration.
20  !
30 !
40 ! The routine Waitforkey displays a message on the instrument's
50 ! display and the console to prompt the operator to connect the
60 ! calibration standard. Once the standard is connected, the
70 ! ENTER key on the computer keyboard is pressed to continue.
80 !
90 ! EXAMP2F Response and Match Measurement Calibration for O/E Devices
100 !
110   ASSIGN @Nwa TO 716                  ! Assign an I/O path to the analyzer
120 !
130   CLEAR SCREEN
140 ! Initialize the analyzer
150   ABORT 7                             ! Generate an IFC (Interface Clear)
160   CLEAR @Nwa                          ! SDC (Selected Device Clear)
170 !
180   OUTPUT @Nwa;"MENUOFF;"              ! Turn softkey menu off.
190   OUTPUT @Nwa;"MEASOE2;"              ! O/E port 2
200   OUTPUT @Nwa;"CALIOERM;"             ! Response & Match CAL
210 !
220   OUTPUT @Nwa;"REFL;"                 ! Reflection CAL
230 !
240 ! Forward open CAL
250   CALL Waitforkey("CONNECT OPEN AT PORT 1")
260   !
270   OUTPUT @Nwa;"OPC?;CLASS11A;"      ! Only one standard in class
280   ENTER @Nwa;Reply                    ! Read in the 1 returned
290   OUTPUT @Nwa;"DONE;"                 ! Finished with class standards
300  !
310  ! Forward short CAL
320   CALL Waitforkey("CONNECT SHORT AT PORT 1")
330   !
340   OUTPUT @Nwa;"OPC?;CLASS11B;"      ! Only one standard in class
350   !
360   ENTER @Nwa;Reply                    ! Read in the 1 returned
370   OUTPUT @Nwa;"DONE;"                 ! Finished with class standards
380  !
390  ! Forward load CAL
400   CALL Waitforkey("CONNECT LOAD AT PORT 1")
410   OUTPUT @Nwa;"CLASS11C;"
```

```
420   OUTPUT @Nwa;"OPC?;STANA;"         ! Select the first standard, A
430   ENTER @Nwa;Reply                    ! Read in the 1 returned
440   OUTPUT @Nwa;"DONE;"                 ! Finished with class standards
450 !
460 ! Reverse open CAL
470   CALL Waitforkey("CONNECT OPEN AT PORT 2")
480 !
490   OUTPUT @Nwa;"OPC?;CLASS22A;"      ! Only one standard in class
500   ENTER @Nwa;Reply                    ! Read in the 1 returned
510   OUTPUT @Nwa;"DONE;"                 ! Finished with class standards
520 !
530 ! Reverse short CAL
540   CALL Waitforkey("CONNECT SHORT AT PORT 2")
550 !
560   OUTPUT @Nwa;"OPC?;CLASS22B;"      ! Only one standard in class
570  ENTER @Nwa;Reply                    ! Read in the 1 returned
580  OUTPUT @Nwa;"DONE;"                 ! Finished with class standards
590 !
600 ! Reverse load CAL
610  CALL Waitforkey("CONNECT LOAD AT PORT 2")
620  OUTPUT @Nwa;"CLASS22C;"
630  OUTPUT @Nwa;"OPC?;STANA;"           ! Select the first standard, A
640  ENTER @Nwa;Reply                    ! Read in the 1 returned
650  OUTPUT @Nwa;"DONE;"                 ! Finished with class standards
660 !
670  DISP "COMPUTING REFLECTION CALIBRATION COEFFICIENTS"
680 !
690  OUTPUT @Nwa;"REFD;"                  ! Reflection portion complete
700 !
710  OUTPUT @Nwa;"TRAN;"                  ! Transmission portion begins
720 !
730  CALL Waitforkey("CONNECT THRUS BETWEEN ELECTRICAL AND OPTICAL PORTS")
740  DISP "MEASURING FORWARD TRANSMISSION"
750  OUTPUT @Nwa;"OPC?;FWDT;"             ! Measure forward transmission
760  ENTER @Nwa;Reply                     ! Read in the 1 returned
765 ! Select Calibration Standard Type
770  PRINT "Enter one of the following numbers"
775  PRINT "1. Do THRUS standard"
780  PRINT "2. Do THRU/SRC standard
785  INPUT std type
790  ! Select Std Type
795  CASE 1
800    OUTPUT @Nwa;"STANA;"
```

```
802  CASE 2
804    OUTPUT @ NWa;"STANB;"
810  OUTPUT @Nwa;"DONE;"                  ! Finished with class standards
820 !
830  OUTPUT @Nwa;"OPC?;FWDM;"             ! Measure forward match thru
840  ENTER @Nwa;Reply                     ! Read in the 1 returned
850 !
890 !
920 !
930  OUTPUT @Nwa;"TRAD;"                  ! Transmission CAL complete
940 !
950  INPUT "SKIP ISOLATION CAL? Y OR N.",An$
960  IF An$="Y" THEN
970    OUTPUT @Nwa;"OMII;"                ! Skip isolation cal
980    GOTO 1180                          !"COMPUTING CALIBRATION COEFFICIENTS"
990  END IF
1000  !
1010  CALL Waitforkey("ISOLATE TEST PORTS")
1020 !
1030  OUTPUT @Nwa;"ISOL;"                  ! Isolation CAL
1040  OUTPUT @Nwa;"AVERFACT10;"            ! Average for 10 sweeps
1050  OUTPUT @Nwa;"AVEROON;"               ! Turn on averaging
1060  DISP "MEASURING REVERSE ISOLATION"
1070  OUTPUT @Nwa;"OPC?;REVI;"             ! Measure reverse isolation
1080  ENTER @Nwa;Reply                     ! Read in the 1 returned
1090 !
1100  DISP "MEASURING FORWARD ISOLATION"
1110  OUTPUT @Nwa;"OPC?;FWDI;"             ! Measure forward isolation
1120  ENTER @Nwa;Reply                     ! Read in the 1 returned
1130 !
1140  OUTPUT @Nwa;"ISOD;AVEROOFF;"         ! Isolation complete averaging off
1150  OUTPUT 717;"PG;"                     ! Clear analyzer display prompt
1160 !
1170  OUTPUT @Nwa;"REMD;                  ! press Done Response & Match
1180  DISP "COMPUTING CALIBRATION COEFFICIENTS"
1190  OUTPUT @Nwa;"OPC?;SAVE2;"             ! Save THE TWO PORT CAL
1200  ENTER @Nwa;Reply                     ! Read in the 1 returned
1210 !
1220  DISP "DONE WITH RESPONSE AND MATCH CAL. CONNECT TEST DEVICE."
1230  OUTPUT @Nwa;"MENUON;"                ! Turn softkey menu on
1240 !
1250  OUTPUT @Nwa;"OPC?;WAIT;"             ! Wait for the analyzer to finish
1260  ENTER @Nwa;Reply                     ! Read the 1 when complete
```

```
1270  LOCAL @Nwa                          ! Release GPIB control
1280 !
1290  END
1300 !
1310 ! ************************ Subroutines ******************************
1320 !
1330  SUB Waitforkey(Lab$)
1340 ! Position and display prompt on the analyzer display
1350    OUTPUT 717;"PG;PU;PA390,3700;PD;LB";Lab$;", PRESS ENTER WHEN READY;"&CHR$(3)
1360 !
1370    DISP Lab$&"   Press ENTER when ready";   ! Display prompt on console
1380    INPUT A$                            ! Read ENTER keypress on controller
1390    OUTPUT 717;"PG;"                    ! Clear analyzer display
1400  SUBEND
```

Example 2G: Adapter Removal Calibration for E/E Devices

This program shows how to accurately measure a "non-insertable" 2-port device. A device is termed "non-insertable" if its connectors do not match those of the analyzer front panel. More information on the adapter removal technique can be found in your analyzer's user's guide.

The following is an outline of the program's processing sequence:

- An I/O path is assigned for the analyzer.
- The system is initialized.
- The internal disk is selected as the active storage device.
- The system operator is prompted for the name of the instrument state file which has a 2-port calibration performed for Port 1's connector.
- The calibration arrays for Port 1 are recalled from the corresponding disk file.
- The system operator is prompted for the known electrical delay value of the adapter.
- The new calibration coefficients, with the effects of the adapter removed, are computed by the analyzer using the adapter delay in conjunction with the calibration arrays for both ports.
- The analyzer is released from remote control and the program ends.

CAUTION Do not mistake the line switch for the disk eject button. If the line switch is mistakenly pushed, the instrument will be turned off, losing all settings and data that have not been saved.

Running the Program

The analyzer is initialized and the internal disk drive is selected. The operator is queried for

the name of the instrument state file having a 2-port calibration performed for Port 1's connector. The calibration arrays for Port 1 are recalled from the corresponding disk file. The system operator is prompted for the name of the instrument state file having a 2-port calibration performed for Port 2's connector. The calibration arrays for Port 2 are recalled from the corresponding disk file. The system operator is prompted for the known electrical delay of the adapter and this value is written to the analyzer. The calibration coefficients with adapter effects removed are computed and the program ends.

BASIC Program Listing

```
1     ! This program demonstrates how to do adapter removal over GPIB.
2     !
3     ! EXAMP2G Adapter Removal Calibration for E/E Devices
4     !
5 REAL Delay                  ! Adapter electrical delay in picoseconds
6     !
7 ASSIGN @Nwa TO 716          ! Assign an I/O path for the analyzer
8 CLEAR SCREEN
9     ! Initialize the system
10 ABORT 7                    ! Generate an IFC (Interface Clear)
11 CLEAR @Nwa                 ! SDC (Selected Device Clear) analyzer
12 OUTPUT @Nwa;"OPC?;PRES;"   ! Preset the analyzer and wait
13 ENTER @Nwa;Reply           ! Read in the 1 returned
14    !
15    ! Select internal disk.
16    !
17 OUTPUT @Nwa;"INTD;"
18    !
19    ! Assign file #1 to the filename that has a 2-port
20    ! cal previously performed for Port 1's connector.
21    !
22 PRINT "Enter the name of the instrument state file which"
23 PRINT "has a 2-port cal performed for Port 1's connector"
24 INPUT "",F1$
25 OUTPUT @Nwa;"TITF1""";F1$;""";"
26    !
27    ! Recall the cal set for Port 1.
28    !
29 DISP "Loading cal arrays, please wait"
30 OUTPUT @Nwa;"CALSPORT1;"
31 OUTPUT @Nwa;"OPC?;NOOP;"
32 ENTER @Nwa;Reply
33    !
34    ! Assign file #2 to the filename that has a 2-port
35    ! cal previously performed for Port 2's connector.
36    !
37 CLEAR SCREEN
38 PRINT "Enter the name of the instrument state file which"
39 PRINT "has a 2-port cal performed for Port 2's connector"
40 INPUT "",F2$
41 OUTPUT @Nwa;"INTD;TITF2""";F2$;""";"
42    !
43    ! Recall the cal set for Port 2.
44    !
45 DISP "Loading cal arrays, please wait"
46 OUTPUT @Nwa;"CALSPORT2;"
47 OUTPUT @Nwa;"OPC?;NOOP;"
48 ENTER @Nwa;Reply
49    !
50    ! Set the adapter electrical delay.
51    !
52 INPUT "Enter the electrical delay for the adapter in picoseconds",Delay
53 OUTPUT @Nwa;"ADAP1"&VAL$(Delay)&"PS;"
54    !
55    ! Perform the "remove adapter" computation.
56    !
```

```
57 DISP “Computing cal coefficients...”
58 OUTPUT @Nwa;”MODS;”
59 OUTPUT @Nwa;”OPC?;WAIT;”
60 ENTER @Nwa;Reply
61 LOCAL 7                    ! Release GPIB control
62 DISP “Program completed”
63 END
```

Example 2H: Using Raw Data to Create a Calibration (Simmcal) for E/E Devicse

This program simulates a full 2-port cal by measuring the raw data for each “standard” and then loading it later into the appropriate arrays. The program can be adapted to create additional calibrations using the same arrays. It uses the analyzer’s default 3.5mm cal kit.

CAUTION This feature is not currently supported with TRL calibrations.

The following is an outline of the programs' processing sequence:

- An I/O path is assigned for the analyzer.
- The system is initiated.
- The instrument ID is queried so that the program can later branch accordingly for the S11C and S22C (load) classes.
- The number of points is set to correspond to the size of the dimensioned memory arrays and ASCII data format is selected.
- The 3.5-mm calibration kit is selected, sweep time is set to 1 second, and the analyzer is placed into hold mode.
- S11 measurement is selected for gathering the forward reflection standards.
- The system operator is prompted to connect each of the three standards, one at a time.
- Following each prompt, a single sweep is taken and the raw measured data for that standard is read from the analyzer into a corresponding memory array in the controller.
- S22 measurement is selected for gathering the reverse reflection standards.
- The system operator is prompted in the same manner as before and the raw data for the three standards is measured and stored away as before.
- The system operator is prompted to make the thru connection between Port 1 and Port 2.
- S21 measurement is selected, a single sweep is taken and the raw data is read into an array corresponding to forward transmission.
- S11 measurement is selected, a single sweep is taken and the raw data is read into an array corresponding to forward thru match.
- S12 measurement is selected, a single sweep is taken and the raw data is read into an array corresponding to reverse transmission.
- S22 measurement is selected, a single sweep is taken and the raw data is read into an

array corresponding to reverse thru match.

- The analyzer begins the normal 2-port calibration procedure, but with the default beep turned off.
- A single sweep is taken for the measurement of each standard to provide “dummy” data, which is immediately replaced with the previously measured raw data from the array corresponding to that measurement.
- The analyzer uses the raw data to compute the error coefficients and is placed back into continuous sweep mode.
- The analyzer is released from remote control and the program ends.

Running the Program

The system is initialized, the number of points is set to 51, and the 3.5-mm calibration kit is selected. Sweep time is set to 1 second and the analyzer is placed into hold mode.

The S11 measurement is selected and the system operator is prompted to connect each of the three forward reflection standards, one at a time. Following each prompt, a single sweep is taken, which concludes with a beep from the external controller.

The S22 measurement is selected for gathering the reverse reflection standards. The system operator is prompted in the same manner as before and the three standards are measured as before.

The system operator is prompted to make the thru connection between Port 1 and Port 2. A single sweep is taken for each of the four S-parameters, each concluding with a beep.

The analyzer begins the normal 2-port calibration procedure, but with the default beep turned off. A single sweep is taken for each measurement of each standard, providing “dummy” data which is immediately replaced with the data from the array corresponding to that measurement. The analyzer computes the error correction coefficients and is placed back into continuous sweep mode. The default beep is re-enabled and the program ends.

```
10    ! This program simulates a full 2-port cal by first getting the
20    ! raw data for each “standard” and then loading it into the
30    ! appropriate arrays later.
100   !
110   ! EXAMP2H Using Raw Data to Create a Calibration (Simmcal) for E/E Devices
120   !
130   !
140   !
150   ! Allocate the arrays. The numbers correspond to the subsequent
160   ! cal coefficient array that will be written.
170   !
180   DIM Array01(1:51,1:2) ! forward OPEN measurement
190   DIM Array02(1:51,1:2) ! forward SHORT
200   DIM Array03(1:51,1:2) ! forward LOAD
210   DIM Array04(1:51,1:2) ! forward ISOLATION if necessary
220   DIM Array05(1:51,1:2) ! forward LOAD MATCH
```

```
230   DIM Array06(1:51,1:2) ! forward TRANS
240   DIM Array07(1:51,1:2) ! reverse OPEN
250   DIM Array08(1:51,1:2) ! reverse SHORT
260   DIM Array09(1:51,1:2) ! reverse LOAD
270   DIM Array10(1:51,1:2) ! reverse ISOLATION if necessary
280   DIM Array11(1:51,1:2) ! reverse LOAD MATCH
290   DIM Array12(1:51,1:2) ! reverse TRANS
300   !
310   ! Initialize the system
320   ASSIGN @Nwa TO 716          ! Assign an I/O path for the analyzer
330   ABORT 7                     ! Generate an IFC (Interface Clear)
340   CLEAR @Nwa                  ! SDC (Selected Device Clear) analyzer
350   CLEAR SCREEN
360   !
440   ! Preset the analyzer, set to 51 points, ASCII format, desired cal
450   ! kit definition (3.5mm).  Sweep time set to 1 second (could be whatever
460   ! user would like), analyzer put in hold mode.
470   !
480   OUTPUT @Nwa;"opc?;pres;"
490   ENTER @Nwa;X
500   OUTPUT @Nwa;"POIN51;FORM4;"
510   OUTPUT @Nwa;"CALK3.5MM;SWET1S;HOLD;"
520   !
530   ! Select S11 to gather the forward reflection standards
540   ! (open, short, load).
550   !
560   OUTPUT @Nwa;"S11;"
570   INPUT "CONNECT OPEN AT PORT 1",X
580   OUTPUT @Nwa;"opc?;sing;"
590   ENTER @Nwa;X
600   BEEP
610   OUTPUT @Nwa;"OUTPRAW1"
620   ENTER @Nwa;Array01(*)
630   !
640   INPUT "CONNECT SHORT AT PORT 1",X
650   OUTPUT @Nwa;"opc?;sing;"
660   ENTER @Nwa;X
670   BEEP
680   OUTPUT @Nwa;"OUTPRAW1"
690   ENTER @Nwa;Array02(*)
700   !
710   INPUT "CONNECT BROADBAND LOAD AT PORT 1",X
720   OUTPUT @Nwa;"opc?;sing;"
```

```
730   ENTER @Nwa;X
740   BEEP
750   OUTPUT @Nwa;”OUTPRAW1”
760   ENTER @Nwa;Array03(*)
770   !
780   ! Now select S22 to gather the reverse reflection standards.
790   !
800   OUTPUT @Nwa;”S22”
810   INPUT “CONNECT OPEN AT PORT 2”,X
820   OUTPUT @Nwa;”opc?;sing;”
830   ENTER @Nwa;X
840   BEEP
850   OUTPUT @Nwa;”OUTPRAW1”
860   ENTER @Nwa;Array07(*)
870   !
880   INPUT “CONNECT SHORT AT PORT 2”,X
890   OUTPUT @Nwa;”opc?;sing;”
900   ENTER @Nwa;X
910   BEEP
920   OUTPUT @Nwa;”OUTPRAW1”
930   ENTER @Nwa;Array08(*)
940   !
950   INPUT “CONNECT BROADBAND LOAD AT PORT 2”,X
960   OUTPUT @Nwa;”opc?;sing;”
970   ENTER @Nwa;X
980   BEEP
990   OUTPUT @Nwa;”OUTPRAW1”
1000  ENTER @Nwa;Array09(*)
1010  !
1020  INPUT “CONNECT THRU [PORT1 TO PORT 2]”,X
1030  !
1040  ! Now select S21 to gather forward transmission raw array.
1050  !
1060  DISP “MEASURING FORWARD TRANSMISSION”
1070  OUTPUT @Nwa;”S21;OPC?;SING;”
1080  ENTER @Nwa;Reply
1090  BEEP
1100  OUTPUT @Nwa;”OUTPRAW1”
1110  ENTER @Nwa;Array06(*)
1120  !
1130  ! Now select S11 to gather forward match raw array.
1140  !
1150  OUTPUT @Nwa;”S11;OPC?;SING;”
```

```
1160  ENTER @Nwa;Reply
1170  BEEP
1180  OUTPUT @Nwa;"OUTPRAW1"
1190  ENTER @Nwa;Array05(*)
1200  !
1210  ! Now select S12 for reverse transmission raw array.
1220  !
1230  DISP "MEASURING REVERSE TRANSMISSION"
1240  OUTPUT @Nwa;"S12;OPC?;SING;"
1250  ENTER @Nwa;Reply
1260  BEEP
1270  OUTPUT @Nwa;"OUTPRAW1"
1280  ENTER @Nwa;Array12(*)
1290  !
1300  ! Now select S22 for reverse match raw array.
1310  !
1320  OUTPUT @Nwa;"S22;OPC?;SING;"
1330  ENTER @Nwa;Reply
1340  BEEP
1350  OUTPUT @Nwa;"OUTPRAW1"
1360  ENTER @Nwa;Array11(*)
1370  !
1380  ! Done with gathering measurements except for isolation.  If
1390  ! isolation desired, then put forward isolation into 'Array04',
1400  ! reverse isolation into 'Array10'.
1410  !
1420  ! Now download and let analyzer compute the full 2-port error
1430  ! correction.
1440  !
1450  ! First select the calibration type desired.
1460  !
1470  OUTPUT @Nwa;"CALIFUL2;"
1480  !
1490  ! Turn off the beep indicating standard done.
1500  !
1510  OUTPUT @Nwa;"BEEPDONEOFF;"
1520  !
1530  ! Set up for the reflection standards.
1540  !
1550  OUTPUT @Nwa;"REFL;"
1560  !
1570  ! Input the forward 'open' standard's raw array.  For all of
1580  ! these, the analyzer is first taking a "dummy" measurement, goes
```

```
1590  ! into hold, then the computer downloads the data using an
1600  ! INPUCALC command which overwrites the “dummy” data with the raw
1610  ! array gathered previously.
1620  !
1630  OUTPUT @Nwa;”OPC?;CLASS11A;”
1640  ENTER @Nwa;Reply
1650  OUTPUT @Nwa;”INPUCALC01”,Array01(*)
1660  !
1670  ! Input the forward ‘short’ standard’s raw array.
1680  !
1690  OUTPUT @Nwa;”OPC?;CLASS11B;”
1700  ENTER @Nwa;Reply
1710  OUTPUT @Nwa;”INPUCALC02”,Array02(*)
1720  !
1730  ! Input the forward ‘load’ standards’s raw array.
1740  !
1780    OUTPUT @Nwa;”CLASS11C;OPC?;STANA;”
1800  ENTER @Nwa;Reply
1810  OUTPUT @Nwa;”INPUCALC03”,Array03(*)
1820  !
1830  ! Input reverse ‘open’.
1840  !
1850  OUTPUT @Nwa;”OPC?;CLASS22A;”
1860  ENTER @Nwa;Reply
1870  OUTPUT @Nwa;”INPUCALC07”,Array07(*)
1880   !
1890   ! Input reverse ‘short’.
1900   !
1910  OUTPUT @Nwa;”OPC?;CLASS22B;”
1920  ENTER @Nwa;Reply
1930  OUTPUT @Nwa;”INPUCALC08”,Array08(*)
1940   !
1950   ! Input reverse ‘load’.
1960   !
2000    OUTPUT @Nwa;”CLASS22C;OPC?;STANA;”
2020  ENTER @Nwa;Reply
2030  OUTPUT @Nwa;”INPUCALC09”,Array09(*)
2040   !
2050   ! Tell analyzer that reflection measurements done.
2060   !
2070  OUTPUT @Nwa;”REFD;”
2080  DISP “COMPUTING REFLECTION CALIBRATION COEFFICIENTS”
2090   !
```

```
2100   ! Now start the transmission standard downloads.
2110   !
2120  OUTPUT @Nwa;"TRAN;"
2130   !
2140   ! Now input the forward transmission raw arrays.
2150   !
2160  OUTPUT @Nwa;"OPC?;FWDT;"
2170  ENTER @Nwa;Reply
2180  OUTPUT @Nwa;"INPUCALC06",Array06(*)
2190   !
2200  OUTPUT @Nwa;"OPC?;FWDM;"
2210  ENTER @Nwa;Reply
2220  OUTPUT @Nwa;"INPUCALC05",Array05(*)
2230   !
2240   ! Now input the reverse transmission arrays.
2250   !
2260   !DISP "MEASURING REVERSE TRANSMISSION"
2270  OUTPUT @Nwa;"OPC?;REVT;"
2280  ENTER @Nwa;Reply
2290  OUTPUT @Nwa;"INPUCALC12",Array12(*)
2300   !
2310  OUTPUT @Nwa;"OPC?;REVM;"
2320  ENTER @Nwa;Reply
2330  OUTPUT @Nwa;"INPUCALC11",Array11(*)
2340   !
2350   ! Tell analyzer that transmission inputs done.
2360   !
2370  OUTPUT @Nwa;"TRAD"
2380   !
2390   ! Omitting isolation for this example.  Could be easily
2400   ! incorporating by using method shown for tranmission and
2410   ! reflection.
2420   !
2430  OUTPUT @Nwa;"ISOL;"
2440  OUTPUT @Nwa;"OMII;"              !IF ISOLATION CAL NOT DESIRED
2450  !  Here's how to download isolation.  Un-comment these lines.
2460  !
2470  !OUTPUT @Nwa;"OPC?;REVI;" ! reverse isolation term
2480  !ENTER @Nwa;Reply
2490  !OUTPUT @Nwa;"INPUCALC10",Array10(*)
2500  !
2510  !OUTPUT @Nwa;"OPC?;FWDI;" ! forward isolation term
2520  !ENTER @Nwa;Reply
```

```
2530  !OUTPUT @Nwa;"INPUCALC04",Array04(*)
2540  !
2550  ! Tell analyzer that done with isolation measurements.
2560  !
2570  OUTPUT @Nwa;"ISOD;"
2580  DISP "COMPUTING CALIBRATION COEFFICIENTS"
2590   !
2600   ! Tell analyzer to compute full 2-port error coefficients.
2610   !
2620  OUTPUT @Nwa;"OPC?;SAV2;"
2630  ENTER @Nwa;Reply
2640  DISP "DONE"
2650   !
2660   ! Put analyzer back into continuous sweep so that you can verify
2670   ! the proper application of the error correction.
2680   !
2690  OUTPUT @Nwa;"CONT;"
2700  OUTPUT @Nwa;"BEEPDONEON;"    ! Re-enable the beep
2710  LOCAL 7                      ! Release GPIB control
2720  END
```

Example 2I: Response and Match Calibration for E/O Devices

A response and match calibration effectively removes the frequency response errors of the test setup for transmission measurements of optical-to-electrical devices.

The following is an outline of the program's processing sequence:

- An I/O path is assigned for the analyzer.
- The system is initialized.
- The softkey menu is deactivated.
- The operator is prompted to connect the standards for the reflection portion of the calibration.
- The operator is prompted to connect the standards for the transmission portion of the calibration.
- The operator is prompted to connect the standards for the isolation portion of the calibration.
- The calibration is saved.
- The analyzer is released from remote control and the program ends.

```
10  ! This program guides the operator through an E/O Response & Match calibration.
20  !
30  !
```

```
40 ! The routine Waitforkey displays a message on the instrument's
50 ! display and the console, to prompt the operator to connect the
60 ! calibration standard. Once the standard is connected, the
70 ! ENTER key on the computer keyboard is pressed to continue.
80 !
90 ! LEXAMP2I Response and Match Calibration for E/O Devices
100 !
110   ASSIGN @Nwa TO 716                ! Assign an I/O path for the analyzer
120 !
130   CLEAR SCREEN
140 ! Initialize the system
150   ABORT 7                           ! Generate an IFC (Interface Clear)
160   CLEAR @Nwa                        ! SDC (Selected Device Clear)
170   OUTPUT @Nwa;"MENUOFF;"            ! Turn softkey menu off.
180 !
190   OUTPUT @Nwa;"MEASEO1;"            ! E/O Measurment
200   OUTPUT @Nwa;"CALIEORM;"           ! Response & MaAtch 1 port CAL initiated
210   OUTPUT @Nwa;"SOFT1;"
220 ! Open reflection CAL
230   CALL Waitforkey("CONNECT OPEN AT PORT 1 (REFLECTION PORT)")
240 !
250   OUTPUT @Nwa;"OPC?;CLASS11A;"
260 !
270   ENTER @Nwa;Reply                   ! Read in the 1 returned
280 ! OUTPUT @Nwa;"DONE;"                ! Finished with class standards
290 !
300 ! Short reflection CAL
310   CALL Waitforkey("CONNECT SHORT AT PORT 1 (REFLECTION PORT)")
320 !
330   OUTPUT @Nwa;"OPC?;CLASS11B;"      ! Only one standard in class
340 !
350   ENTER @Nwa;Reply                   ! Read in the 1 returned
360 ! OUTPUT @Nwa;"DONE;"                ! Finished with class standards
370 !
380 ! Reflection load CAL
390   CALL Waitforkey("CONNECT LOAD AT PORT 1 (REFLECTION PORT)")
400   OUTPUT @Nwa;"CLASS11C;"
410   OUTPUT @Nwa;"OPC?;STANA;"         ! Select the first standard, A
420   ENTER @Nwa;Reply                   ! Read in the 1 returned
430   OUTPUT @Nwa;"DONE;"                ! Finished with class standards
440   OUTPUT @Nwa;"REFD;"                ! Finished with class standards
450  !
460   DISP "COMPUTING CALIBRATION COEFFICIENTS"
```

```
470  !
480 ! Starting FWD TRANS THRUS
490   OUTPUT @Nwa;"FWDT;"
500   CALL Waitforkey("CONNECT THRUS BETWEEN ELECTRICAL AND OPTICAL PORTS")
505   PRINT "Enter one of the following numbers"
510   PRINT "1. Do THRUS standard"
515   PRINT "2. Do THRUS/RCVR standard"
520   INPUT Std type
525 ! Select Std type
530   CASE 1
535     OUTPUT @Nwa;"STANA;"
540   CASE 2
545     OUTPUT @Nwa;"STANB;"
560  OUTPUT @Nwa;"DONE;"
570 ! Select Isolation
580 OUTPUT @Nwa;"ISOL;"
590 CALL Waitforkey("ISOLATE PORTS ")
600 OUTPUT @Nwa;"OMII;"
610 !
620 OUTPUT @Nwa;"RAMD;"
630 !
640   OUTPUT @Nwa;"OPC?;SAVE1;"            ! Save the ONE PORT CAL
650   ENTER @Nwa;Reply                    ! Read in the 1 returned
660   DISP "E/O RESPONSE & MATCH 1-PORT CAL COMPLETED. CONNECT TEST DEVICE."
670   OUTPUT @Nwa;"MENUON;"               ! Turn softkey menu on
680 !
690   OUTPUT @Nwa;"OPC?;WAIT;"            ! Wait for the analyzer to finish
700   ENTER @Nwa;Reply                    ! Read the 1 when complete
710   LOCAL @Nwa                          ! Release GPIB control
720   END
730 !
740 ! *************************** Subroutines ******************************
750 !
760 Waitforkey:   ! Prompt routine to read a keypress on the controller
770   SUB Waitforkey(Lab$)
780 !   Position and display text on the analyzer display
790     OUTPUT 717;"PG;PU;PA390,3700;PD;LB";Lab$;", PRESS ENTER WHEN READY;"&CHR$(3)
800!
810    DISP Lab$&"  Press ENTER when ready";! Display prompt on console
820    INPUT A$                            ! Read ENTER key press
830 !
840    OUTPUT 717;"PG;"                    ! Clear analyzer display
850  SUBEND
```

Example 2J: Take4 — Error Correction Processed on an External PC for E/E Devices

Take4 mode offloads the error correction process to an external PC in order to increase throughput on the analyzer.

When using the analyzer with error correction turned off, it will only sweep in one direction, collecting data for the parameter selected under the **Meas,** key. To emulate the error correction process in an external computer, you collect the raw data for each of the four S-parameters.

Take4 initiates a mode in which every measurement cycle is characterized by sweeping in both the forward and reverse directions and collecting raw data for all four S-parameters. Using previously extracted calibration arrays, you can then extract the raw data (or the pre-raw data, as explained later in this section) for the S-parameters and perform the error correction in an external computer. When measuring more than one parameter, this process can be done in less time than if using the normal instrument error correction and data transfer (Refer to Table 8-8 on page 8-57).

NOTE This mode is intended for remote use only. Any attempt to change the measured parameter or any attempt to apply a calibration will turn off the Take4 mode. The displayed trace data is always uncorrected S11, regardless of what the display may indicate.

Using the Take4 mode requires the following steps:

Manual steps:

1. Set up the measurement state.
2. Turn off raw offsets by selecting **System, CONFIGURE MENU, RAW OFFSET OFF.** This selection achieves two things:
   - Eliminates attenuator offsets and sampler hardware offsets from the cal arrays, which are generated in the 2-port error correction. This makes the cal arrays and the eventual OUTPPRE arrays compatible, both using pre-raw data.
   - Eliminates sampler correction, a frequency response correction that is normally contained in pre-raw data. This is done because sampler correction is not needed for data that will be fully corrected, and because instrument states recall faster without it. To realize this efficiency, you must also disable spur avoidance (see next step).
3. Optional step: Turn off spur avoidance by selecting **System, CONFIGURE MENU, SPUR AVOID OFF.** Spur avoidance creates a table of values as part of the sampler offset table. The creation of this table takes considerable time during a recall of an instrument state. Turning off spur avoidance will save time during frequency changes and instrument state recalls.
4. Perform a 2-port error correction and save it to a register.
5. Connect the device under test (DUT).

The instrument is now configured for the program to read the correction arrays and apply the Take4 mode.

Programming steps:

6. Extract the twelve calibration arrays using the commands OUTPCALC[01-12].
7. Enable Take4 mode using the command TAKE4ON.
8. Take a sweep and extract the four pre-raw or raw arrays.
   - To extract pre-raw data arrays (see previous discussion on raw offsets), you can use the commands SWPSTART (initiate a single sweep) with OUTPPRE[1-4]. These commands are more efficient than SING and OUTPRAF[1-4] because the analyzer will respond to OUTPPRE1 and OUTPPRE2 as soon as the forward sweep is done and transfer the data during the reverse sweep. With SING, the GPIB bus is held off until the entire sweep is complete.
   - To extract raw data arrays, you can use the commands SING (initiate a single sweep) with OUTPRAW[1-4], or the slightly faster OUTPRAF[1-4]. If the cal arrays were created using **RAW OFFSET ON,** you should use this method so that your measurement data is compatible with the calibration data.
9. Apply the calibration arrays (see Table 4-3 on page 4-21) to either the pre-raw or raw data as described in programming example 2H and in the user's guide (see the figure titled “Full 2-Port Error Model”).

Table 8-8. Measurement Speed: Data Output and Error Correction to an External PC

Mode (data output to external PC)	Time (secs) 1-parameter	Time (secs) 2-parameters	Time (secs) 3-parameters	Time (secs) 4-parameters
Full band, IF BW=3700, 201 points, SPUR AVOID OFF, RAW OFFSET OFF, Blank Display ON				
Take4	0.780	0.780	0.780	0.780
Normal error correction	0.712	0.907	0.970	1.03
Narrow band, IF BW=3700, 201 points, CF=1.8GHz, Span=200MHz, SPUR AVOID OFF, ,RAW OFFSET OFF, Blank Display ON				
Take4	0.215	0.215	0.215	0.215
Normal error correction	0.151	0.224	0.290	0.350
Take4 mode used in conjunction with an laptop, 133 MHz Pentium, running Agilent VEE 4.0 as program language.				

Programming example 2J is the complete execution of a two port error correction offloaded to an external PC.

The following is an outline of the program's processing sequence:

- An I/O path is assigned for the analyzer. Binary mode is used for data transfers in order to get the fastest response.
- The system is initialized.
- The state of raw offsets is queried and turned off if they had been on.
- The analyzer is placed into local mode and the system operator is prompted to set up a 2-port calibration before continuing.
- The calibration coefficients are read from the analyzer into memory arrays.
- The calibration is turned off and the analyzer is placed into TAKE4 mode and HOLD mode.
- The operator is prompted to connect the DUT and select which S-parameter to send back to the analyzer.
- The currently displayed data is saved to the analyzer's internal memory to initialize the memory array.
- The analyzer is set up to display memory only and the default beep is turned off.
- The operator is prompted to press any key to terminate the program.
- A sweep is initiated and the main loop of the program begins.
- After the sweep concludes, the four pre-raw S-parameters are read from the analyzer into an array in the computer.

- The error-corrected (calibrated) S-parameters are calculated using the pre-raw data and calibration coefficients.
- The calibrated data for the S-parameter selected earlier is sent into the analyzer and saved to the analyzer's internal memory.
- A new sweep is initiated and the loop repeats if there has been no keyboard activity.
- Upon exit of the loop, the analyzer is set up to display the active measurement trace.
- The analyzer's internal calibration is turned back on and continuous sweep mode is resumed.
- The analyzer is released from remote control and the program ends.

Running the Program

The analyzer is initialized and raw offsets are turned off. After the analyzer is placed in local mode, the operator is prompted to set up a 2-port calibration before continuing. The resulting calibration coefficients are read from the analyzer into memory arrays.

Next, the calibration is turned off and the analyzer is placed into TAKE4 mode and HOLD mode. After being prompted to connect the DUT, the operator selects which S-parameter to send back to the analyzer. The currently displayed data is saved to the analyzer's internal memory and the analyzer is set up to display memory only. The operator is prompted to press any key to terminate the program, a sweep is initiated and the main loop of the program begins.

After the sweep concludes, the four pre-raw S-parameters are read from the analyzer into memory arrays. The error-corrected (calibrated) S-parameters are calculated and the calibrated data for the S-parameter selected earlier is read into the analyzer and saved to the analyzer's internal memory. A new sweep is initiated and the loop repeats if there has been no keyboard activity.

Upon exit of the loop, the analyzer is set up to display the active measurement trace. The analyzer's internal calibration is turned back on and continuous sweep mode is resumed before the program ends.

BASIC Program Listing

```
1     ! This program demonstrates the TAKE4 mode.
2     ! The program first asks the user to set up the instrument
3     ! with a 2-port calibration.  The subroutine “Read_Cal_co”
4     ! is used to read the 12 term error correction arrays into
5     ! a (N x 12) 2-dimension array (N = number of points).  This will
6     ! be used in the “Calc_2_port” subroutine.  The program turns off
7     ! error correction, puts the analyzer in hold, turns on TAKE4
8     ! mode, and starts a sweep.  The subroutine “Read_4_raw” reads in
9     ! the uncorrected data.  The subroutine “Calc_2_port” calculates
10    ! the error correction and returns the corrected arrays.
11    ! The corrected S-parameter is re-input to the analyzer, stored
12    ! in the memory trace and displayed in memory for a visual
13    ! indication of the take4 function.
14    !
15    ! EXAMP2J Take 4-Error Correction Processed on an External PC
16    !
17    !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
```

```
18    !
19    ! Initialize Arrays and Variables
20    !
21    !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
22    !
23 INTEGER Hdr,Length
24 COMPLEX S11x,S21x,S12x,S22x,D
25 COMPLEX Calcoe(1:1601,1:12)    ! Cal Coefficients
26 COMPLEX S11r(1:1601)           ! Pre-Raw Data
27 COMPLEX S21r(1:1601)           ! Pre-Raw Data
28 COMPLEX S12r(1:1601)           ! Pre-Raw Data
29 COMPLEX S22r(1:1601)           ! Pre-Raw Data
30 COMPLEX S11a(1:1601)           ! Corrected Data
31 COMPLEX S21a(1:1601)           ! Corrected Data
32 COMPLEX S12a(1:1601)           ! Corrected Data
33 COMPLEX S22a(1:1601)           ! Corrected Data
34    !
35    ! Initialize output commands
36    !
37 DIM Out_cmd$(1:12)[10]
38 DATA “OUTPCALC01”,”OUTPCALC02”,”OUTPCALC03”,”OUTPCALC04”
39 DATA “OUTPCALC05”,”OUTPCALC06”,”OUTPCALC07”,”OUTPCALC08”
40 DATA “OUTPCALC09”,”OUTPCALC10”,”OUTPCALC11”,”OUTPCALC12”
41 READ Out_cmd$(*)
42    !
43    ! Setup Network Analyzer
44    !
45 ASSIGN @Nwa TO 716             ! Assign an I/O path for the analyzer
46 ASSIGN @Nwdat TO 716;FORMAT OFF! Binary mode to read and write data
47 ABORT 7                        ! Generate an IFC (Interface Clear)
48 CLEAR @Nwa                    ! SDC (Selected Device Clear) analyzer
49 CLEAR SCREEN
50    !
51 OUTPUT @Nwa;”RAWOFFS?;”       ! Query whether raw offsets are on
52 ENTER @Nwa;I
53 IF I=1 THEN
54 PRINT “Raw offsets must be turned off prior to calibration.”
55 PRINT “Turning them off now.”
56 OUTPUT @Nwa;”RAWOFFSOFF;”
57 END IF
58    !
59    !
60 Check_for_cal:                 ! Turn on two-port cal, check and read
61 REPEAT
62 LOCAL @Nwa
63 INPUT “Set up a 2-port cal, hit return when ready”,A
64 OUTPUT @Nwa;”CORR?;”
65 ENTER @Nwa;I
66 UNTIL I=1
67    !
68    ! Read the Calibration Coefficients
69    !
70 DISP “Reading in Calibration Coefficient Arrays: Please wait.”
71 GOSUB Read_cal_co
72    !
73    ! Setup TAKE4 Mode,
74    !
75 OUTPUT @Nwa;”Corroff;take4on;hold;”
```

```
76    !
77    ! Choose an S-Parameter to send back to the Network Analyzer
78    !
79 REPEAT
80 INPUT “SELECT S-Parameter: 1=S11, 2=S21, 3=S12, 4=S22”,Disp
81 SELECT Disp
82 CASE 1
83  Title$=”S11”
84  Again=0
85 CASE 2
86  Title$=”S21”
87  Again=0
88 CASE 3
89  Title$=”S12”
90  Again=0
91 CASE 4
92  Title$=”S22”
93  Again=0
94 CASE ELSE
95  Again=1
96 END SELECT
97 UNTIL Again=0
98 OUTPUT @Nwa;”TITL”””&Title$&”””;”
99    !
100   ! For this demonstration, we will return corrected values to the
101   ! memory trace.  Therefore, display memory only
102   !
103   !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
104   !
105   ! Note:  Displaying MEMORY only inhibits the analyzer’s data
106   !        processing.  Raw, Data, and Formatted arrays are not
107   !        updated.  PreRaw is good.
108   !
109 OUTPUT @Nwa;”DATI;DISPMEMO;BEEPDONEOFF;”
110 PRINT “PRESS ANY KEY TO STOP”
111 Time1=TIMEDATE
112   !
113   ! Take the first sweep
114   !
115 OUTPUT @Nwa;”OPC?;SWPSTART;”
116 Run=1
117 Count=0
118   !
119   ! Now keep looping until any key is pressed
120   !
121 Timefmt:IMAGE “Cycle: “,2D,5X,” 2-port Cal: “,2D.DD,X,”secs, + displayed:
“,2D.DDD,X,”seconds.”
122 ON KBD GOSUB Stop_running
123 REPEAT
124 Count=Count+1
125 ENTER @Nwa;Done             ! Read the OPC from the SWPSTART Command
126 GOSUB Read_4_raw            ! Read the four raw S-parameters
127 GOSUB Calc_2_port           ! Calculate the Corrected S-parameters
128 Time2=TIMEDATE
129 OUTPUT @Nwa;”INPUDATA;”     ! Input them into the data array
130 OUTPUT @Nwdat;Hdr,Length    ! Data header, same as the cal coeff’s
131 SELECT Disp
132 CASE 1
```

```
133 OUTPUT @Nwdat;S11a(*)       ! Send corrected S11 data to analyzer OR
134 CASE 2                      !
135 OUTPUT @Nwdat;S21a(*)       ! Send corrected S21 data to analyzer OR
136 CASE 3                      !
137 OUTPUT @Nwdat;S12a(*)       ! Send corrected S12 data to analyzer OR
138 CASE 4
139 OUTPUT @Nwdat;S22a(*)       ! Send corrected S22 data to analyzer
140 END SELECT
141 OUTPUT @Nwa;"DATI;"         ! Put the data into memory
142 OUTPUT @Nwa;"OPC?;SWPSTART;"! and start another sweep
143 Time3=TIMEDATE
144 DISP USING Timefmt;Count,Time2-Time1,Time3-Time1
145 Time1=TIMEDATE
146 UNTIL Run=0
147 OUTPUT @Nwa;"DISPDATA;CORRON;CONT;"
148 ABORT 7
149 LOCAL @Nwa
150 STOP
151 Stop_running:   ! Terminate program upon keyboard input
152 Run=0
153 OFF KBD
154 RETURN
155 Read_4_raw:     ! Read in the pre-raw arrays
156 A$="OUTPPRE"
157 FOR B=1 TO 4
158 Out_cmd1$=A$&VAL$(B)&";"    ! Build up the OUTPPREXX commands
159 OUTPUT @Nwa;Out_cmd1$
160 ENTER @Nwdat;Hdr,Length     ! Read in the header
161 SELECT B
162      !
163      !  Now read in each raw array
164      !
165 CASE 1
166   ENTER @Nwdat;S11r(*)
167 CASE 2
168   ENTER @Nwdat;S21r(*)
169 CASE 3
170   ENTER @Nwdat;S12r(*)
171 CASE 4
172   ENTER @Nwdat;S22r(*)
173 END SELECT
174 NEXT B
175 RETURN
176 Read_cal_co:    ! This loops through 12 times, reading each cal.
177                 ! coefficient.  First set up the FORM
178 OUTPUT @Nwa;"FORM3;HOLD;"
179 OUTPUT @Nwa;"POIN?;"
180 ENTER @Nwa;Numpoints
181   !
182   ! Redimension the Calcoe array according to the number of points
183   !
184 REDIM Calcoe(1:Numpoints,1:12)
185   !
186   ! Also redimension all the other arrays used here, as this
187   ! routine only runs once at setup.
188   !
189 REDIM S11a(1:Numpoints)
190 REDIM S21a(1:Numpoints)
```

```
191 REDIM S12a(1:Numpoints)
192 REDIM S22a(1:Numpoints)
193 REDIM S11r(1:Numpoints)
194 REDIM S21r(1:Numpoints)
195 REDIM S12r(1:Numpoints)
196 REDIM S22r(1:Numpoints)
197 FOR Cx=1 TO 12
198 OUTPUT @Nwa;Out_cmd$(Cx)    ! OUTPCALCXC commands
199 ENTER @Nwdat;Hdr,Length     ! Read the header using FORMAT OFF mode
200 FOR N=1 TO Numpoints
201   ENTER @Nwdat;Calcoe(N,Cx) ! Read data using FORMAT OFF mode
202 NEXT N
203 NEXT Cx
204   !
205 RETURN
206 Calc_2_port:    ! Perform 2 Port Calibration
207 FOR N=1 TO Numpoints
208      !
209      ! First correct for crosstalk, directivity, and tracking
210      !
211      !  Subtract Directivity, divide by tracking
212 S11x=(S11r(N)-Calcoe(N,1))/Calcoe(N,3)
213      !
214      !  Subtract Crosstalk, divide by tracking
215 S21x=(S21r(N)-Calcoe(N,4))/Calcoe(N,6)
216      !
217      !  Subtract Crosstalk, divide by tracking
218 S12x=(S12r(N)-Calcoe(N,10))/Calcoe(N,12)
219      !
220      !  Subtract Directivity, divide by tracking
221 S22x=(S22r(N)-Calcoe(N,7))/Calcoe(N,9)
222      !
223      ! Now calculate the common denominator
224      !
225 D=(1+S11x*Calcoe(N,2))*(1+S22x*Calcoe(N,8))-(S21x*S12x*Calcoe(N,5)*Calcoe(N,11))
227      !
228      ! Now calculate each S-parameter
229      !
230 S11a(N)=((S11x*(1+S22x*Calcoe(N,8)))-(S21x*S12x*Calcoe(N,5)))/D
231 S21a(N)=((1+S22x*(Calcoe(N,8)-Calcoe(N,5)))*(S21x))/D
232 S12a(N)=((1+S11x*(Calcoe(N,2)-Calcoe(N,11)))*(S12x))/D
233 S22a(N)=((S22x*(1+S11x*Calcoe(N,2)))-(S21x*S12x*Calcoe(N,11)))/D
234 NEXT N
235 RETURN
236 END
```

Measurement Data Transfer Examples

There are two methods that can be used to read trace information from the analyzer:

- selectively, using the trace markers
- completely, using the trace-data array

If only specific information (such as a single point on the trace or the result of a marker search) is required, the marker output command can be used to read the information. If all of the trace data is required, see Examples 3B through 3E for examples of the various formats available.

Trace-Data Formats and Transfers

Refer to Table 8-9 on page 8-66. This table shows the number of bytes required to transfer a 201-point trace in the different formats. As you will see in the first example (FORM 4), ASCII data is the easiest to transfer, but the most time consuming due to the number of bytes in the trace. If you are using a PC-based controller, a more suitable format would be FORM 5. To use any trace data format other than FORM 4 (ASCII data) requires some care in transferring the data to the computer. Data types must be matched to read the bytes from the analyzer directly into the variable array. The computer must be told to stop formatting the incoming data and treat it as a binary-data transfer. All of the binary data formats also have a four-byte header to deal with. The first two bytes are the ASCII characters "#A" that indicate that a fixed length block transfer follows, and the next two bytes form an integer containing the number of bytes in the block to follow. The header must be read in to separate it from the rest of the block data that is to be mapped into an array. "Array-Data Formats" on page 5-7 discusses the different types of formats and their compositions.

Data may also be transferred from several different locations in the trace-processing chain. These examples will illustrate formatted-data transfers, but other locations in the trace-data processing chains may be accessed. See Figure 6-1 on page 6-3.

In this section, an example of each of the data formats will be shown for comparison. In general, FORM 1 (internal binary format) should be used for traces that are not being utilized for data content. Calibration data that is being transferred to a file and back is good example. See "Example 3D: Data Transfer Using Frequency-Array Information" on page 8-70.

Arrays which will be interpreted or processed within your program should be in FORM 2, 3 or 5, whichever is appropriate for your computer. Example 3C shows how to transfer a trace in these formats.

In Examples 3B and 3C, the frequency counterpart of each data point in the array is also determined. Many applications generate a frequency and magnitude, or a phase array for the test results. Such data may be required for other data processing applications (such as comparing data from other measurements).

In Example 3B, the frequency data is constructed from the frequency span information. Alternatively, it is possible to read the frequencies directly out of the instrument with the `OUTPLIML` command. `OUTPLIML` reports the limit-test results by transmitting the stimulus point tested, a number indicating the limit-test results, and then the upper and lower limits at that stimulus point (if available). The number indicating the limit results is a –1 for no

test, 0 for fail, and 1 for pass. If there are no limits available, the analyzer transmits zeros. For this example, we delete the limit test information and keep the stimulus information.

In Example 3C, the limit-test array is read into the controller and used to provide the values directly from the analyzer memory. Reading from the limit-test array is convenient, although it outputs the results in ASCII format (form 4), which may be slow. If there is no other way to obtain the frequency data, this transfer time may be acceptable. Frequency information becomes more difficult to determine when not using the linear sweep mode. Log-frequency sweeps and list-frequency sweeps have quite different values for each data point. For these special cases, the additional time spent reading out the limit test results is an acceptable solution for obtaining the valid frequency information for each data point in the trace.

Example 3A: Data Transfer Using Markers

Markers are the simplest form of trace-data transfer. A marker may be positioned using one of three methods:

- by a frequency location
- by an actual data point location
- by a trace-data value

In the following example, the marker is positioned on the trace's maximum value. Once positioned on the trace, the trace data at that point can be read into the controller. The marker data is always returned in FORM 4, ASCII format. Each number is sent as a 24-character string. Characters can be digits, signs, or decimal points. All characters should be separated by commas. In the case of markers, three numbers are sent. The display format determines the values of the marker responses. See Table 5-1 on page 5-6.

When using trace data, it is important to control the analyzer's sweep function (and therefore the trace data) from the computer. Using the computer to control the instrument's sweep ensures that the data you read into the controller is in a quiescent or steady state. It also ensures that the measurement is complete.

The following is an outline of the program's processing sequence:

- An I/O path is assigned for the analyzer.
- The system is initialized.
- The selected frequency span is swept once.
- The marker is activated and placed on the maximum trace value.
- The three marker values are output to the controller and displayed.
- The instrument is returned to local control and the program ends.

Running the Program

Run the program. The analyzer is preset and a sweep is taken. Marker 1 is enabled and positioned on the largest value in the trace. The marker is output to the controller and printed on the controller display. The analyzer is returned to local control. Position the marker using the front panel knob or data-entry keys, and compare the displayed value on the analyzer with the value that was transmitted to the controller.

The three values returned to the controller are:

1. reflection, in dB
2. a non-significant value
3. the stimulus frequency at the maximum point

A non-significant value means that the analyzer returned a value that is meaningless in this data format.

Table 5-1 on page 5-6 provides an easy reference for the types of data returned with the various data-format operational modes.

BASIC Program Listing

```
10  ! This program takes a sweep on the analyzer and turns on a marker.
20  ! The marker is positioned on the trace maximum and the marker data
30  ! is output in ASCII format.
40  !
50  ! EXAMP3A Data Transfer Using Markers
60  !
70  ASSIGN @Nwa TO 716                  ! Assign an I/O path for the analyzer
80  !
90  CLEAR SCREEN
100 ! Initialize the analyzer
110 ABORT 7                             ! Generate an IFC (Interface Clear)
120 CLEAR @Nwa                          ! SDC (Selective Device Clear)
130 OUTPUT @Nwa;"OPC?;PRES;"            ! Preset the analyzer and wait
140 ENTER @Nwa;Reply                    ! Read in the 1 returned
150 !
160 OUTPUT @Nwa;"OPC?;SING"             ! Single sweep mode and wait
170 ENTER @Nwa;Reply                    ! Read 1 when sweep complete
180 !
190 OUTPUT @Nwa;"MARK1;"                ! Turn on marker 1
200 OUTPUT @Nwa;"SEAMAX;"               ! Find the maximum
210 !
220 OUTPUT @Nwa;"OUTPMARK;"             ! Request the current marker value
230 ENTER @Nwa;Value1,Value2,Stim       ! Read three marker values
240 !
250 ! Show the marker data received.
260 PRINT " Value 1"," Value 2","     Stimulus (Hz)"
270 PRINT Value1,Value2,Stim            ! Print the received values
280 PRINT
290 PRINT " Compare the active marker block with the received values"
300 !
310 LOCAL @Nwa                          ! Release GPIB control
320 END
```

Example 3B: Data Transfer Using FORM 4 (ASCII Transfer)

This example shows you how to transfer a trace array from the analyzer using FORM 4, an ASCII data transfer.

Table 8-9. Analyzer Array-Data Formats

Format type	Type of Data	Bytes per Data Value	Bytes per point 2 data values	(201 pts) Bytes per trace	Total Bytes with header
FORM 1	Internal Binary	3	6	1206	1210
FORM 2	IEEE 32-bit Floating-Point	4	8	1608	1612
FORM 3	IEEE 64-bit Floating-Point	8	16	3216	3220
FORM 4	ASCII Numbers	24 (Typical)	50 (Typical)	10,050 (Typical)	10,050* (Typical)
FORM 5	PC-DOS 32-bit Floating-Point	4	8	1608	1612
*No header is used in FORM 4.					

The next most common data transfer is to transfer a trace array from the analyzer. Table 8-9 shows the relationship of the two values-per-point that are transferred to the analyzer. When FORM 4 is used, each number is sent as a 24-character string, each character represented by a digit, sign, or decimal point. Each number is separated from the previous number with a comma. Since there are two numbers-per-point, a 201-point transfer in FORM 4 takes 10,050 bytes. This form is useful only when input-data formatting is difficult with the instrument controller. Refer to Table 8-9 for a comparison with the other formats.

An example of a simple data transfer using FORM 4 (ASCII data transfer) is shown in this program. A fairly common requirement is to create frequency-amplitude data pairs from the trace data. No frequency information is included with the trace data transfer, because the frequency data must be calculated. Relating the data from a linear frequency sweep to frequency can be done by querying the analyzer start frequency, the frequency span, and the number of points in the sweep. Given that information, the frequency of point N in a linear frequency sweep is:

$$F = StartFrequency + (N-1) \times \frac{Span}{(Points-1)}$$

Example 3B illustrates this technique. It is a straight-forward solution for linear uniform sweeps. For other sweep types, frequency data is more difficult to construct and may best be read directly from the analyzer's limit-test array. See “Example 3D: Data Transfer Using Frequency-Array Information” on page 8-70.

The following is an outline of the program's processing sequence:

- An I/O path is assigned for the analyzer.
- The system is initialized.

- The trace-data array is allocated.
- The trace length is set to 11.
- The selected frequency span is swept once.
- The FORM 4, ASCII format is set.
- The formatted trace is read from the analyzer and displayed.
- The frequency increments between the points are calculated.
- The marker is activated and placed at the lowest frequency of the analyzer (50 MHz).
- The instrument is returned to local control and the program ends.

Running the Program

Run the program and watch the controller console. The analyzer will perform an instrument preset. The program will then print out the data values received from the analyzer. The marker is activated and placed at the left-hand edge of the analyzer's display. Position the marker with the knob and compare the values read with the active marker with the results printed on the controller console. The data points should agree exactly. Keep in mind that no matter how many digits are displayed, the analyzer is specified to measure:

- magnitude to a resolution of 0.001 dB
- phase to a resolution of 0.01 degrees
- group delay to a resolution of 0.01 ps

Changing the display format will change the data sent with the OUTPFORM transfer. See Table 8-9 for a list of the specific data that is provided with each format. The data from OUTPFORM reflects all the post processing such as:

- electrical delay
- trace math
- smoothing

BASIC Program Listing

```
10  ! This program shows an ASCII format trace data transfer using form 4.
20  ! The data is received as a string of ASCII characters, 24 characters
30  ! per data point and transferred into a real array in the controller. The
40  ! corresponding frequency data is calculated from the analyzer settings.
50  !
60  ! EXAMP3B Data Transfer Using LORT (ASCII Transfer)
70  !
80  ASSIGN @Nwa TO 716                    ! Assign an I/O path to the analyzer
90  !
100 CLEAR SCREEN
110 ! Initialize
120 ABORT 7                               ! Generate an IFC (Interface Clear)
130 CLEAR @Nwa                            ! SDC (Selective Device Clear)
140 OUTPUT @Nwa;"OPC?;PRES;"              ! Preset the analyzer
150 ENTER @Nwa;Reply                      ! Read the 1 when complete
160 !
170 ! Trace values are two elements per point, display format dependent
180 DIM Dat(1:11,1:2)                    ! Trace data array
190 !
```

```
200 OUTPUT @Nwa;"POIN 11;"                ! Set trace length to 11 points
210 OUTPUT @Nwa;"OPC?;SING;"              ! Single sweep mode and wait
220 ENTER @Nwa;Reply                      ! Read reply
230 !
240 OUTPUT @Nwa;"FORM4;"                  ! Set form 4 ASCII format
250 OUTPUT @Nwa;"OUTPFORM;"               ! Send formatted trace to controller
260 ENTER @Nwa;Dat(*)                     ! Read in data array from analyzer
270 !
280 ! Now to calculate the frequency increments between points
290 OUTPUT @Nwa;"POIN?;"                  ! Read number of points in the trace
300 ENTER @Nwa;Num_points
310 OUTPUT @Nwa;"STAR?;"                  ! Read the start frequency
320 ENTER @Nwa;Startf
330 OUTPUT @Nwa;"SPAN?;"                  ! Read the span
340 ENTER @Nwa;Span
350 !
360 F_inc=Span/(Num_points-1)            ! Calculate fixed frequency increment
370 !
380 PRINT "Point","Freq (MHz)","  Value 1","  Value 2"
390 IMAGE 3D,7X,5D.3D,3X,3D.4D,3X,3D.4D   ! Formatting for controller display
400 !
410 FOR I=1 TO Num_points                 ! Loop through data points
420   Freq=Startf+(I-1)*F_inc             ! Calculate frequency of data point
430   PRINT USING 390;I,Freq/1.E+6,Dat(I,1),Dat(I,2)! Print analyzer data
440 NEXT I
450 !
460 OUTPUT @Nwa;"MARKDISC;"               ! Discrete marker mode
470 OUTPUT @Nwa;"MARK1 3E+8;"             ! Position marker at 300 KHz
480 !
490 OUTPUT @Nwa;"OPC?;WAIT;"              ! Wait for the analyzer to finish
500 ENTER @Nwa;Reply                      ! Read the 1 when complete
510 LOCAL 7                               ! Release GPIB control
520 !
530 PRINT
540 PRINT "Position the marker with the knob and compare the values"
550 !
560 END
```

Example 3C: Data Transfer Using Floating-Point Numbers

This example program illustrates data transfer using FORM 3 in which data is transmitted in the floating-point formats. FORM 2 is nearly identical except for the IEEE 32-bit format of 4 bytes-per-value. FORM 5 reverses the order of the bytes to conform with the PC conventions for defining a real number.

The block-data formats have a four-byte header. The first two bytes are the ASCII characters "#A" that indicate that a fixed-length block transfer follows, and the next two bytes form an integer containing the number of bytes in the block to follow. The header must be read in so that data order is maintained.

This transfer is more than twice as fast than a FORM 4 transfer. With the FORM 4 transfer, 10,050 bytes are sent (201 points × 2 values-per-point × 24 bytes-per-value). Using FORM 2 to transfer the data, only 1612 bytes are sent (201 points × 2 values-per-point × 4 bytes-per-value). See "Array-Data Formats" on page 5-7.

The following is an outline of the program's processing sequence:

- An I/O path is assigned for the analyzer.
- The system is initialized.
- The integer variables are defined to contain the header information.
- The number of points in the trace is set to 11.
- The selected frequency span is swept once.
- Data-transfer format 3 is set.
- The headers are read from the trace.
- The array size is calculated and allocated.
- The trace data is read in and printed.
- The marker is activated and placed at the lowest frequency of the analyzer (50 MHz).
- The instrument is returned to local control and the program ends.

Running the Program

Run the program. The computer displays the number of elements and bytes associated with the transfer of the trace, as well as the first 10 data points. Position the marker and examine the data values. Compare the displayed values with the analyzer's marker values.

BASIC Program Listing

```
10  ! This program shows how to read in a data trace in IEEE 64 bit
20  ! format. The array header is used to determine the length of the
30  ! array and to allocate the array size.
40  !
50  ! Program Example 3C Data Transfer Using Floating-Point Numbers
60  !
70  CLEAR SCREEN
80  ! Initialize the analyzer
90  ASSIGN @Nwa TO 716                  ! Assign an I/O path for the analyzer
100 ASSIGN @Nwadat TO 716;FORMAT OFF    ! Binary data path definition
110 !
120 ABORT 7                             ! Generate an IFC ( Interface Clear)
130 CLEAR @Nwa                          ! SDC (Selected Device Clear)
140 OUTPUT @Nwa;"OPC?;PRES;"            ! Preset the analyzer and wait
150 ENTER @Nwa;Reply                    ! Read the 1 when completed
160 !
170 INTEGER Dheader,Dlength             ! Integer variables for header info
180 Numpoints=11                        ! Number of points in the trace
190 OUTPUT @Nwa;"POIN";Numpoints;";"    ! Set number of points in trace
200 !
210 !  Set up data transfer
220 OUTPUT @Nwa;"OPC?;SING"             ! Single sweep and wait
230 ENTER @Nwa;Reply                    ! Read the 1 when completed
240 !
250 OUTPUT @Nwa;"FORM3;"                ! Select form 3 format
260 OUTPUT @Nwa;"OUTPFORM;"             ! Send formatted output trace
270 !
280 ENTER @Nwadat;Dheader,Dlength       ! Read headers from trace data
```

```
290 !
300 ALLOCATE Dat(1:Dlength/16,1:2)       ! Use length to determine array size
310 ENTER @Nwadat;Dat(*)                 ! Read in trace data
320 !
330 PRINT “Size of array “;Dlength/16;” elements”
340 PRINT “Number of bytes “;Dlength
350 !
360 ! Print out the data array
370 PRINT “Element”,”Value 1”,”    Value 2”
380 IMAGE 3D,6X,3D.4D,6X,3D.4D
390 FOR I=1 TO Numpoints                 ! Loop through the data points
400   PRINT USING 380;I,Dat(I,1),Dat(I,2)
410 NEXT I
420 !
430 OUTPUT @Nwa;”MARKDISC;”              ! Discrete marker mode
440 OUTPUT @Nwa;”MARK1 3E+8;”            ! Position marker at 300 KHz
450 !
460 OUTPUT @Nwa;”OPC?;WAIT;”             ! Wait for the analyzer to finish
470 ENTER @Nwa;Reply                     ! Read the 1 when complete
480 LOCAL @Nwa                           ! Release GPIB control
490 !
500 PRINT
510 PRINT “Position the marker with the knob and compare the values.”
520 !
530 END
```

Example 3D: Data Transfer Using Frequency-Array Information

Example 3C was used to read in the trace-data array. Example 3D explains how to use the limit-test array to read the corresponding frequency values for the completed trace array into the controller. The analyzer is set to sweep from 50 MHz to 200 MHz in log-frequency mode with the number of points in the trace set to 11. This makes it very difficult to compute the frequency-point spacing in the trace. The points are equally spaced across the trace, but not equally spaced in relation to frequency (because the frequency span is displayed in a logarithmic scale, as opposed to a linear scale). The limit-test data array may be read from the analyzer to provide the frequency values for each data point. Four values are read for each data point on the analyzer. The test results and limit values are not used in this example. Only the frequency values are used. This technique is an effective method of obtaining the non-linear frequency data from the analyzer display. The test data and frequencies are printed on the controller display and the marker is enabled to allow the operator to examine the actual locations on the analyzer display.

The following is an outline of the program's processing sequence:

- An I/O path is assigned for the analyzer.
- The system is initialized.
- The integer variables for the header information are defined.
- The number of points in the trace is set to 11.
- The frequency span (50 MHz to 200 MHz) is selected.
- The log-frequency sweep is selected.

- The data-transfer format 3 is set.
- The headers are read from the trace.
- The array size is calculated and allocated.
- The trace data is read in.
- The limit-test array is calculated and allocated.
- The limit-line test array is read in.
- The table header is printed.
- The program cycles through the trace values.
- The trace data and frequency are printed.
- The discrete-marker mode is activated.
- The marker is activated and placed at the lowest frequency of the analyzer (50 MHz).
- The instrument is returned to local control and the program ends.

Running the Program

Run the program. Observe the controller display. The corresponding frequency values are shown with the trace-data values. Position the marker and observe the relationship between the frequency values and the point spacing on the trace. Compare the trace-data values on the analyzer with those shown on the controller display.

BASIC Program Listing

```
10  ! This program shows an ASCII format trace data transfer using form 4.
20  ! The data is received as a string of ASCII characters, 24 characters
30  ! per data point and transferred into a real array in the controller. The
40  ! corresponding frequency data is calculated from the analyzer settings.
50  !
60  ! EXAMP3B Data Transfer Using LORT (ASCII Transfer)
70  !
80  ASSIGN @Nwa TO 716                    ! Assign an I/O path to the analyzer
90  !
100 CLEAR SCREEN
110 ! Initialize
120 ABORT 7                               ! Generate an IFC (Interface Clear)
130 CLEAR @Nwa                            ! SDC (Selective Device Clear)
140 OUTPUT @Nwa;"OPC?;PRES;"              ! Preset the analyzer
150 ENTER @Nwa;Reply                      ! Read the 1 when complete
160 !
170 ! Trace values are two elements per point, display format dependent
180 DIM Dat(1:11,1:2)                    ! Trace data array
190 !
200 OUTPUT @Nwa;"POIN 11;"                ! Set trace length to 11 points
210 OUTPUT @Nwa;"OPC?;SING;"              ! Single sweep mode and wait
220 ENTER @Nwa;Reply                      ! Read reply
230 !
240 OUTPUT @Nwa;"FORM4;"                  ! Set form 4 ASCII format
250 OUTPUT @Nwa;"OUTPFORM;"               ! Send formatted trace to controller
260 ENTER @Nwa;Dat(*)                     ! Read in data array from analyzer
270 !
```

```
280 ! Now to calculate the frequency increments between points
290 OUTPUT @Nwa;"POIN?;"                  ! Read number of points in the trace
300 ENTER @Nwa;Num_points
310 OUTPUT @Nwa;"STAR?;"                  ! Read the start frequency
320 ENTER @Nwa;Startf
330 OUTPUT @Nwa;"SPAN?;"                  ! Read the span
340 ENTER @Nwa;Span
350 !
360 F_inc=Span/(Num_points-1)            ! Calculate fixed frequency increment
370 !
380 PRINT "Point","Freq (MHz)","  Value 1","  Value 2"
390 IMAGE 3D,7X,5D.3D,3X,3D.4D,3X,3D.4D   ! Formatting for controller display
400 !
410 FOR I=1 TO Num_points                 ! Loop through data points
420   Freq=Startf+(I-1)*F_inc             ! Calculate frequency of data point
430   PRINT USING 390;I,Freq/1.E+6,Dat(I,1),Dat(I,2)! Print analyzer data
440 NEXT I
450 !
460 OUTPUT @Nwa;"MARKDISC;"               ! Discrete marker mode
470 OUTPUT @Nwa;"MARK1 3E+8;"             ! Position marker at 300 KHz
480 !
490 OUTPUT @Nwa;"OPC?;WAIT;"              ! Wait for the analyzer to finish
500 ENTER @Nwa;Reply                      ! Read the 1 when complete
510 LOCAL 7                               ! Release GPIB control
520 !
530 PRINT
540 PRINT "Position the marker with the knob and compare the values"
550 !
560 END
```

Example 3E: Data Transfer Using FORM 1 (Internal-Binary Format)

FORM 1 is used for rapid I/O transfer of analyzer data. It contains the least number of bytes-per-trace and does not require re-formatting in the analyzer. This format is more difficult to convert into a numeric array in the controller.

The following is an outline of the program's processing sequence:

- An I/O path is assigned for the analyzer.
- The system is initialized.
- The integer variables for the header information are defined.
- The string variable for the header is defined.
- The selected frequency span is swept once.
- The internal-binary format is selected.
- The error-corrected data is output from the analyzer.
- The two data-header characters and the two length bytes are read in.
- The string buffer is allocated for data.
- The trace data is read into the string buffer.
- The analyzer is restored to continuous-sweep mode and queried for command completion.

- The instrument is returned to local control and the program ends.

Running the Program

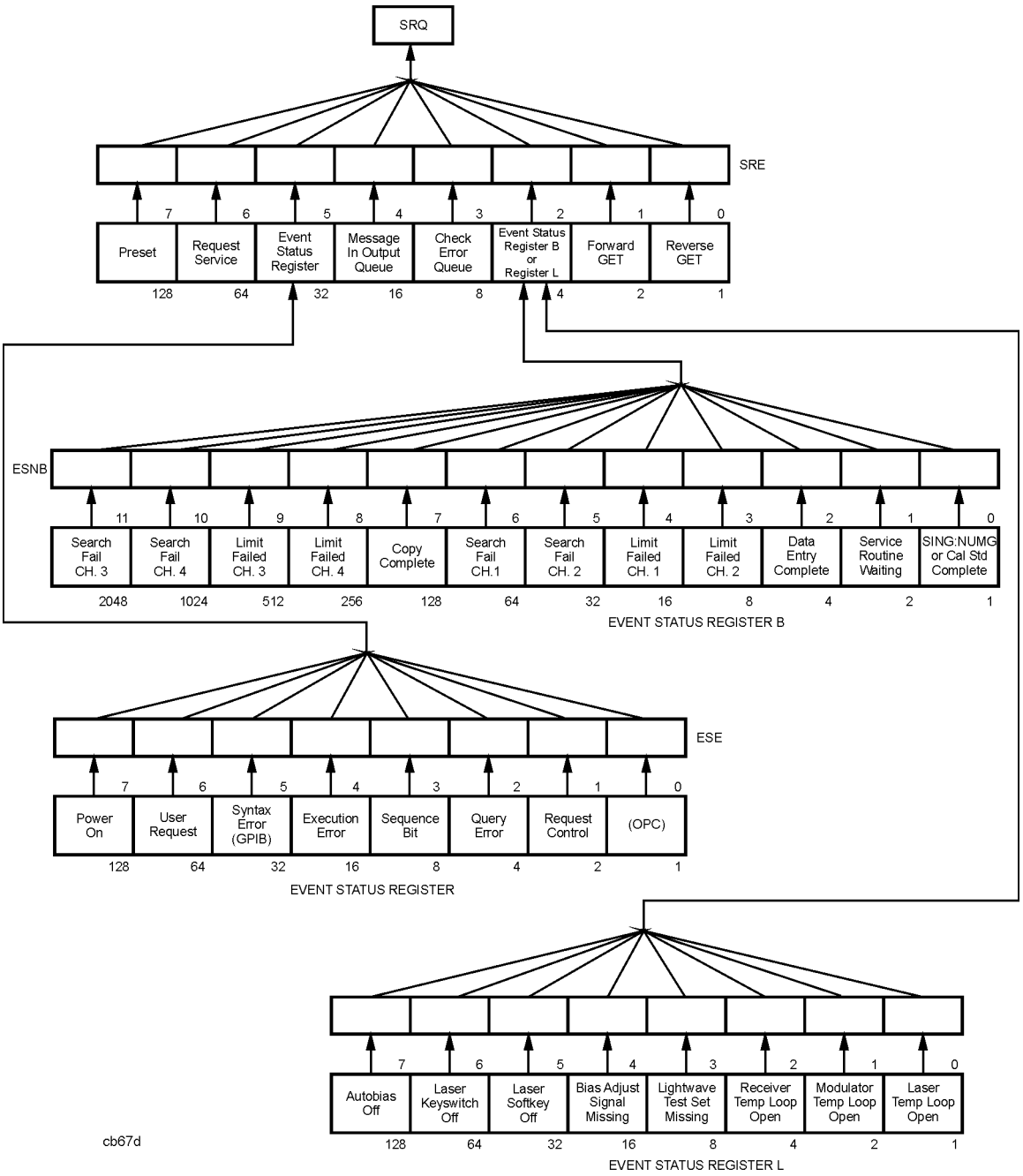
The analyzer is initialized. The header and the number of bytes in the block transfer are printed on the controller display. Once the transfer is complete, the number of bytes in the data string is printed. Compare the two numbers to be sure that the transfer was completed.

BASIC Program Listing

```
10  ! This program is an example of a form 1, internal format data
20  ! transfer. The data is stored in a string dimensioned to the
30  ! length of the data being transferred.
40  !
50  ! EXAMP3E Data Transfer Using FORM 1 (Internal Binary Format)
60  !
70  ASSIGN @Nwa TO 716                    ! Assign an I/O path for the analyzer
80  ASSIGN @Nwa_bin TO 716;FORMAT OFF     ! Binary path for data transfer
90  !
100 CLEAR SCREEN
110 ! Initialize the analyzer
120 ABORT 7                               ! Send IFC Interface Clear
130 CLEAR @Nwa                            ! SDC (Selective Device Clear)
140 OUTPUT @Nwa;"OPC?;PRES;"              ! Preset the analyzer and wait
150 ENTER @Nwa;Reply                      ! Read the 1 when completed
160 !
170 INTEGER Length                        ! Header length 2 bytes
180 DIM Header$[2]                        ! Header string 2 bytes
190 !
200 OUTPUT @Nwa;"OPC?;SING;"              ! Single sweep and wait
210 ENTER @Nwa;Reply                      ! Read the 1 when completed
220 !
230 OUTPUT @Nwa;"FORM1;"                  ! Select internal binary format
240 OUTPUT @Nwa;"OUTPDATA;"               ! Output error corrected data
250 !
260 ! Read in the data header two characters and two bytes for length
270 ! "#,2A"
280 !      # no early termination, terminate when ENTER is complete
290 !      2A read two chars
300 !
310 ENTER @Nwa_bin USING "#,2A";Header$   ! Read header as 2 byte string
320 ENTER @Nwa_bin;Length                 ! Read length as 2 byte integer
330 PRINT "Header ";Header$,"Array length";Length
340 !
350 ALLOCATE Data$[Length]                ! String buffer for data bytes
360 ! "+,-K" format statement
370 ! + EOI as a terminator LF is suppressed and read as data
380 ! -K All characters are read and not interpreted LF is included
390 ENTER @Nwa_bin USING "+,-K";Data$     ! Read trace into string array
400 !
410 PRINT "Number of bytes received ";LEN(Data$)
420 !
430 OUTPUT @Nwa;"CONT;"                   ! Restore continuous sweep
440 OUTPUT @Nwa;"OPC?;WAIT;"              ! Wait for the analyzer to finish
450 ENTER @Nwa;Reply                      ! Read the 1 when complete
460 !
470 LOCAL @Nwa                            ! Release GPIB control
480 END
```

Measurement Process Synchronization Examples

Figure 8-1. Status Reporting Structure



Status Reporting

The analyzer has a status reporting mechanism, illustrated in Figure 8-1, that provides information about specific analyzer functions and events. The status byte is an 8-bit register

with each bit summarizing the state of one aspect of the instrument. For example, the error queue summary bit will always be set if there are any errors in the queue. The value of the status byte can be read with the GPIB serial poll operation. This command does not automatically put the instrument in remote mode, which gives you access to the analyzer front-panel functions. The status byte can also be read by sending the command OUTPSTAT. Reading the status byte does not affect its value.

The status byte summarizes the error queue, as mentioned before. It also summarizes two event-status registers that monitor specific conditions inside the instrument. The status byte also has a bit (6) that is set when the instrument is issuing a service request over GPIB, and a bit (0) that is set in the event-status register when the analyzer has data to send out over GPIB. See "Error Reporting" on page 7-2 for a discussion of the event-status registers.

Example 4A: Using the Error Queue

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

The following is an outline of the program's processing sequence:

- An I/O path is assigned for the analyzer.
- The system is initialized.
- The error-message string is allocated.
- The analyzer is released from remote control.
- The program begins an endless loop to read the error queue.
- The status byte is read with a serial poll.
- The program tests to see if an error is present in the queue.
- The error-queue bit is set.
- The program requests the content of the error queue.
- The error number and string are read.
- The error messages are printed until there are no more errors in the queue.
- The instrument is returned to local control.
- The controller emits a beep to attract the attention of the operator and resumes searching for errors.

Running the Program

Run the program. The analyzer goes through the preset cycle. Nothing will happen at first. The program is waiting for an error condition to activate the error queue. To cause an error, press a blank softkey. The message CAUTION: INVALID KEY will appear on the analyzer. The computer will beep and print out two error messages. The first line will be the invalid key error message, and the second line will be the NO ERRORS message. To clear the error queue,

you can either loop until the NO ERRORS message is received, or wait until the bit in the status register is cleared. In this case, we wait until the status bit in the status register is clear. Note that while the program is running, the analyzer remains in the local mode and the front-panel keys may be accessed.

The error queue will hold up to 20 errors until all the errors are read out or the instrument is preset. It is important to clear the error queue whenever errors are detected. Otherwise, old errors may be mistakenly associated with the current instrument state.

Press **System,** and then the unlabeled key several times quickly and watch the display. The number of errors observed should correspond to the number of times you pressed the key.

As another example, press **Cal, CORRECTION ON.** A complete list of error messages and their descriptions can be found in your analyzer's reference guide.

The program is in an infinite loop waiting for errors to occur. End the program by pressing **Reset,** or **Break,** on the controller keyboard.

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

BASIC Program Listing

```
10  ! This program is an example of using the error queue to detect
20  ! errors generated by the analyzer. The status byte is read and
30  ! bit 3 is tested to determine if an error exists. The error queue
40  ! is printed out and emptied.
50  !
60  ! EXAMP4A Using the Error Queue
70  !
80  ASSIGN @Nwa TO 716                  ! Assign an I/O path for the analyzer
90  !
100 CLEAR SCREEN
110 ! Initialize the analyzer
120 ABORT 7                             ! Generate an IFC (Interface Clear)
130 CLEAR @Nwa                          ! SDC (Selective Device Clear)
140 OUTPUT @Nwa;"OPC?;PRES;"            ! Preset the analyzer and wait
150 ENTER @Nwa;Reply                    ! Read the 1 when complete
160 !
170 DIM Error$[50]                      ! String for analyzer error message
180 !
190 LOCAL @Nwa                          ! Release analyzer from remote control
200 !
210 LOOP                                ! Endless loop to read error queue
220   REPEAT
230     Stat=SPOLL(@Nwa)                ! Read status byte with serial poll
240   UNTIL BIT(Stat,3)                 ! Test for error queue present
250   !
260 !                 Error queue bit is set
270   REPEAT                            ! Loop until error number is 0
280     OUTPUT @Nwa;"OUTPERRO;"         ! Request error queue contents
290     ENTER @Nwa;Err,Error$           ! Read error number and string
300     PRINT Err,Error$                ! Print error messages
310   UNTIL Err=0                       ! No more errors in queue
320 !
330   LOCAL @Nwa                        ! Release analyzer from remote
```

```
340   BEEP 600,.2                          ! Beep to attract attention
350 END LOOP                               ! Repeat error search
360 !
370 END
```

Example 4B: Generating Interrupts

It is also possible to generate interrupts using the status-reporting mechanism. The status-byte bits can be enabled to generate a service request (SRQ) when set. In turn, the instrument controller can be set up to generate an interrupt on the SRQ and respond to the condition which caused the SRQ.

To generate an SRQ, a bit in the status byte is enabled using the command SREn. A one (1) in a bit position enables that bit in the status byte. Hence, SRE 8 enables an SRQ on bit 3, the check-error queue, since the decimal value 8 equals 00001000 in binary representation. Whenever an error is put into the error queue and bit 3 is set, the SRQ line is asserted, illuminating the (S) indicator in the GPIB status block on the front panel of the analyzer. The only way to clear the SRQ is to disable bit 3, re-enable bit 3, or read out all the errors from the queue.

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

To generate an SRQ from an event-status register, enable the desired event-status register bit. Then enable the status byte to generate an SRQ. For instance, ESE 32;SRE 32; enables the syntax-error bit. When the syntax-error bit is set, the summary bit in the status byte will be set. This will, in turn, enable an SRQ on bit 5 of the status byte, the summary bit for the event-status register.

The following is an outline of the program's processing sequence:

- An I/O path is assigned for the analyzer.
- The system is initialized.
- The status registers are cleared.
- The event-status register bit 5 is enabled.
- The status-register bit 5 is enabled.
- The interrupt pointer is enabled and points to a subroutine.
- Two bad commands are sent to the analyzer to generate errors.
- The controller reads a serial-poll byte from GPIB in the event of an interrupt.
- The program tests for an SRQ.
- If the SRQ is not generated by the analyzer, the subroutine stops and displays SRQ FROM OTHER DEVICE.

- If the SRQ was generated by the analyzer, the program reads the status byte and event-status register.
- If bit 5 in the event-status register is set, the program prints: SYNTAX ERROR FROM ANALYZER.
- If bit 5 in the event-status register is *not* set, the program prints: SYNTAX ERROR BIT NOT SET.
- The SRQ interrupt is re-enabled on the bus.
- At the finish, the interrupt is deactivated.
- The analyzer is released from remote control and the program ends.

Running the Program

Run the program. The computer will preset the analyzer, then pause for a second or two. After pausing, the program sends an invalid command string "STIP 2 GHZ;" to cause a syntax error. This command is intended to be "STOP 2 GHZ;". The computer will display a series of messages from the SRQ-handler routine. The analyzer will display CAUTION: SYNTAX ERROR and the incorrect command, pointing to the first character it did not understand.

The SRQ can be cleared by reading the event-status register and clearing the latched bit, or by clearing the enable registers with CLES. The syntax-error message on the analyzer display can only be cleared by the GPIB Device Clear (DCL) message or Selected Device Clear (SDC) message. Device Clear is not commonly used because it clears every device on the bus. Selected Device Clear can be used to reset the input and output queue and the registers of a specific instrument on the bus. This will also clear all the interrupt definitions.

BASIC Program Listing

```
10  ! This program is an example of using an SRQ based interrupt to
20  ! detect an error condition in the analyzer. In this example, a
30  ! syntax error is generated with an invalid command. The status byte
40  ! is read in and tested. The error queue is read, printed out and
50  ! then cleared.
60  !
70  ! EXAMP4B Generating Interrupts
80  !
90  ASSIGN @Nwa TO 716                  ! Assign an I/O path for the analyzer
100 !
110 CLEAR SCREEN
120 ! Initialize the analyzer
130 ABORT 7                             ! Generate and IFC (Interface Clear)
140 CLEAR @Nwa                          ! SDC (Selective Device Clear)
150 OUTPUT @Nwa;"OPC?;PRES;"            ! Preset the analyzer and wait
160 ENTER @Nwa;Reply                    ! Read the one from the analyzer
170 !
180 DIM Error$[50]                      ! String for analyzer error message
190 ! Set up syntax error interrupt
200 OUTPUT @Nwa;"CLES;"                 ! Clear the status registers
210 !
220 ! Generate SRQ when bit 5 is set
230 OUTPUT @Nwa;"ESE 32;"               ! Event status register bit 5 enabled
240 !
```

```
250 ! Generate bit 5 in status register when syntax error occurs
260 OUTPUT @Nwa;"SRE 32;"               ! Status register bit 5 enabled
270 !
280 ! Setup the interrupt pointer to a subroutine
290 ON INTR 7 GOSUB Srq_det             ! When interrupt occurs go to Srq_det
300 Stat=SPOLL(@Nwa)                    ! Clear any pending SRQs
310 ENABLE INTR 7;2                     ! Set interrupt on GPIB bit 2 (SRQ)
320 !
330 DISP "Waiting for bad syntax"
340 WAIT 2                              ! Pause for 2 seconds
350 !
360 OUTPUT @Nwa;"STIP 2GHZ;;"           ! Send bad STOP command syntax
370 !
380 WAIT 2                              ! Pause for 2 seconds
390 DISP ""                             ! Clear display line
400 GOTO Finish                         ! Exit program example
410 !
420 !************************* Subroutines *****************************
430 !
440 Srq_det:                            ! SRQ handler
450 Stat=SPOLL(@Nwa)                    ! Read serial poll byte from GPIB
460 PRINT "Stat from Serial Poll";Stat
470 IF BIT(Stat,6) THEN                 ! Test for SRQ
480   PRINT "SRQ received from analyzer"
490 ELSE                                ! No SRQ from analyzer
500   PRINT "SRQ from other device"
510   STOP                              ! Stop if not from analyzer
520 END IF
530 !
540 IF BIT(Stat,5) THEN                 ! Event status register bit set
550   PRINT "Event Status Register caused SRQ"
560 ELSE                                ! Some other bit set
570   PRINT "Some other bit caused the SRQ"
580   STOP                              ! Stop if bit not set
590 END IF
600   !
610 REPEAT
620   OUTPUT @Nwa;"OUTPERRO;"           ! Read analyzer error queue
630   ENTER @Nwa;Err,Error$             ! Read error number and string
640   PRINT Err,Error$                  ! Print error message
650 UNTIL Err=0                         ! No more errors in queue
660 !
670 PRINT                               ! White space
680 ENABLE INTR 7;2                     ! Re-enable SRQ interrupt on GPIB
690 RETURN
700 !
710 !************************* End Subroutines *****************************
720 !
730 Finish:                             ! End of program and exit
740 DISP "Finished"
750 OFF INTR 7                          ! Turn off interrupt
760 LOCAL @Nwa                          ! Release GPIB control
770 END
```

Example 4C: Power Meter Calibration

NOTE This example program will not work with BASIC for Windows.

For increased accuracy of the analyzer's PORT 1-output power, a power meter calibration is available. This measurement-accuracy enhancement technique is described in your analyzer's user's guide. The example described will perform the sample and sweep calibration under GPIB remote control.

The power meter is usually connected to PORT 1 for the forward measurements. Its address must be set correctly and it must be connected to the GPIB. The power meter address can be set by pressing: **Local, SET ADDRESSES, ADDRESS P MTR/GPIB,** and using the **up,** and **down,** keys or the numeric key pad to complete the process. The appropriate command must be selected for the model number of power meter being used. Press **POWER MTR: [ ],** until the model being used is displayed between the brackets.

The correction factors for the power sensor are entered into the analyzer. All of these steps are explained in your analyzer's user's guide.

The number of readings-per-point must also be selected before starting. The number of points directly affects the measurement time of the calibration sequence. The power meter must be triggered and read by the analyzer for each trace point. Typically, two readings-per-point is considered appropriate. More than two readings-per-point could lead to unacceptable processing time.

To control a power meter calibration via GPIB, the analyzer must be set to pass-control mode. The analyzer must step to the next point in the sweep and read the power present at the power meter sensor. For this operation to take place, the system controller must set up the measurement and then pass control to the analyzer to read each data point in the sweep. After reading the data point from the power meter, the analyzer passes control back to the system controller. The analyzer then sets up to measure the next point and again requests control from the system controller. This process continues until the analyzer signals that the entire sweep has been measured point-by-point.

The following is an outline of the program's processing sequence:

- An I/O path is assigned for the analyzer.
- The system is initialized.
- The number of points in the trace is set.
- The number of readings-per-point is set.
- The frequency span is set.

NOTE The frequency span of this example program may need to be modified in order to correspond to the frequency ranges of your analyzer.

- The reference channel is measured.
- The power meter calibration array is allocated.
- The power meter model is chosen.
- The status registers are cleared.
- The request-control summary bit is enabled.
- The pass-control mode is enabled.
- A calibration sweep is taken to begin the sequence.
- The status byte is read until control is requested.
- The computer passes control to the analyzer.
- The display is cleared and the analyzer is set to talker/listener mode.
- The GPIB interface status is read until control is returned.
- The program loops until all the points have been measured.
- The power meter calibration is enabled.
- The calibration data is output to the controller in FORM 4, ASCII format.
- The power meter-calibration factors are read into the controller.
- The analyzer is released from remote control and the program ends.

Running the Program

The analyzer is preset and the power meter-calibration routine begins. The analyzer displays the message "WAITING FOR GPIB CONTROL" when it is requesting control. The system controller display prints "Passing Control" when control is passed to the analyzer. The controller displays "Waiting for request" while the analyzer has control and is reading the power meter.

The interaction of the messages and the movement of the cursor allow observation of the calibration process. Once the calibration is complete, the analyzer displays "POWER METER CAL IS COMPLETE" and the system controller displays "Finished with Power meter Cal".

The power meter-calibration mode (with one sweep of correction data) is enabled and the calibration is switched on. At the completion of the program, talker/listener mode is restored, the event-status registers are cleared (to halt the status-byte interaction), the power meter correction factors are displayed, the sweep is placed in continuous-sweep mode, the analyzer is released from GPIB control, and the program ends.

BASIC Program Listing

```
10  ! This routine does a power meter cal using pass control.
20  ! A measurement cycle takes place on each point of the trace. The
30  ! point is measured by the power meter and the measured value read
40  ! into the analyzer. The command TAKCS; arms this measurement mode.
50  ! The number of measurements is determined by the number of points in
60  ! the trace, the number of readings per point and an extra measurement
70  ! cycle to release the powr meter.
80  ! Control is passed to the analyzer, the point is measured and
90  ! the data is transferred to the analyzer. Control is passed back to
100 ! the controller and the cycle begins again. Serial poll is used to
110 ! read the status byte of the analyzer and test the logic.
120 ! The GPIB interface status register is monitored to determine when
130 ! control is returned to the interface from the analyzer.
140 !
150 ! EXAMP4C Power Meter Calibration
160 !
170 ASSIGN @Nwa TO 716                  ! Assign an I/O path for the analyzer
180 !
190 CLEAR SCREEN
200 ! Initialize the analyzer
210 ABORT 7                             ! Generate an IFC (Interface Clear)
220 CLEAR @Nwa                          ! SDC (Selective Device Clear)
230 OUTPUT @Nwa;"OPC?;PRES;"            ! Preset the analyzer and wait
240 ENTER @Nwa;Reply                    ! Read the 1 when complete
250 !
260 INTEGER Stat
270 !
280 ! Set up the analyzer parameters
290 Numpoints=11                        ! Number of points in the trace
300 Numreads=2                          ! Number of readings per point
310 Startf=1.00E+8                      ! Start frequency
320 Stopf=5.0E+8                        ! Stop frequency
330 !
340 OUTPUT @Nwa;"POIN";Numpoints;";"    ! Set trace length to numpoints
350 OUTPUT @Nwa;"NUMR";Numreads;";"     ! Set number of readings per point
360 OUTPUT @Nwa;"STAR";Startf           ! Set start frequency
370 OUTPUT @Nwa;"STOP";Stopf            ! Set stop frequency
380 OUTPUT @Nwa;"MEASR;"                ! Measure the reference channel
390 !
400 ALLOCATE Pmcal(1:Numpoints)         ! Create power meter cal array
410 !
420 ! Store the original trace for comparison
430 OUTPUT @Nwa;"DATI;"
440 OUTPUT @Nwa;"DISPDATM;"
450 OUTPUT @Nwa;"AUTO;"
460 !
470 ! Select the power meter being used for cal
480 ! OUTPUT @Nwa;"POWM ON;"            ! Select 436A power meter
490 OUTPUT @Nwa;"POWMOFF;DEBUON;"       ! Select 437B/438A power meter
500 !
510 ! Set analyzer GPIB, status regs to interrupt on pass control
520 OUTPUT @Nwa;"CLES;"                 ! Clear status registers
530 OUTPUT @Nwa;"ESE2;"                 ! Enable request control summary bit
540 OUTPUT @Nwa;"SRE32;"                ! SRQ on events status register
550 !
560 PRINT "Beginning Power Meter CAL"
```

```
570 OUTPUT @Nwa;"USEPASC;"              ! Enable pass control operation
580 OUTPUT @Nwa;"TAKCS;"                ! Take Cal Sweep
590 !
600 FOR I=1 TO Numpoints*Numreads+1     ! Points * Number of readings plus 1
610   ! Serial poll does not place analyzer in remote operation
620   ! and does not require the analyzer to process the command.
630   !
640   REPEAT                            ! Repeat until SRQ detected
650     Stat=SPOLL(@Nwa)                ! Serial poll to read status byte
660     DISP "Stat ";Stat;" Waiting for request"
670   UNTIL BIT(Stat,6)                 ! SRQ detected for request control
680   OUTPUT @Nwa;"ESR?;"               ! Read status register to clear
690   ENTER @Nwa;Reply                  ! Read and discard register value
700   !
710   PRINT "Passing Control"           ! status read and pasing control
720   PASS CONTROL @Nwa                 ! Pass control to analyzer
730   !
740   REPEAT
750   ! Read GPIB interface state information register.
760     STATUS 7,6;Gpib                 ! Test GPIB register for control
770   !
780   ! Reading the interface status register does not interact with the
790   ! analyzer. Bit 6 is set when control is returned.
800   !
810     DISP "Waiting for control"
820   UNTIL BIT(Gpib,6)                 ! Loop until control is returned
830 NEXT I
840 !
850 PRINT "Finished with Power meter Cal"
860 DISP ""                             ! Clear display message
870 !
880 OUTPUT @Nwa;"TALKLIST;"             ! Restore Talker/Listener operation
890 OUTPUT @Nwa;"CLES;"                 ! Clear and reset status byte operation
900 !
910 OUTPUT @Nwa;"PWMCONES;"             ! Power meter cal correct one sweep
920 OUTPUT @Nwa;"OPC?;WAIT;"            ! Wait for the analyzer to finish
930 ENTER @Nwa;Reply                    ! Read the 1 when complete
940 !
950 ! Read the power meter cal correction factors
960 OUTPUT @Nwa;"FORM4;"                ! ASCII data format to read cal data
970 OUTPUT @Nwa;"OUTPPMCAL1;"           ! Request the power meter cal factors
980 ENTER @Nwa;Pmcal(*)                 ! Read the factors
990 !
1000! Display the power meter cal factors
1010 PRINT "Point","Factor"
1020 FOR I=1 TO Numpoints               ! Cycle throught the factors
1030 PRINT I,Pmcal(I)
1040 NEXT I
1050!
1060 CLEAR @Nwa                         ! SDC (Selective Device Clear)
1070 LOCAL @Nwa                         ! Release GPIB control
1080 END
```

Analyzer System Setup Examples

Saving and Recalling Instrument States

NOTE The most efficient option for storing and recalling analyzer states is using the analyzer's internal registers to save the CAL data. Recalling these registers is the fastest solution to restoring analyzer setups. See your analyzer's user's guide for detailed information on the analyzer's internal storage registers.

In the event that all the registers have been used, the internal disk drive is not used, or if internal memory limitations exist, then these external solutions become viable.

The purpose of this example is to demonstrate several programming options for storing and recalling entire instrument states over GPIB. The examples describe two different processes for storing and recalling instrument states. The first example accomplishes the task using the learn string. The second example involves reading both the learn string and the calibration arrays out of the analyzer and storing them to disk or storing them in the system controller itself.

Using the learn string is a very rapid way of saving the instrument state, but using direct disk access has the advantage of automatically storing calibrations, cal kits, and data along with the instrument state.

A complete analyzer setup requires sending the learn string and a calibration array to set the analyzer parameters. The CAL array may also be placed in the analyzer, just as if a calibration was performed. By sending both sets of data, the analyzer may be quickly setup for a measurement.

Several different measurements may be required in the course of testing a device. An efficient way of performing multiple measurements is to send both the calibration array and the learn string, and then perform the measurements.

Example 5A: Using the Learn String

The learn string is a very fast and easy way to read an instrument state. The learn string includes all front-panel settings, the limit table for each channel, and the list-frequency table. It can be read out of the analyzer with the command `OUTPLEAS`, and input to the analyzer with the command `INPULEAS`.

The following is an outline of the program's processing sequence:

- An I/O path is assigned for the analyzer.
- The system is initialized.
- The string storage is allocated.
- The learn string is requested.
- The string is read without any processing.

- The analyzer is released from remote control.
- The instrument state is changed by the operator.
- The learn string is sent back to the analyzer.
- The analyzer is released from remote control and the program ends.

Running the Program

Run the program. When the program stops, change the instrument state and press **Enter,** on the controller. The analyzer will be returned to its original state by using the learn string.

BASIC Program Listing

```
1   ! This program shows how to retrieve a learn string from the analyzer
2   ! into a string array. The state of the analyzer is then changed and the
3   ! learn string re-loaded to return the analyzer to the previous settings.
4   !
5   ! EXAMP5A Using the Learn String
6   !
7 OPTION BASE 1
8 ASSIGN @Nwa TO 716                    ! Assign an I/O path for the analyzer
9 ASSIGN @Nwa_bin TO 716;FORMAT OFF
10  !
11 CLEAR SCREEN
12  ! Initialize the analyzer
13 ABORT 7                              ! Generate an IFC (Interface Clear)
14 CLEAR @Nwa                           ! SDC (Selected Device Clear)
15  !
16 INTEGER Header,Length                ! 2-byte header and length of string
17  !
18 OUTPUT @Nwa;"OUTPLEAS;"              ! Output the learn string
19 ENTER @Nwa_bin;Header,Length         ! Read header and length first
20  !
21 ALLOCATE INTEGER State(Length/2)     ! Integer array to contain the string
22  !
23 ENTER @Nwa_bin;State(*)              ! Read the string
24 LOCAL @Nwa                           ! Release GPIB control
25  !
26 INPUT "Change state and press ENTER",A$
27  !
28 OUTPUT @Nwa;"INPULEAS;"              ! Send the learnstring to analyzer
29 OUTPUT @Nwa_bin;Header,Length,State(*)
30 DISP "Analyzer state has been restored!"
31  !
32 OUTPUT @Nwa;"OPC?;WAIT;"             ! Wait for the analzyer to finish
33 ENTER @Nwa;Reply                     ! Read the 1 when complete
34 LOCAL @Nwa                           ! Release GPIB control
35 END
```

Example 5B: Reading Calibration Data

This example demonstrates:

- how to read measurement calibration data out of the analyzer
- how to read it back into the analyzer

- how to determine which calibration is active

The data used to perform measurement-error correction is stored inside the analyzer in one (or more) of twelve calibration-coefficient arrays. Each array is a specific error coefficient, and is stored and transmitted as an error-corrected data array. Each point is a real/imaginary pair, and the number of points in the array is the same as the number of points in the sweep. The five data formats also apply to the transfer of calibration-coefficient arrays. Your analyzer's user's guide contains information on the storage locations for calibration coefficients and different calibration types.

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

A computer can also store calibration coefficients in the analyzer. To do this, declare the type of calibration data about to be stored in the analyzer just as if you were about to perform that calibration. Then, instead of calling up different classes, transfer the calibration coefficients using the `INPUCALCnn;` commands. The variables `nn` are a data pair appended to the command representing a calibration number from `01` through `12`. When all the coefficients are stored in the analyzer, activate the calibration by issuing the mnemonic `SAVC;`, and trigger a sweep on the analyzer.

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

The following is an outline of the program's processing sequence:

- An I/O path is assigned for the analyzer.
- A binary path is assigned.
- The system is initialized.
- The calibration types and number of arrays are defined.
- The integer variables for reading the headers are defined.
- The calibration type and number of arrays are read by the controller.
- The output is formatted in FORM 3.
- The number of points in the trace is read.
- The memory is allocated for the calibration arrays.
- Each calibration array is requested from the analyzer.
- Header information is read with a binary I/O path.
- The elements from each calibration array are read in.
- The next calibration array is requested until all the arrays have been read.
- The calibration type is sent to the analyzer.
- Each calibration array is sent.

- The calibration is activated.
- The analyzer is released from remote control and the program ends.

Running the Program

Before executing the program, perform a calibration.

The program is able to detect which type of calibration is active. With that information, it predicts how many arrays to read out. When all the arrays have been sent to the computer, the program prompts the user. The operator then turns the calibration off or performs a completely different calibration on the analyzer and continues the program. The computer reloads the old calibration. The operator should not preset the analyzer because the instrument settings must be the same as those that were present when the calibration was taken.

NOTE The retransmitted calibration is associated with the current instrument state: the instrument has no way of knowing the original state associated with the calibration data. For this reason, it is recommended that the learn string be used to store the instrument state whenever calibration data is stored. The next example demonstrates how to reload the analyzer state with both the learn string and the calibration arrays.

BASIC Program Listing

```
10  ! This program shows how to manipulate calibration data from the analyzer.
20  ! It demonstrates how to read calibration data from the analyzer, and
30  ! how to replace it. The type of calibration active is determined and
40  ! the program reads in the correct number of arrays. The number of points
50  ! in the trace, and in the cal array, is determined and used to dimension
60  ! storage arrays.
70  !
80  ! EXAMP5B Reading the Calibration Data
90  !
100 ASSIGN @Nwa TO 716                    ! Assign an I/O path for the analyzer
110 ASSIGN @Nwa_bin TO 716;FORMAT OFF     ! Assign binary path
120 !
130 CLEAR SCREEN
140 ! Initialize the analyzer
150 ABORT 7                               ! Generate an IFC (Interface Clear)
160 CLEAR @Nwa                            ! SDC (Selected Device Clear)
170 !
180 ! Data for determining CAL type and number of arrays
190 DATA “CALIRESP”,1,”CALIRAI”,2,”CALIS111”,3
200 DATA “CALIS221”,3,”CALIFUL2”,12,”CALIEORM”,7,”CALIOERM”,10
210 DATA “NOOP”,0
220 !
230 INTEGER Hdr,Lgth,I,J                  ! Integers for reading headers
240 !
250 READ Calt$,Numb                       ! Read CAL type from data statement
260 IF Numb=0 THEN GOTO 690               ! If no CAL type is present Exit
270 OUTPUT @Nwa;Calt$;”?;”                ! Query if CAL type is active
280 ENTER @Nwa;Active                     ! Read 1 if active
290 IF NOT Active THEN GOTO 250           ! Load another CAL type and re-try
300 !
```

```
310 PRINT Calt$,Numb                        ! Active CAL and number of arrays
320 !
330 OUTPUT @Nwa;"FORM3;"                    ! Form 3 IEEE 64 bit floating point
340 OUTPUT @Nwa;"POIN?;"                    ! Request trace length
350 ENTER @Nwa;Poin                         ! Read number of points
360 ALLOCATE Cal(1:Numb,1:Poin,1:2)         ! Arrays for CAL arrays
370 !         Number of arrays, number of points real and imag value per point
380 !
390 FOR I=1 TO Numb                                ! Read arrays
400   OUTPUT @Nwa USING "K,ZZ";"OUTPCALC",I        ! Format I to add 0 in command
410   ENTER @Nwa_bin;Hdr,Lgth                      ! Read header & length from array
420   FOR J=1 TO Poin                              ! Read elements for CAL array
430     ENTER @Nwa_bin;Cal(I,J,1),Cal(I,J,2)       ! Read real & imag pair elements
440   NEXT J                                       ! Next location in array
450 NEXT I                                         ! Next CAL array
460 !
470 ! All CAL arrays have been read
480 !
490 INPUT "PRESS RETURN TO RE-TRANSMIT CALIBRATION",Dum$
500 !
510 OUTPUT @Nwa;"FORM3;"                           ! Use same format as read
520 OUTPUT @Nwa;Calt$;";"                          ! Send CAL type to analyzer
530 !
540 FOR I=1 TO Numb                                ! Send each array in CAL
550   DISP "TRANSMITTING ARRAY: ",I                ! Show array number
560   OUTPUT @Nwa USING "K,ZZ";"INPUCALC",I        ! Send array number 0 format
570   OUTPUT @Nwa_bin;Hdr,Lgth                     ! Send header & array length
580   FOR J=1 TO Poin                              ! Send each array element
590     OUTPUT @Nwa_bin;Cal(I,J,1),Cal(I,J,2)      ! Real and Imag pair
600   NEXT J                                       ! Next element in array
610 NEXT I                                         ! Next array
620 !
630 OUTPUT @Nwa;"SAVC;"                            ! Activate CAL
640 !
650 OUTPUT @Nwa;"CONT;"                            ! Restore continuous sweep
660 OUTPUT @Nwa;"OPC?;WAIT;"                       ! Wait for analyzer to finish
670 ENTER @Nwa;Reply                               ! Read the 1 when complete
680 !
690 DISP "Finished with CAL transfer"
700 LOCAL @Nwa                                     ! Release GPIB control
710 END
```

Example 5C: Saving and Restoring the Analyzer Instrument State

NOTE The instrument state may also be stored in the analyzer's internal registers. This is the fastest and most efficient method for toggling between instrument states. This example is for use when the analyzer's internal memory is full, or when the are other internal-memory limitations.

This example demonstrates how to use both the learn string and the calibration arrays to completely re-program the analyzer state. If you were performing two entirely different measurements on a device and wanted to quickly change between instrument states and perform the measurements, this example program is a potential solution.

The example will request the learn string and calibration array from the analyzer and store them in a disk file on the system controller. Once the storage is complete, the operator will be prompted to change the state of the analyzer and then re-load the state that was previously stored in the disk file. Once the file is created on the disk, the state information can be retrieved from the controller and restored on the analyzer.

NOTE The disk file can only be created once. Errors will occur if the operator repeatedly tries to recreate the file.

For this example, only a thru calibration will be performed and transferred. This means only one calibration array will be read from the analyzer and written to the disk file with the instrument state. To work with more elaborate calibrations, additional arrays will need to be defined and transferred to the disk file. This is not difficult, but requires some further programming steps which were omitted in the interest of presenting a simple example.

The following is an outline of the program's processing sequence:

- An I/O path is assigned for the analyzer.
- A binary path is assigned.
- The integers for reading the headers are defined.
- The system is initialized.
- An array is created to hold the learn string.
- The learn string is requested by the controller.
- The number of points in the trace is read.
- The controller allocates an array for the calibration data.
- The calibration data is read into the controller.
- The controller creates and assigns a data file for the calibration array and the learn string.
- The learn string and calibration array are stored in the disk file.
- The operator presses **Enter,** on the controller to read the calibration data back into the analyzer.
- The learn string is read from the disk file and output to the analyzer.
- The calibration array is read in from the disk file and stored in the analyzer.
- The analyzer is returned to continuous-sweep mode.
- The analyzer is released from remote control and the program ends.

Running the Program

Setup the analyzer and perform a through calibration.

Run the program. The program prompts the operator to change the state of the analyzer and then press **Enter,** to continue. At this point, the analyzer state is stored on the disk file in the controller. Pressing **Enter,** will begin the transfer from the disk file to internal arrays within the controller and then on to the analyzer.

Once completed:

- The original state will be restored.
- The analyzer will be sweeping.
- The analyzer will be calibrated.
- COR will be displayed on the analyzer's display.

BASIC Program Listing

```
10  ! This program reads an instrument state and stores it in a disk file.
20  ! The learn string and CAL array are both read into the controller and
30  ! then transferred to a disk file for storage. The file contents are
40  ! then restored to the analyzer.
50  !
60  ! EXAMP5C Saving and Restoring the Analyzer Instrument State
70  !
80  OPTION BASE 1
90  ASSIGN @Nwa TO 716                  ! Assign an I/O path for the analyzer
100 ASSIGN @Nwa_bin TO 716;FORMAT OFF   ! Assign a binary path
110 !
120 INTEGER Header,Str_len,Cal_len      ! Integer 2 byte format for headers
130 !
140 CLEAR SCREEN
150 ! Initialize the analyzer
160 ABORT 7                             ! Generate an IFC (Interface Clear)
170 CLEAR @Nwa                          ! SDC (Selected Device Clear)
180 !
190 OUTPUT @Nwa;"OPC?;SING;"            ! Place analyzer in single sweep
200 ENTER @Nwa;Reply                    ! Read the 1 when complete
210 !
220 OUTPUT @Nwa;"OUTPLEAS;"             ! Request learn string
230 ENTER @Nwa_bin;Header,Str_len       ! Read header and length first
240 ALLOCATE INTEGER State(Str_len/2)   ! Integer array to contain the string
250 ENTER @Nwa_bin;State(*)             ! Read the string
260 !
270 ! Allocate an array for storing the CAL data
280 OUTPUT @Nwa;"POIN?;"                ! Find number of points in trace
290 ENTER @Nwa;Num_points               ! Read number to allocate array
300 ALLOCATE Cal_array(1:Num_points,1:2)   ! Real and Imag for each point
310 !
320 ! Read Cal array
330 OUTPUT @Nwa;"FORM3;"                ! Form 3 64 bit floating point data
340 OUTPUT @Nwa;"OUTPCALC01;"           ! Request the cal array
350 !
360 ! Read the #A and 2 byte length as integers
370 ENTER @Nwa_bin;Header,Cal_len,Cal_array(*)     ! Read cal array data
380 !
390 ! Write instrument state data to disk file
400 File$="A:DATAFILE"                          ! Example BASIC for Windows data file
410 ! File$ = "DATAFILE:,1406"                  ! Example Workstation BASIC data file
420 CREATE BDAT File$,1,Str_len+Cal_len+8         ! Create data file once!
430 ASSIGN @File TO File$                         ! Assign I/O path to file
440 OUTPUT @File;Header,Str_len,State(*)          ! Send learn string to disk file
450 OUTPUT @File;Header,Cal_len,Cal_array(*)      ! Send CAL arrays to disk file
460 ASSIGN @File TO *                             ! Close file
470 !
480 INPUT "Cal data received. Press ENTER to send it back.",A$
```

```
490 !
500 ! Read arrays from file
510 !
520 ASSIGN @File TO File$                ! Open file for reading arrays
530 ENTER @File;Header,Str_len           ! Read learn string headers from file
540 ALLOCATE INTEGER State2(Str_len/2)   ! new learn string array from file
550 ENTER @File;State2(*)                ! Read learn string from file
560 !
570 ENTER @File;Header,Cal_len           ! Read CAL data headers from file
580 Arrsize=Cal_len/16                   ! Array is 2 numbers, 8 bytes per number
590 ALLOCATE Cal_array2(1:Arrsize,1:2)   ! new cal array from file
600 ENTER @File;Cal_array2(*)            ! Read cal array from disk file
610 !
620 ! Send Learn string back
630 OUTPUT @Nwa;"INPULEAS;"              ! Send learn string to analyzer
640 OUTPUT @Nwa_bin;Header,Str_len,State2(*)
650 !
660 ! Send Cal array back
670 OUTPUT @Nwa;"CALIRESP;"              ! Send CAL type (Response)
680 OUTPUT @Nwa;"FORM3;INPUCALC01;"      ! Output CAL array to analyzer
690 OUTPUT @Nwa_bin;Header,Cal_len,Cal_array2(*)
700 OUTPUT @Nwa;"OPC?;SAVC;"             ! Save the CAL array
710 ENTER @Nwa;Reply                     ! Read the 1 when complete
720 !
730 OUTPUT @Nwa;"CONT;"                  ! Start the analyzer sweeping
740 OUTPUT @Nwa;"OPC?;WAIT;"             ! Wait for the analyzer to finish
750 ENTER @Nwa;Reply
760 LOCAL @Nwa                           ! Release GPIB control
770 END
```

List-Frequency and Limit-Test Table Examples

Using List Frequency Sweep Mode

The analyzer normally takes data points spaced at regular intervals across the overall frequency range of the measurement. For example, for a 2 GHz frequency span using 201 points, data will be taken at intervals of 10 MHz. The list frequency sweep mode allows the operator to select the specific points, or frequency spacing between points, at which measurements are to be made. This mode of operation allows flexibility in setting up tests that ensure device performance. By only sampling specific points, measurement time is reduced. List frequency sweeps are also discussed in your analyzer's user's guide.

The following three example programs illustrate the use of the analyzer's list frequency sweep mode to perform arbitrary frequency testing. Examples 6A (Stepped List Mode) and 6B (Swept List Mode) allow the operator to construct a table of list frequency segments which is then loaded into the analyzer's list frequency table. There are a maximum of 30 segments available. Each segment stipulates a start and stop frequency, and the number of data points to be taken over that frequency range. In Example 6B (Swept List Mode), each segment also stipulates a power value and IF bandwidth. List frequency segments can be overlapped in the stepped list mode but not in the swept list mode. The total number of points in all the segments must not exceed 1601.

Examples 6A (Stepped List Mode) and 6B (Swept List Mode) take advantage of the computer's capabilities to simplify creating and editing a list frequency table. The table is entered and completely edited before being transmitted to the analyzer. To simplify the programming task, options such as entering center frequency, frequency span, or step size are not included.

Example 6C lets the operator select a specific segment to "zoom-in." A single instrument can be programmed to measure several different devices, each with its own frequency range, using a single calibration. When a specific device is connected, the operator selects the appropriate segment for that device.

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

The list frequency information may be acquired using the limit-test results array. The actual stimulus points are available as the first element in the array.

The list frequency table is also carried as part of the learn string. While the table cannot be modified as part of the learn string, it can be stored and recalled with very little effort by storing and recalling the learn string. See "Example 5A: Using the Learn String" on page 8-84 and "Learnstring and Calibration-Kit String" on page 6-5 for details on using learn strings.

Example 6A: Setting Up a List Frequency Table in Stepped List Mode

The purpose of this example is to show how to create a list frequency table in stepped list mode and then transmit the table to the analyzer.

In the stepped list mode, the source steps to the next frequency point where it stops long enough for the analyzer to take data. For electrically long devices, this mode ensures that the measurement will not be impacted by IF delay. In addition, this mode provides the most flexibility in specifying the list of frequencies.

The following is an outline of the program's processing sequence:

- An I/O path is assigned for the analyzer.
- The system is initialized.
- The existing list frequencies are edited and cleared.
- The number of segments to define is read in.
- An array for the list segments is defined.
- The parameters for each segment are requested.
- If the operator wants to edit, the segment parameters are re-entered.
- The new list is sent to the analyzer.
- The analyzer is released from remote control and the program ends.

Running the Program

The program displays the frequency list table as it is entered. During editing, the displayed table is updated as each line is edited. The table is not re-ordered. At the completion of editing, the table is entered into the analyzer, and list frequency sweep mode is switched ON. During editing, pressing **Enter,** leaves an entry at the old value.

The start frequency, stop frequency, and number of points for the last segment entered may be observed on the analyzer's display.

Activate a marker and select the discrete-marker mode to observe the point spacing. Use an exaggerated scale with just a few points to find the list-frequency spacing between points.

BASIC Program Listing

```
10  ! This program shows how to enter and edit a list frequency table.
20  ! Any existing table is deleted and a new table is defined and
30  ! edited. This list is then sent to the analyzer. Any number of
40  ! segments or points may be entered. Be sure not to enter more than
50  ! 1601 points or 30 segments.
60  !
70  !  EXAMP6A Setting Up a List-Frequency Table in Stepped List Mode
80  !
90    ASSIGN @Nwa TO 716               ! Assign an I/O path for the analyzer
100 !
110   CLEAR SCREEN
120 ! Initialize the analyzer
130   ABORT 7                          ! Generate an IFC (Interface Clear)
140   CLEAR @Nwa                       ! SDC (Selective Device Clear)
150   OUTPUT @Nwa;"OPC?;PRES;"         ! Preset the analyzer and wait
160   ENTER @Nwa;Reply                 ! Read the 1 when complete
170 !
180 !  Edit the list frequency table and set its type
190 !  LISTTYPE = LSTP    (stepped list mode)
200 !
```

```
210   OUTPUT @Nwa;"EDITLIST;LISTTYPELSTP;"
220 !
230   OUTPUT @Nwa;"CLEL;"              ! Clear the existing list frequencies
240 !
250   INPUT "Number of segments?",Numb ! Read number of segments to define
260   ALLOCATE Table(1:Numb,1:3)       ! Define an array for the list segments
270 !
280   PRINT USING "10A,15A,15A,20A";"SEGMENT","START(MHZ)","STOP(MHZ)","NUMBER OF
POINTS"
290 !
300   FOR I=1 TO Numb      ! Cycle through the segments and read in the values
310     GOSUB Loadpoin
320   NEXT I
330 !
340   LOOP
350     INPUT "DO YOU WANT TO EDIT? Y OR N",An$
360   EXIT IF An$="N"
370     INPUT "ENTRY NUMBER?",I        ! Get an entry number
380     GOSUB Loadpoin                 ! Go load point
390   END LOOP
400 !
410   OUTPUT @Nwa;"EDITLIST"           ! Send the new list to the analyzer
420   FOR I=1 TO Numb                         ! Send one segment at a time
430     OUTPUT @Nwa;"SADD;"                   ! Add a segment
440     OUTPUT @Nwa;"STAR";Table(I,1);"MHZ;"  ! Start frequency
450     OUTPUT @Nwa;"STOP";Table(I,2);"MHZ;"  ! Stop frequency
460     OUTPUT @Nwa;"POIN",Table(I,3),";"     ! Number of points
470     OUTPUT @Nwa;"SDON;"                   ! Segment done
480   NEXT I                           ! Next segment to send to the analyzer
490 !
500   OUTPUT @Nwa;"EDITDONE;"          ! Done with list
510   OUTPUT @Nwa;"LISFREQ;"           ! Set list frequency mode
520 !
530   OUTPUT @Nwa;"OPC?;WAIT;"         ! Wait for analyzer to finish
540   ENTER @Nwa;Reply                 ! Read the 1 when complete
550   LOCAL @Nwa                       ! Release GPIB control
560   STOP                             ! End of main program
570 !
580 !   *************************Subroutines ****************************
590 !
600 Loadpoin:                          ! Sub to read in each segment value
610   INPUT "START FREQUENCY? (MHZ)",Table(I,1) ! Read start frequency
620   INPUT "STOP FREQUENCY? (MHZ)",Table(I,2)  ! Read stop frequency
630   INPUT "NUMBER OF POINTS?",Table(I,3)      ! Read number of points in seg
640   IF Table(I,3)=1 THEN Table(I,2)=Table(I,1)! Single point same start stop
650 !
660 ! Print new segment into table on display
670   PRINT TABXY(0,I+1);I;TAB(10);Table(I,1);TAB(25);
680   PRINT Table(I,2);TAB(40),Table(I,3)
690   RETURN
700   END
```

Example 6B: Setting Up a List Frequency Table in Swept List Mode

The purpose of this example is to show how to create two tables: a list frequency table in swept list mode, and a limit-test table. Both tables are then transmitted to the analyzer.

In the swept list mode, the source sweeps through each segment, with the analyzer taking data during the sweep. This can increase throughput by up to 6 times over a stepped sweep. In addition, this mode expands the list table to include a power value and IF bandwidth for each defined segment. The only restriction is that you cannot specify segments with overlapping frequencies. For more information on the swept list mode, refer to your analyzer's user's guide.

The following is an outline of the program's processing sequence:

- An array for the type of limit line is defined and initialized.
- An array for the list table (frequency list and limit lines) is defined and initialized with data.
- An I/O path is assigned for the analyzer.
- The system is initialized.
- A variable is initialized with the number of segments in the list table.
- The analyzer is placed into hold mode and (for ES model analyzers) the port powers are uncoupled for the active channel.
- The existing list frequencies are edited and the analyzer swept list mode is selected.
- The analyzer is instructed to set the IF bandwidth and port power levels according to the values from the list table.
- The new frequency list table is sent to the analyzer.
- The sweep mode is set to list frequency mode and S21 (transmission) measurement. A single sweep is taken.
- The analyzer display is autoscaled.
- The existing limit lines are edited and cleared.
- The new limit table is sent to the analyzer.
- The limit lines and limit test are turned on.
- The analyzer is released from remote control and the program ends.

The program is written as follows:

Running the Program

The program requires no input from the operator. The list frequency table data and limit-test table data is read directly from the program code into the array. Next, the analyzer is set up to respond to the IF bandwidth and port power parameters of the list frequency table. After the list frequency data is entered into the analyzer, the list frequency sweep mode is initiated and a single sweep is taken. Lastly, the limit-test table data is entered into the analyzer and the limit lines and limit test are activated.

The analyzer should now indicate whether the measurement trace passes or fails the limit test.

BASIC Program Listing

```
10 ! This program creates a swept list table for a specific filter measurement.
```

```
20 ! The program first builds a list frequency table from a hardcoded set of
30 ! list segments.  It then builds a limit table based on the same hardcoded
40 ! data.  When modifying the table data below, make sure that no two segments
50 ! overlap in frequency.
60 !
70 ! EXAMP6B Setting Up a List-Frequency Table in a Swept List Mode
80 !
90 !-----------------------------------------------------------------
100! The following constants are used to represent limit line “type”
110! in the table below.
120!
130   No_limit=0
140!
150! 1 = flat line
160! 2 = sloped line
170! 3 = single point  (also used to terminate a line segment)
180!-----------------------------------------------------------------
190!
200   DIM Limtype$(1:3)[2]
210   DATA FL, SL, SP
220   READ Limtype$(*)
230!
240! The list below has the following entries:
250!    Start:  start frequency
260!    Stop:   Segment stop
270!    Pts:    Segment number of points
280!    P1:     Power at port 1
290!    P2:     Power at port 2 (ES model analyzers only)
300!    IFBW:   Segment IFBW
310!    upper:  Upper Limit
320!    lower:  Lower Limit
330!    strt type:  Limit Line type for start of segment
340!    end  type:  Limit Line type for end of segment
350!-----------------------------------------------------------------
360   DIM Listtable(1:6,1:10)
370 Freqlist: !                                                |strt|end
380! List: Start |   Stop | Pts | P1 | P2 | IFBW | uppr | lower |type|type
390! ---------------------------------------------------------------------
400! ---------------------------------------------------------------------
410   DATA  570.000,  588.000,    5,  10,   0,   10,  -90,   -200,  1,   0
420   DATA  588.000,  598.000,   11,   0,   0,  100,  -85,   -200,  1,   3
430   DATA  600.000,  664.000,   15, -10, -10, 3700,    0,      0,  0,   0
440   DATA  664.000,  678.000,  100, -10, -10, 3700,    0,     -6,  1,   3
450   DATA  678.000,  768.000,   10, -10, -10, 1000,    0,      0,  0,   0
460   DATA  768.000,  768.000,    1,  10,   0,   10,  -90,   -200,  3,   3
470! ---------------------------------------------------------------------
480   READ Listtable(*)
490!
500   ASSIGN @Nwa TO 716               ! Assign an I/O path for the analyzer
510!
520   CLEAR SCREEN
530! Initialize the system
540   ABORT 7
550   CLEAR @Nwa
560   OUTPUT @Nwa;”OPC?;PRES;”
570   ENTER @Nwa;Done
580   Numb=SIZE(Listtable,1)           ! Number of segments in list table
590   OUTPUT @Nwa;”HOLD;”              ! Hold mode allows faster set up
```

```
600!
610!  For ET model analyzers, comment out this next line
620   OUTPUT @Nwa;"PORTPUNCPLD;"        ! Uncouple ports
630!
640!  Create the list frequency table from the table above
650!  LISTTYPE = LSWP    (swept list mode)
660!
670   OUTPUT @Nwa;"EDITLIST;LISTTYPELSWP;"
680!
690!  Turn on list power mode for each port (uncoupled)
700!  Turn on list ifbw mode
710!
720   OUTPUT @Nwa;"LISIFBWMON;"         ! IF bandwidth set by list
730   OUTPUT @Nwa;"S21;LISPWRMON;"      ! Port 1 power set by list
740!
750!  For ET model analyzers, comment out this next line
760   OUTPUT @Nwa;"S22;LISPWRMON;"      ! Port 2 power set by list
770!
780   FOR I=1 TO Numb
790     OUTPUT @Nwa;"SADD;STAR";Listtable(I,1);"MHZ;"
800     OUTPUT @Nwa;"STOP";Listtable(I,2);"MHZ;"
810     OUTPUT @Nwa;"POIN";Listtable(I,3);";"
820     OUTPUT @Nwa;"S11;"                          ! Port 1 active
830     OUTPUT @Nwa;"SEGPOWER";Listtable(I,4),";"
840!
850!    For ET model analyzers, comment out these next two lines
860     OUTPUT @Nwa;"S22;"                          ! Port 2 active
870     OUTPUT @Nwa;"SEGPOWER";Listtable(I,5),";"
880!
890     OUTPUT @Nwa;"SEGIFBW";Listtable(I,6),";"
900     OUTPUT @Nwa;"SDON;"
910   NEXT I
920   OUTPUT @Nwa;"EDITDONE;"
930   OUTPUT @Nwa;"LISFREQ;S21;OPC?;SING;"
940   ENTER @Nwa;Done
950   OUTPUT @Nwa;"AUTOSCAL;WAIT;"
960!
970!  Now create the corresponding limit table
980!
990   OUTPUT @Nwa;"EDITLIML;CLEAL;"     ! Initiate the limit table
1000  FOR I=1 TO Numb
1010    IF Listtable(I,9)<>No_limit THEN
1020      OUTPUT @Nwa;"SADD"            ! Add a new limit segment
1030      OUTPUT @Nwa;";LIMS";Listtable(I,1);"MHZ"
1040      OUTPUT @Nwa;";LIMU";Listtable(I,7)
1050      OUTPUT @Nwa;";LIML";Listtable(I,8)
1060      OUTPUT @Nwa;";LIMT";Limtype$(Listtable(I,9))
1070      OUTPUT @Nwa;";SDON;"
1080      IF Listtable(I,10)<>No_limit THEN !
1090        OUTPUT @Nwa;"SADD "         ! Add a new limit segment
1100        OUTPUT @Nwa;";LIMS";Listtable(I,2);"MHZ"
1110        OUTPUT @Nwa;";LIMT";Limtype$(Listtable(I,10))
1120        OUTPUT @Nwa;";SDON;"
1130      END IF
1140    END IF
1150  NEXT I
1160  OUTPUT @Nwa;"EDITDONE;LIMILINEon;LIMITESTon;"
1170!
```

```
1180  LOCAL @Nwa
1190  END
```

Example 6C: Selecting a Single Segment from a Table of Segments

This example program demonstrates how to define a single segment as the operating-frequency range of the analyzer from a table of segments stored in the controller. The program assumes that a list frequency table has already been entered into the analyzer, either manually, or using the program in Example 6A or Example 6B.

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

The following is an outline of the program's processing sequence:

- An I/O path is assigned for the analyzer.
- The system is initialized.
- The list frequency segment is edited.
- The largest segment number is set.
- The highest segment number is requested.
- The number of actual segments is read in.
- A list frequency table is defined and the segments are read in to the controller from the analyzer.
- The operator selects one of the segments of the sweep.
- The controller “zooms-in” and sweeps the defined segment.
- The operator ends the program by entering segment number “0.”
- The analyzer returns to sweeping all the segments in the table.
- The activation loop is ended and the program ends.

Running the Program

The program will read the parameters for each list-frequency segment from the analyzer, and build a table containing all the segments. The parameters of each segment will be printed on the computer screen. If there are more than 17 segments, the program will pause. Press **Return,** to see more segments. The maximum number of segments that can be read is 30 (the maximum number of segments the analyzer can hold). Use the computer's **Page Up,** and **Page Down,** keys to scroll through the list of segments if there are more than 17.

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

terminate.

BASIC Program Listing

```
10  ! This program shows how to select a single segment from a list
20  ! frequency sweep and activate it as the sweep. The list frequency
30  ! table is read from the analyzer and displayed on the computer
40  ! screen. The operator is prompted to select a segment and the
50  ! program then activates it. All the segments are activated upon
60  ! completion.
70  !
80  ! EXAMP6C Selecting a Single Segment from a Table of Segments
90  !
100 ASSIGN @Nwa TO 716                  !  Assign an I/O path for the analyzer
110 !
120 CLEAR SCREEN
130 ! Initialize the analyzer
140 ABORT 7                             ! Generate an IFC (Interface Clear)
150 CLEAR @Nwa                          ! SDC (Selected Device Clear)
160 !
170 ! Print header for table of existing segments
180 PRINT USING "10A,15A,15A,20A";"SEGMENT","START(MHZ)","STOP(MHZ)","NUMBER OF
POINTS"
190 OUTPUT @Nwa;"EDITLIST;"             ! Edit list frequency segment
200 OUTPUT @Nwa;"SEDI30;"               ! Set largest segment number
210 OUTPUT @Nwa;"SEDI?;"                ! Request number of highest segment
220 ENTER @Nwa;Numsegs                  ! Read number of actual segments
230 !
240 ! Setup table and read segments from analyzer
250 ALLOCATE Table(1:Numsegs,1:3)       ! Allocate table of segments
260 FOR I=1 TO Numsegs                  ! Cycle through segments
270   GOSUB Readlist                    ! Read in segment definitions
280 NEXT I                              ! Next segment
290 !
300 ! Loop and read segment to be activated
310 LOOP                                ! Request operator to enter segment
320   INPUT "SELECT SEGMENT NUMBER: (0 TO EXIT)",Segment
330 EXIT IF Segment=0                   ! Exit point
340   OUTPUT @Nwa;"EDITDONE;";"SSEG";Segment;";" ! Set active segment to sweep
350 END LOOP                            ! End activation loop
360 !
370 OUTPUT @Nwa;"ASEG;"                 ! Set all segment sweep
380 DISP "PROGRAM ENDED"
390 !
400 OUTPUT @Nwa;"OPC?;WAIT;"            ! Wait for analyzer to finish
410 ENTER @Nwa;Reply                    ! Read the 1 when complete
420 LOCAL @Nwa                          ! Release GPIB control
430 STOP                                ! End of main program
440 !
450 ! ************************* Subroutines ******************************
460 !
470 Readlist:                           ! Read segment list from analyzer
480 OUTPUT @Nwa;"EDITLIST;"             ! Edit segment list
490 OUTPUT @Nwa;"SEDI",I,";"            ! Select segment to edit
500 OUTPUT @Nwa;"STAR;"                 ! Send start freq to display value
510 OUTPUT @Nwa;"OUTPACTI;"             ! Output active function value
520 ENTER @Nwa;Table(I,1)               ! Read start frequency
```

```
530 OUTPUT @Nwa;"STOP;"                ! Send stop freq to display value
540 OUTPUT @Nwa;"OUTPACTI;"            ! Output active function value
550 ENTER @Nwa;Table(I,2)              ! Read stop frequency
560 OUTPUT @Nwa;"POIN;"                ! Send number of points to display
570 OUTPUT @Nwa;"OUTPACTI;"            ! Output active function value
580 ENTER @Nwa;Table(I,3)              ! Read number of points
590 !
600 IF I=18 THEN                       ! Pause if more than 17 segments
610 INPUT "PRESS RETURN FOR MORE",A$   ! Read Return to continue
620 END IF
630 ! Print new header for segment data
640 IMAGE 4D,6X,4D.6D,3X,4D.6D,3X,4D   ! Format image to disp segment data
650 PRINT USING 640;I;Table(I,1)/1.E+6;Table(I,2)/1.E+6;Table(I,3)
660 RETURN
670 !
680 END
```

Using Limit Lines to Perform PASS/FAIL Tests

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

Example 6D illustrates the first step, setting up limits. Example 6E performs the actual limit testing.

Example 6D: Setting Up a Limit-Test Table

The purpose of this example is to show how to create a limit-test table and transmit it to the analyzer.

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

The limit-test table is also carried as part of the learn string. While it cannot be modified as part of the learn string, the learn string can be stored and recalled with very little effort. See "Example 5A: Using the Learn String" on page 8-84 and "Learnstring and Calibration-Kit String" on page 6-5 for details on using learn strings.

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

This example automates the front-panel operation of entering a limit-test table. Front-panel operation and limits are discussed in your analyzer's user's guide.

The following is an outline of the program's processing sequence:

- An I/O path is assigned for the analyzer.

- The system is initialized.
- The limit lines are edited and cleared.
- The number of limits is requested.
- The limit table is created.
- The string array of limit types is created.
- The operator is prompted to enter the new limit values.
- The new limit table is sent back to the analyzer.
- The limit line is activated.
- The limit test is activated.
- The analyzer is returned to local control and the program ends.

Running the Program

CAUTION This example program will delete any existing limit lines before entering the new limits. If this is not desired, omit the line(s) that clear the existing limits (in this case, the command "CLEL;" contained in LINE 190). This program begins by presetting the analyzer. The programmer will have to add the necessary command lines to set the analyzer to the operating conditions required for testing. The example program will show the limit lines defined, but the limits will always fail without additional analyzer setup.

The program displays the limit table as it is entered. During editing, the displayed table is updated as each line is edited. The table is not reordered. At the completion of editing, the table is entered into the analyzer, and limit-testing mode switched ON. The analyzer will rearrange the table in ascending order starting with the lowest start frequency entry. During editing, simply pressing **Enter,** leaves an entry at the old value.

BASIC Program Listing

```
10  ! This program shows how to create a limit table and send it to the
20  ! analyzer. The operator enters the desired limits when prompted for
30  ! the stimulus value, upper and lower value and type of limit
40  ! desired. Once the table is created, the limits are sent to the
50  ! analyzer and activated.
60  !
70  ! EXAMP6D Setting Up a Limit Test Table
80  !
90  ASSIGN @Nwa TO 716                  ! Assign an I/O path for the analyzer
100 !
110 CLEAR SCREEN
120 ! Initialize the analyzer
130 ABORT 7                             ! Generate an IFC (Interface clear)
140 CLEAR @Nwa                          ! SDC (Selected Device Clear)
150 OUTPUT @Nwa;"OPC?;PRES;"            ! Preset the analyzer and wait
160 ENTER @Nwa;Reply                    ! Read the 1 when completed
170 !
180 OUTPUT @Nwa;"EDITLIML;"             ! Edit limit lines
190 OUTPUT @Nwa;"CLEL;"                 ! Clear any existing limits
```

```
200 INPUT “NUMBER OF LIMITS?”,Numb       ! Request the number of limits
210 ALLOCATE Table(1:Numb,1:3)           ! Create a table
220 ALLOCATE Limtype$(Numb)[2]           ! Create string array of limit types
230 !
240 ! Print out the header for the table
250 PRINT USING “10A,20A,15A,20A”;”SEG”,”STIMULUS (MHz)”,”UPPER (dB)”,”LOWER
(dB)”,”TYPE”
260 !
270 ! Prompt the operator to enter the limit values
280 FOR I=1 TO Numb                      ! Cycle through the limits
290   GOSUB Loadlimit                    ! Go read limit values
300 NEXT I                               ! Next limit value
310 !
320 ! Allow the operator to edit the array entered
330 LOOP                                 ! Cycle to edit limit lines
340   INPUT “DO YOU WANT TO EDIT? Y OR N”,An$
350 EXIT IF An$=”N”                      ! Exit loop on N and send to analyzer
360   INPUT “ENTRY NUMBER?”,I            ! Read limit number to edit
370   GOSUB Loadlimit                    ! Go read limit values
380 END LOOP                             ! Next edit entry
390 !
400 ! Send the limit line array segments to the analyzer
410 OUTPUT @Nwa;”EDITLIML;”              ! Edit the limit
420 FOR I=1 TO Numb                      ! Each segment of the limit
430   OUTPUT @Nwa;”SADD;”                  ! Add segment
440   OUTPUT @Nwa;”LIMS”;Table(I,1);”MHZ”  ! Send segment stimulus value
450   OUTPUT @Nwa;”LIMU”;Table(I,2);”DB”   ! Upper limit value
460   OUTPUT @Nwa;”LIML”;Table(I,3);”DB”   ! Lower limit value
470   IF Limtype$(I)=”FL” THEN OUTPUT @Nwa;”LIMTFL;” ! Flat limit
480   IF Limtype$(I)=”SL” THEN OUTPUT @Nwa;”LIMTSL;” ! Sloping limit
490   IF Limtype$(I)=”SP” THEN OUTPUT @Nwa;”LIMTSP;” ! Point limit
500   OUTPUT @Nwa;”SDON;”                ! Segment done
510 NEXT I                               ! next segment
520 !
530 OUTPUT @Nwa;”EDITDONE;”              ! Edit complete
540 OUTPUT @Nwa;”LIMILINEON;”            ! Turn limit line on
550 OUTPUT @Nwa;”LIMITESTON;”            ! Turn limit test on
560 !
570 OUTPUT @Nwa;”OPC?;WAIT;”             ! Wait for the analyzer to finish
580 ENTER @Nwa;Reply                     ! Read the 1 when complete
590 !
600 LOCAL @Nwa                           ! Release GPIB control
610 STOP                                 ! End of main program
620 !
630 !*************************** Subroutines ******************************
640 !
650 Loadlimit:                           ! Sub to interact to load data
660 INPUT “STIMULUS VALUE? (MHz)”,Table(I,1)  ! and print table created
670 INPUT “UPPER LIMIT VALUE? (DB)”,Table(I,2)
680 INPUT “LOWER LIMIT VALUE? (DB)”,Table(I,3)
690 INPUT “LIMIT TYPE? (FL=FLAT, SL=SLOPED, SP=SINGLE POINT)”,Limtype$(I)
700 !
710                     ! Format and display table values
720 PRINT
TABXY(0,I+1);I;TAB(10);Table(I,1);TAB(30);Table(I,2);TAB(45),Table(I,3),TAB(67);Limty
pe$(I)
730 RETURN                                ! Next limit value
740 !
```

```
750 END
```

Example 6E: Performing PASS/FAIL Tests While Tuning

The purpose of this example is to demonstrate the use of the limit/search-fail bits in event-status register B, to determine whether a device passes the specified limits. Limits can be entered manually, or using Example 6D.

The limit/search-fail bits are set and latched when limit testing or a marker search fails. There are four bits, one for each channel for both limit testing and marker search. See Figure 7-1 on page 7-3 and Table 5-1 on page 5-6 for additional information. Their purpose is to allow the computer to determine whether the test/search executed was successful. They are used in the following sequence:

1. Clear event-status register B.
2. Trigger the limit test or marker search.
3. Check the appropriate fail bit.

When using limit testing, the best way to trigger the limit test is to trigger a single sweep. By the time the single sweep command (`SING`) finishes, limit testing will have occurred.

NOTE If the device is tuned during the sweep, it may be tuned into and then out of limit, causing a limit test to qualify as “passed” when the device is not in fact within the specified limits.

When using marker searches (max, min, target, and widths), outputting marker or bandwidth values automatically triggers any related searches. Therefore, all that is required is to check the fail bit after reading the data.

In this example, several consecutive sweeps must qualify as “passing” in order to ensure that the limit-test pass was not extraneous due to the device settling or operator tuning during the sweep. Upon running the program, the number of “passed” sweeps for qualification is entered. For very slow sweeps, a small number of sweeps such as two are appropriate. For relatively fast sweeps, where the device requires time to settle after tuning, as many as six or more sweeps may be more appropriate.

The following is an outline of the program's processing sequence:

- An I/O path is assigned for the analyzer.
- The system is initialized.
- The pass counter is initialized on entry.
- The analyzer takes a sweep.
- The event-status register B byte is output and the channel-1 limit is tested.
- If the device fails the first sweep, the operator is prompted to ensure it is tuned correctly and the device is measured again.
- If the device passes the first sweep, the operator is prompted not to touch the device as testing continues.

- If the device passes the required number of sweeps, the operator is prompted that the device has passed and to connect the next device for testing.
- The program initializes the pass counter and begins to measure the new device.

Running the Program

NOTE This program assumes a response calibration (through calibration) or a full 2-port calibration has been performed prior to running the program.

Set up a limit-test table on channel 1 for a specific device either manually, or using the program in Example 6D.

Run the program, and enter the number of passed sweeps desired for qualification. After entering the qualification number, connect the device. When a sweep passes, the computer beeps. When enough consecutive sweeps qualify the device as “passing,” the computer emits a dual-tone beep to attract the attention of the operator, and then prompts for a new device.

To test the program's pass/fail accuracy, try causing the DUT to fail by loosening the cables connecting the DUT to the analyzer and running the program again.

BASIC Program Listing

```
10  ! This program demonstrates Pass/Fail tests using limit lines. The
20  ! program uses the latch-on-fail limit bits in event status register
30  ! B to determine if the device performance passes the specified test
40  ! limit lines.  It then requires that the device passes for multiple
50  ! consecutive sweeps in order to ensure that the device is static in
60  ! the response and not varying. The operator specifies how many sweeps
70  ! are required to pass the test.
80  !
90  ! EXAMP6E Performing PASS/FAIL Tests while Tuning
100 !
110 ASSIGN @Nwa TO 716                  ! Assign an I/O path for the analyzer
120 !
130 CLEAR SCREEN
140 ! Initialize the analyzer  No preset to retain settings for testing
150 ABORT 7                             ! Generate an IFC (Interface Clear)
160 CLEAR @Nwa                          ! SDC (Selected Device Clear)
170 !
180 INPUT “Number of consecutive passed sweeps for qualification?”,Qual
190 Pass=0                              ! Initialize pass counter on entry
200 !
210 Tune: DISP “TUNE DEVICE AS NECESSARY”  ! Device is not passing warning
220 !
230 Measure:OUTPUT @Nwa;”OPC?;SING;”    ! Single sweep and wait
240 ENTER @Nwa;Reply                    ! Read the 1 when completed
250 !
260 OUTPUT @Nwa;”ESB?;”                 ! Event status register B byte
270 ENTER @Nwa;Estat                    ! Reading byte clears the register
280 !
290 IF BIT(Estat,4) THEN                ! Bit 4 is failed limit on channel 1
300      IF Pass>0 THEN BEEP 1200,.05   ! passed before? Now not passing beep
310      Pass=0                         ! Reset pass to 0
320      GOTO Tune                      ! Adjust and measure again
330 END IF
```

```
340   !
350 BEEP 2500,.01                        ! Limit test passed passing beep
360 Pass=Pass+1                          ! Increment number of passes
370 DISP “LEAVE DEVICE ALONE”            ! Warn not to adjust as it passed
380 !
390 IF Pass<Qual THEN GOTO Measure       ! If not enough passes to qualify
400 !
410 ! Device passed
420 DISP “DEVICE PASSED!”                ! Number of passes enough to qualify
430 FOR I=1 TO 10                        ! Announce the device passed and
440      BEEP 1000,.05                   ! prompt operator to connect new
450      BEEP 2000,.01                   ! device to test.
460 NEXT I
470 !
480 INPUT “PRESS RETURN FOR NEXT DEVICE”,Dum$
490 Pass=0                               ! Initialize pass counter
500 GOTO Measure                         ! Begin measurement
510 !
520 END
```

Report Generation Examples

The analyzer has three operating modes with respect to GPIB. These modes can be changed by accessing softkeys in the **Local** menu. System-controller mode is used when no computer is present. This mode allows the analyzer to control the system. The other two modes allow a remote system controller to coordinate certain actions: in talker/listener mode the remote system controller can control the analyzer, as well as coordinate plotting and printing; and in pass-control mode the remote system controller can pass active control to the analyzer so that the analyzer can plot, print, control a power meter, or load/store to disk. The amount of peripheral interaction is the main difference between talker/listener and pass-control mode.

Example 7A: Operation Using Talker/Listener Mode

The commands `OUTPPLOT` and `OUTPPRIN` allow talker/listener mode plotting and printing via a one way data path from the analyzer to the plotter or printer. The computer sets up the path by addressing the analyzer to talk, the plotter to listen, and then releasing control of the analyzer in order to transfer the data. The analyzer will then make the plot or print. When it is finished, it asserts the End or Identify (EOI) control line on GPIB. The controller detects the presence of EOI and re-asserts control of the GPIB. This example program makes a plot using the talker/listener mode.

NOTE One of the attributes of the `OUTPPLOT` command is that the plot can include the current softkey menu. The plotting of the softkeys is enabled with the `PSOFTON;` command and disabled with the `PSOFTOFF;` command.

The following is an outline of the program's processing sequence:

- An I/O path is assigned for the analyzer.
- The system is initialized.
- The selected frequency span is swept once.
- The plot command is sent to the analyzer.
- The analyzer is set to talker mode and the plotter is set to listener mode.
- The plot is spooled to the plotter.
- The analyzer is set to listener mode when the controller detects an EOI from the analyzer.
- The controller puts the analyzer back in continuous-sweep mode.
- The analyzer is returned to local control and the program ends.

Running the Program

The analyzer will go into remote, and make the plot. During the plot, the computer will display the message `Plotting and waiting for EOI`. When the plot is completed, the analyzer asserts the EOI line on the GPIB. The computer detects this and displays the `End of plot` message.

If a problem arises with the plotter, such as no pen or paper, the analyzer cannot detect the

situation because it only has a one-way path of communication. Hence, the analyzer will attempt to continue plotting until the operator intervenes and aborts the plot by pressing the analyzer's **Local,** key.

Pressing **Local,** will do the following:

- Aborts the plot.
- Causes the warning message `CAUTION: PLOT ABORTED`.
- Asserts EOI to return control of the bus to the system controller.

Because of possible peripheral malfunctions, it is generally advisable to use pass-control mode, which allows two way communication between the peripherals and the analyzer.

BASIC Program Listing

```
10  ! This example shows a plot operation under the control of the
20  ! analyzer. The analyzer is commanded to output plot data, the
30  ! plotter is addressed to listen, and the analyzer to talk. The
40  ! controller watches for EOI at the end of the plot sequence and
50  ! then regains control of the GPIB operations.
60  !
70  ! EXAMP7A Operation Using Talker/Listener Mode
80  !
90  ASSIGN @Nwa TO 716                  ! Assign an I/O path for the analyzer
100 !
110 CLEAR SCREEN
120 ! Initialize analzyer without preset to preserve data
130 ABORT 7                             ! Generate an IFC ( Interface Clear)
140 CLEAR @Nwa                          ! SDC (Selected Device Clear)
150 !
160 OUTPUT @Nwa;"OPC?;SING;"            ! Stop sweep and prepare for plot
170 ENTER @Nwa;Reply                    ! Read in "1" when completed
180 !
190 OUTPUT @Nwa;"OUTPPLOT;"             ! Send plot command
200 SEND 7;UNL LISTEN 5 TALK 16 DATA    ! Unlisten address devices and plot
210 DISP "Plotting and waiting for EOI"
220 WAIT .5                             ! Pause 500 mS to start process
230 !
240 REPEAT                              ! Loop until EOI detected bit is set
250   STATUS 7,7;Stat                   ! Read GPIB interface register 7
260 UNTIL BIT(Stat,11)                  ! Test bit 11 EOI on GPIB
270 !
280 End_plot:DISP "End of plot"
290 !
300 OUTPUT @Nwa;"CONT;"                 ! Restore continuous sweep
310 OUTPUT @Nwa;"OPC?;WAIT;"            ! Wait for analyzer to finish
320 ENTER @Nwa;Reply                    ! Read the 1 when complete
330 LOCAL @Nwa                          ! Release remote control
340 END
```

Example 7B: Controlling Peripherals Using Pass-Control Mode

NOTE This example program will not work with BASIC for Windows.

If the analyzer is in pass-control mode and it receives a command telling it to plot, print, control a power meter, or store/load to disk, it sets bit 1 in the event-status register to indicate that it requires control of the bus. If the computer then uses the GPIB pass-control command to pass control to the analyzer, the analyzer will take control of the bus and access the peripheral. When the analyzer no longer requires control, it will pass control back to the computer.

In this example, the pass-control mode is used to allow the analyzer to dump a screen display to a printer.

Pass-control mode allows the analyzer to control the printer while sending the screen display to be printed. The analyzer requests control from the instrument controller and the controller allows the analyzer to take control of the GPIB and dump the plot. The instrument controller must not interact with the GPIB while this remote analyzer control is taking place. Once the printer-dump operation is complete, the analyzer passes control back to the controller and the controller continues programming the analyzer.

NOTE The analyzer assumes that the address of the computer is correctly stored in its GPIB addresses menu under **Local, SET ADDRESSES, ADDRESS: CONTROLLER.** If this address is incorrect, control will not return to the computer. Similarly, if control is passed to the analyzer while it is in talker/listener mode, control will not return to the computer.

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

When the analyzer becomes the active system controller, it is free to address devices to talk and listen as required. The only functions denied the analyzer are the ability to assert the interface clear line (IFC), and the remote line (REN). These are reserved for the master system controller. As the active system controller, the analyzer can send and receive messages from printers, plotters, and disk drives.

The following is an outline of the program's processing sequence:

- An I/O path is assigned for the analyzer.
- The system is initialized.
- The status registers are cleared.
- Bit 1 of ESR request control is set.
- The ESR interrupt for SRQ is enabled.
- The pass-control mode is enabled.
- The data is dumped to the printer.
- The program loops until the SRQ bit is set.
- The status byte is read with a serial poll.
- The program tests for bit 6, SRQ.
- If SRQ is detected, the program tests for pass control (bit 5 of the status byte).

- If the analyzer requests control, the system controller gives the analyzer control of the bus.
- The program loops and waits for the analyzer to complete the print dump.
- The analyzer reads the interface.
- If bit 6 is active in the controller, control is returned from the analyzer to the controller.
- The status-byte assignments are cleared.
- The analyzer returns to continuous-sweep mode.
- The analyzer is returned to local control and the program ends.

Running the Program

The analyzer will briefly flash the message `WAITING FOR CONTROL`, before actually receiving control and generating the printer output. The computer will display the `Printing from analyzer and waiting for control` message.

When the printer output is complete, the analyzer passes control back to the address stored as the controller address under the **Local, SET ADDRESSES,** menu. The computer will detect the return of active control and exit the wait loop. The controller will display the message `Control returned from analyzer` and then release the analyzer from remote control.

NOTE Because the program waits for the analyzer's request for control, it can be used to respond to front-panel requests as well. Remove the `"PRINALL;"` command from the program and run the program. Nothing will happen until the operator requests a print, plot, or disk access from the front panel of the analyzer. For example, press **Local, Copy,** and **PRINT MONOCHROME.**

BASIC Program Listing

```
10  ! This example shows a pass-control operation to print the display
20  ! under the analyzer GPIB control. The controller reads the status
30  ! of the analyzer looking for SRQ to indicate that the analyzer is
40  ! requesting control. Once control is passed to the analyzer, the
50  ! controller monitors the status of its interface registers to detect
60  ! when the interface is again the active controller. The analyzer will
70  ! pass control back to the controller when finished.
80  !
90  ! EXAMP7B Controlling Peripherals Using Pass-Control Mode
100 !
110 ASSIGN @Nwa TO 716                  ! Assign an I/O path for the analyzer
120 !
130 CLEAR SCREEN
140 ! Initialize the analyzer without preset to preserve data
150 ABORT 7                             ! Generate an IFC ( Interface Clear)
160 CLEAR @Nwa                          ! SDC (Selected Device Clear)
170 !
180 OUTPUT @Nwa;"OPC?;SING;"            ! Single sweep and stop for print
190 ENTER @Nwa;Reply                    ! Read in "1" when complete
200 !
210 OUTPUT @Nwa;"CLES;"                 ! Clear status registers
220 OUTPUT @Nwa;"ESE2;"                 ! Enable bit 1 of ESR request control
```

```
230 OUTPUT @Nwa;"SRE32;"                ! Enable ESR interrupt for SRQ
240 !
250 OUTPUT @Nwa;"USEPASC;"              ! Enable pass control mode
260 OUTPUT @Nwa;"PRINALL;"              ! Begin printer dump
270 !
280 REPEAT                              ! Loop until SRQ bit is set
290   Stat=SPOLL(@Nwa)                  ! Read status byte with serial poll
300 UNTIL BIT(Stat,6)                   ! Test for bit 6, SRQ
310 !
320 Pass_control:                       ! SRQ detected. Test for pass control
330 IF BIT(Stat,5) THEN                 ! Requested pass control
340   PASS CONTROL @Nwa                 ! Send take control message
350 ELSE                                ! Not bit 5, some other event
360   DISP "SRQ but not request pass control"
370   STOP                              ! Halt program
380 END IF
390 !
400 DISP "Printing from analyzer and waiting for control"
410 !
420 REPEAT                              ! Loop and wait for completion
430   STATUS 7,6;Gpib                   ! Read GPIB interface register
440 UNTIL BIT(Gpib,6)                   ! Bit 6 is active controller
450 !
460 DISP "Control returned from analyzer"
470 OUTPUT @Nwa;"TALKLIST;"             ! Set talker/listener mode again
480 OUTPUT @Nwa;"CLES;"                 ! Clear status byte assignments
490 !
500 OUTPUT @Nwa;"CONT;"                 ! Start analyzer sweeping again
510 OUTPUT @Nwa;"OPC?;WAIT;"            ! Wait for analyzer to finish
520 ENTER @Nwa;Reply                    ! Read the 1 when complete
530 !
540 LOCAL @Nwa                          ! Release GPIB control
550 END
```

Example 7C: Printing with the Parallel Port

This program will select the parallel port and program the analyzer to copy its display to a printer. There are a number of commands associated with the serial and parallel ports which allow you to configure output modes such as the baud rate and the handshake type used by the port and the printer. In this example, the parallel port is configured by the program. The interface may also be configured from the analyzer's front-panel keys by pressing **Local, SET ADDRESSES, PRINTER PORT.** This menu allows manual selection of the parallel-interface parameters.

Since the GPIB port is not being used for the copy operation, programming of the analyzer and measurement operations may continue once the copy operation has been initiated. An internal spooler in the analyzer's memory provides buffering of the printer operation. In the example which follows, the status byte of the analyzer is checked to determine when the print operation is complete.

- An I/O path is assigned to the analyzer.
- The analyzer is initialized.

- A single sweep is taken and the analyzer is placed in hold mode.
- The status registers are cleared.
- The copy-complete bit is set and enabled.
- The printer operation and communication modes are set.
- The print command is sent.
- The analyzer is released from remote control and placed in continuous-sweep mode.
- The analyzer is polled until the status bit representing copy complete is detected.
- The analyzer is released from remote control and the program ends.

Running the Program

Run the program. The analyzer is initialized, set to single-sweep mode, and a sweep is taken. The program sets the system up to print the analyzer's display to a printer connected to the parallel port. At this time, the analyzer can continue making measurements as printer prints the display. When the analyzer display has finished printing, the controller displays the message: "DONE", the analyzer is released from GPIB control, and the program ends.

BASIC Program Listing

```
10  ! This program shows how to set up and print the display through the
20  ! parallel printer port.
30  !
40  ! EXAMP7C Printing with the Parallel Port
50  !
60  ASSIGN @Nwa TO 716                  ! Assign an I/O path for the analyzer
70  !
80  CLEAR SCREEN
90  ! Initialize the analzyer without preset to preserve the data
100 ABORT 7                             ! Generate an IFC (Interface Clear)
110 CLEAR @Nwa                          ! SDC (Selected Device Clear)
120 !
130 OUTPUT @Nwa;"OPC?;SING;"            ! Single sweep and stop for print
140 ENTER @Nwa;Reply                    ! Read the 1 when complete
150 !
160 OUTPUT @Nwa;"CLES;"                 ! Clear status registers
170 OUTPUT @Nwa;"ESNB128;"              ! Enable copy complete
180 OUTPUT @Nwa;"SRE4;"                 ! Enable Event Status Register B
190 OUTPUT @Nwa;"PRNTRAUTF OFF;"        ! Set printer auto feed off
200 OUTPUT @Nwa;"PRNTYPLJ;"             ! Select LaserJet printer
210 OUTPUT @Nwa;"PRNPRTPARA;"           ! Select parallel port for output
240 !
250 OUTPUT @Nwa;"PRINALL;"              ! Print screen
260 !
270 DISP "PRINTING"
280 !
290 !  Set up next measurement over GPIB
300 OUTPUT @Nwa;"CONT;"                 ! Restore continuous sweep
310 !
320 ! Measurements can continue but wait for print to finish
330 REPEAT                              ! Test for bit 2 (4) ESRB
340 Stat=SPOLL(@Nwa)
350 UNTIL BIT(Stat,2)                   ! Wait for printer to complete
```

```
360 !
370 DISP “DONE”
380 LOCAL @Nwa                          ! Release GPIB control
390 END
```

Example 7D: Plotting to a File and Transferring the File Data to a Plotter

Another report-generation technique is to transfer the plotter string to a disk file, and retrieve and plot the disk file at another time. Test time is increased when a hardcopy plot occurs during the measurement process. It may be more convenient to plot the data at another site or time. One solution to this problem is to capture the plot data using the controller and store it to a disk file. This disk file may then be read from the controller and the contents transferred to a plotter. This next example shows a method of accomplishing this task.

The analyzer is initialized without presetting the analyzer. The data that is in place on the analyzer is not disturbed by the program operation. A large string is dimensioned to hold the plotter commands as they are received from the analyzer. The length of this string depends upon the complexity of the analyzer's display. The analyzer is placed in the single-sweep mode and `OPC?;SING;` is used to make sure that operation is complete before plotting. The plotting begins with the `OUTPPLOT;` command.

The string transfer is ended by the controller detecting the EOI line which the analyzer pulls at the end of the transfer. The string transfer terminates and the plot data is now stored in a string in the analyzer.

These strings contain ASCII characters which represent the plotter commands in HPGL graphics language. A disk file is created and the string is written into the file containing the display-plot commands.

Once the strings are transferred to the disk file, the file pointer is rewound and the data is read out into a string for plotting. The string is sent to the plotter which uses the commands to generate a plot.

The following is an outline of the program's processing sequence:

- An I/O path is assigned for the analyzer.
- An I/O path is assigned for the plotter.
- The system is initialized.
- The string for plotter commands is defined.
- The frequency span is swept once.
- The plotter output is requested and read into the plot string.
- A plot file is created in the controller.
- The plot string is stored into the disk file.
- The plot string is read from the disk file and sent to the plotter.
- The analyzer returns to continuous-sweep mode.

- The analyzer is returned to local control and the program ends.

Running the Program

The program begins by initializing the analyzer and placing it into single-sweep mode. The plotter commands are captured into strings in the controller. The controller display prompts `Plotter output complete. Press RETURN to store on disk.` Pressing **Return,** causes the data to be stored to disk. Once this task is complete, the program prompts once more, `Plot to file is complete. Press Return to plot.` After pressing **Return,** again, the string output is sent to the plotter and the plot begins. Once the plot is complete, the program prompts `Plot is complete. End of program.` and the analyzer begins sweeping and returns to local control.

BASIC Program Listing

```
10  ! This program shows how to read the plotter output from the analyzer
20  ! and store it in a disk file as an ASCII file. The disk file is then
30  ! read back into the controller and the plot commands sent to a
40  ! plotter to generate the plot of the analyzer display. This allows
50  ! plotting at a different time than data collection.
60  !
70  ! EXAMP7D Plotting to a File and Transferring the File Data to a Plotter
80  !
90  ASSIGN @Nwa TO 716                    ! Assign an I/O path for the analyzer
100 ASSIGN @Plt TO 705                    ! Assign an I/O path for the plotter
110 !
120 CLEAR SCREEN
130 ! Initialize the analyzer without preset to preserve data
140 ABORT 7                               ! Generate an IFC (Interface Clear)
150 CLEAR @Nwa                            ! SDC (Selected Device Clear)
160 !
170 DIM Plot$[32000]                      ! Define string for plotter commands
180 !
190 OUTPUT @Nwa;"OPC?;SING;"              ! Stop sweep for plot and wait
200 ENTER @Nwa;Reply                      ! Read the 1 when complete
210 OUTPUT @Nwa;"OUTPPLOT;"               ! Request plotter output
220 !
230 ENTER @Nwa;Plot$                      ! Plotter output of analyzer display
240 !
250 INPUT "Plotter output complete. Press RETURN to store on disk.",Reply$
260 !
270 ! Disk file operations
280 ! Create data file on disk  32000/256 = 125 records
290 !CREATE ASCII "PLOTFILE:,1400",125    ! Use only once to generate file
300 ASSIGN @File TO "PLOTFILE:,1400"      ! Assign file I/O path
310 OUTPUT @File;Plot$                    ! Write plot string to file
320 !
330 INPUT "Plot to file is complete. Press Return to plot.",A$
340 !
350 ! Read plotter commands from file and send to plotter
360 RESET @File                           ! Reset file pointer to beginning
370 ENTER @File;Plot$                     ! Read plot string from file
380 OUTPUT @Plt;Plot$                     ! Send plot string to plotter
390 !
400 !
410 DISP "Plot is complete. End of program."
420 OUTPUT @Nwa;"CONT;"                   ! Restore continuous sweep
```

```
430 OUTPUT @Nwa;"OPC?;WAIT;"              ! Wait for analzyer to finish
440 ENTER @Nwa;Reply                      ! Read the 1 when complete
450 LOCAL @Nwa                            ! Release GPIB control
460 END
```

Utilizing PC-Graphics Applications Using the Plot File

You can use Example 7D to generate a plot that can be read into a PC and used in several different graphics generation programs. HPGL is a commonly recognized graphic format and may be used to transfer information to PC application programs such as CorelDRAW!®, Lotus Freelance® and other graphics packages. By importing the graphics data into these application packages, you can generate reports in many word-processors.

You can then use graphic-data files to generate the following:

- test results documentation
- data sheets from testing results
- archival information for a digital-storage medium

If you would like to create a disk file for graphics processing, modify the previous program to only store the plotter commands to the disk file. Once the file is renamed to include the extension ".hpg", the PC will have a DOS-format file that can be imported and examined by the graphics package.

Once the HPGL file is present in the DOS file system, the HPGL file is imported and examined with the graphics package. The text labels may need to be rescaled, but on the whole, the graphics results are quite usable.

Example 7E: Reading Plot Files from a Disk

NOTE This example program will not work with BASIC for Windows.

This example program reads and plots files which have been stored on a LIF formatted disk by the analyzer. The plots may be sent to either a plotter with auto-feed capability, such as the HP 7550B, or an HPGL/2 compatible printer, such as a LaserJet 4 Series (monochrome) printer or a DeskJet 1200C (color) printer.

NOTE Sending plots to disk is discussed in your analyzer's user's guide.

This section provides detailed information on file naming conventions and instructions on printing multiple plots-per-page.

This program requires BASIC 6.0 or greater which provides the use of wild cards with the catalog command.

The peripheral GPIB addresses and the hard copy device selection are hard coded and may need to be changed for different systems configurations. Refer to lines 1130 through 1240 in the example program.

This program example provides the form feeds to separate the plots. If the analyzer has been configured to store multiple plots-per-page, this program will generate those plots. A file naming convention has been devised to allow the program perform several printer-setup functions. These include: initializing the printer for HPGL/2 at the beginning of a page, configuring the printer to plot multiple plots to the same page, if desired, and then sending a page eject (form feed) to the hardcopy device at the completion of the printing process.

The plot file name is made up of four parts. The first three are generated automatically by the analyzer whenever a plot is requested:

1. the prefix “PLOT”
2. a two-digit sequence number in the range of 00 to 31
3. a two-letter output-format code to indicate the plot quadrant position:

    - LU (Left Upper)
    - LL (Left Lower)
    - RU (Right Upper)
    - RL (Right Lower)
    - FP (Full Page)

For example, the first full page plot to a disk would be named “PLOT00FP.” The second plot, to be located in the lower-right hand corner of the page would be named “PLOT01RL,” and so on.

The fourth part is an optional character. It is used to indicate that the file is part of a multiple-file plot on the same graticule. See your analyzer’s user’s guide for information on printing multiple measurements per page.

For detailed information on plotting to disk and outputting the plot files to a printer/plotter see your analyzer’s user’s guide.

The following is an outline of the program's processing sequence:

- Hardcopy device control strings are created.
- The hardcopy output device is defined.
- The disc storing the file names is cataloged.
- Files that match the name specifier in the file name array (Flnm$) are counted.
- Flnm$ is dimensioned for the number of files that match the name specifier.
- The file name array is sorted.
- A form feed is sent to the hardcopy device.
- Each of the files in the file name array is processed and sent to the output device.

    The current file name root string is defined.

    If the hardcopy device is a printer, then an HPGL initialization string is output.

    Each file which matches the file name root string to the hardcopy device is output.

    After a form feed is sent to the hardcopy device, the printing/plotting process begins.

Running the Program

This program allows you to plot or print as many as four analyzer display dumps (stored on a LIF formatted disk) on one side of a single sheet of paper. Refer to the instructions detailed in your analyzer's user's guide.

BASIC Program Listing

```
10  ! This example program reads and plots files which have been stored on
20  ! a LIF formatted disk by the analyzer. The plots are sent to either a
30  ! plotter with auto-feed capability, such as the HP7550B, or an HPGL/2
40  ! compatible printer, such as the LaserJet 4 series (monochrome) or the
50  ! DeskJet 1200C (color).
60  !
70  ! Sending plots to disk is discussed in the Printing, Plotting and Saving
80  ! Measurement Results section of the analyzer Users Guide. The file
90  ! naming conventions are discussed in this section and will provide more
100 ! details.
110 !
120 ! This program example will provide form feeds to separate the plots. If
130 ! multiple plots per page have been stored, this program will generate
140 ! those plots.
150 ! A file naming convention has been devised to allow the program
160 ! to initialize the printer for HPGL/2 at the beginning of a
170 ! page, to plot multiple plots to the same page, if desired, and
180 ! when all plots to the same page have been completed then send a
190 ! page eject (form feed) to the hardcopy device.
200 !
210 ! The plot file name is made up of four parts, the first three of which
220 ! are generated automatically by the analyzer whenever a plot is requested:
230 ! The prefix, “PLOT”, a two digit sequence number, in the range of 00 to
240 ! 31, a two letter output format code to indicate the plot quadrant
250 ! position or full page, LU (Left Upper), LL (Left Lower), RU (Right
260 ! upper), RL (Right Lower) or FP (Full Page). For example, the first
270 ! full page plot to a disk would be named “PLOT00FP”.
280 !
290 ! a. Build hardcopy device control strings.
300 ! b. Define output hardcopy device.
310 ! c. Catalog disc storing the file names which match file name specifier
320 !    in the file name array, Flnm$, and setting the number of files that
330 !    match.
340 ! d. Dimension Flnm$ for the number of files matched.
350 ! e. Sort the file name array.
360 ! f. Form feed the hardcopy device.
370 ! g. Process each of the files in the file name array.
380 !    1. Define the current file name root string.
390 !    2. If hardcopy device is a printer then output HPGL initialization
400 !       string.
410 !    3. Output each file which matches the file name root string to the
420 !       hardcopy device.
430 !    4. Output a form feed to the hardcopy device.
440 !
450 ! This program requires BASIC 6.0 or greater which provides the use of
460 ! wild cards with the catalog command.
470 !
480 ! The peripheral GPIB addresses and the hard copy device selection are
490 ! hard coded and may need to be changed for different systems
500 ! configurations. See lines 1130 to 1240.
```

```
510 !
520 ! EXAMP7E Reading Plot Files from a Disk
530 !
540 WILDCARDS UX;ESCAPE "\"             ! Enable UX style wild cards
550 OPTION BASE 1                       ! Set numeric arrays to start at 1
560 DIM Flnm$(1:200)[14]                ! Plot filename array
570 DIM Hpglinit$[80]                   ! Printer HPGL initialization string
580 DIM Srch$[60]                       ! Search string for plot filenames
590 DIM Esc_chr$[1]                     ! Escape character ASCII 27
600 INTEGER Plt_arry1(1:32767)          ! Plotter command array
610 INTEGER Plt_arry2(1:2,1:32767)      ! Additional plot arrays if required
620 INTEGER Plttr                       ! Plotter for output
630 INTEGER Prntr                       ! Printer for output
640 INTEGER Outputdvc                   ! Output device selected
650 INTEGER Root_mtch                   ! Root plot file flag
660 INTEGER Flnm_idx                    ! Pointer to filename array
670 INTEGER Nbr_files                   ! Number of files which are plot files
680 INTEGER Prfx_lngth                  ! Length of prefix defined in filename
690 INTEGER Root_lngth                  ! Length of root name in file name
700 INTEGER Arry1_sz                    ! Number of data works in plot file
710 INTEGER Arry2_sz                    ! Number of arrays in plot file
720 INTEGER Plttr_addr                  ! Plotter address
730 INTEGER Prntr_addr                  ! Printer address
740 REAL Rcrd_lngth                     ! Record length in plot file
750 REAL Nmbr_rcrds                     ! Number of records in plot file
760 REAL Nmbr_wrds                      ! Number of data words in plot file
770  !
780 Esc_chr$=CHR$(27)          ! Escape character (1B hex) ASCII 27 (Decimal)
790  !
800  !   ***   Build control string for printers   ***
810  !
820  ! Build hardcopy device control string containing setup commands for
830  ! printer output.
840  ! Reset, conditional page eject
850 Hpglinit$=Esc_chr$&"E"
860  ! Page size A  8.5 x 11
870 Hpglinit$=Hpglinit$&Esc_chr$&"&12A"
880  ! Landscape orientation
890 Hpglinit$=Hpglinit$&Esc_chr$&"&l1O"
900  ! No left margin
910 Hpglinit$=Hpglinit$&Esc_chr$&"&a0L"
920  ! No right margin
930 Hpglinit$=Hpglinit$&Esc_chr$&"&a400M"
940  ! No top margin
950 Hpglinit$=Hpglinit$&Esc_chr$&"&l0E"
960  ! Picture frame size 10.66 inches x 7.847 inches
970  ! (720 decipoints per inch)
980 Hpglinit$=Hpglinit$&Esc_chr$&"*c7680x5650Y"
990  ! Move cursor to anchor point
1000 Hpglinit$=Hpglinit$&Esc_chr$&"*p50x50Y"
1010 ! Set picture frame anchor point
1020 Hpglinit$=Hpglinit$&Esc_chr$&"*c0T"
1030 ! Set CMY palette
1040 Hpglinit$=Hpglinit$&Esc_chr$&"*r-3U"
1050 ! Enter HPGL mode with the cursor (pen) at the PCL current save position
1060 !
1070 ! Exit HPGL mode to accept printer command
1080 Hpgl_exit$=Esc_chr$&"%0A"
```

```
1090 !
1100 ! Conditional form feed (page eject)
1110 Form_feed$=Esc_chr$&"E"
1120 !
1130 !   ***   Initialize varibles and assign output device   ***
1140 !
1150 ! Define device selection flags to determine plotter or printer
1160 ! Select device with 1 and set Outputdvc to define it
1170 Plttr=1                        ! Select plotter for output
1180 Prntr=0                        ! Select printer for output
1190 Outputdvc=Plttr                ! define output device as plotter
1200 !
1210 False=0                        ! Define flags for logic tests
1220 True=1
1230 !
1240 Prfx$="PLOT"                   ! define plot file name prefix string
1250 !
1260 !   ***   Initialize GPIB device addresses   ***
1270 !
1280 Prntr_addr=701                 ! Printer GPIB address
1290 Plttr_addr=705                 ! Plotter GPIB address
1300 !
1310 ! Set address of flexible disk drive containing plot files
1320 Msi$=":,1400"
1330 !
1340 ! Define I/O paths for plot output with no formatting on data
1350 IF Outputdvc=Plttr THEN        ! select output device for plotting
1360  ASSIGN @Prntpltdvc TO Plttr_addr;FORMAT OFF    ! Select plotter
1370 ELSE
1380  ASSIGN @Prntpltdvc TO Prntr_slctr;FORMAT OFF   ! Select printer
1390 END IF
1400 !
1410 !   ***  Search disk for plot files  ***
1420 !
1430 ! Define the plot file name specifier:  Prefix (Prfx$), two
1440 ! sequence digits([0-9][0-9]), two output format specifier
1450 ! characters ([FLR][LPU]) and an optional character (s)
1460 Srch$=Prfx$&"[0-9][0-9][FLR][LPU]*"
1470 !
1480 ! Catalog the files that match the plot file name specifier by
1490 ! putting the names in the string array Flnm$ and the number of files
1500 ! in the integer Nbr_files. Suppress the catalog header text.
1510 CAT Srch$&Msi$ TO Flnm$(*);COUNT Nbr_files,NO HEADER,NAMES
1520 !
1530 ! If no files are found then print message and stop
1540 IF Nbr_files=0 THEN
1550  PRINT "No files found;  program terminated"
1560  STOP
1570 END IF
1580 !
1590 !   ***   Sort the plot file names found   ***
1600 !
1610 ! Re-dimension the file name array to the actual number of files that
1620 ! were found.
1630 REDIM Flnm$(1:Nbr_files)
1640 !
1650 ! Sort the file names into alphabetical order
1660 MAT SORT Flnm$(*)
```

```
1670 !
1680 GOSUB Frm_fd                     ! Send a form feed to hardcopy device
1690 PRINT
1700 !
1710 !   ***   Cycle through the filenames and plot the data   ***
1720 !
1730 Flnm_idx=1                       ! Initialize the Flnm$ array index
1740 Prfx_lngth=LEN(Prfx$)            ! Find the length of the Prfx$ string
1750 !
1760 ! Process each of the files in the Flnm$ array
1770 WHILE Flnm_idx<=Nbr_files
1780   ! Define Root$ = file name prefix string plus the two sequence digits
1790   Root$=Flnm$(Flnm_idx)[1;Prfx_lngth+2]
1800   ! Find length of the Root$
1810   Root_lngth=LEN(Root$)
1820   ! If the two output specifier characters are “FP” (full page) then
1830   ! include them as part of the Root$
1840   IF Flnm$(Flnm_idx)[Root_lngth+1;2]=”FP” THEN
1850     Root$=Root$&”FP”
1860     Root_lngth=Root_lngth+2
1870   END IF
1880   !
1890   Root_mtch=True            ! initialize file matches Root$ flag
1900   ! If the ouput device is a printer send HPGL initiliazation string
1910   IF Outputdvc=Prntr THEN
1920     OUTPUT @Prntpltdvc USING “#,K”;Hpglinit$
1930   END IF
1940   !
1950   !   ***   Plot files on the same page   ***
1960   !
1970   ! While the root portion of the file names match Root$, output the
1980   ! plot files to the same page.
1990   WHILE Root_mtch=True
2000     ! Print the name of the plot file
2010     PRINT Flnm$(Flnm_idx),
2020     ! Output the plot file to the hardcopy device
2030     GOSUB Otpt_fl
2040     ! Increment Flnm$ array index
2050     Flnm_idx=Flnm_idx+1
2060     ! If all plot files have been output or the plot file name does not
2070     ! match Root$ then set Root_Mtch=False to end plotting of the same
2080     ! page.
2090     IF Flnm_idx>Nbr_files THEN
2100       Root_mtch=False
2110     ELSE
2120       IF Root$<>Flnm$(Flnm_idx)[1;Root_lngth] THEN Root_mtch=False
2130     END IF
2140   END WHILE              ! Loop to plot to the same page
2150   !
2160   PRINT
2170   ! Output form feed to output device to eject page
2180   GOSUB Frm_fd
2190 END WHILE                ! Loop to plot next page
2200  !
2210 STOP
2220 !
2230 !**************************Subroutines ******************************
2240 !
```

```
2250 Otpt_fl:! Read the file into an array(s) and then output the array(s)
2260 ! to the hardcopy device.
2270 !
2280 ! Open and read the plot file size
2290 ASSIGN @Ldisc TO Flnm$(Flnm_idx)&Msi$
2300 !
2310 ! Get number of records in file from I/O path status registers
2320 STATUS @Ldisc,3;Nmbr_rcrds
2330 !
2340 ! Get record length
2350 STATUS @Ldisc,4;Rcrd_lngth
2360 !
2370 ! Compute the number of words of data in the file
2380 Nmbr_wrds=Nmbr_rcrds*Rcrd_lngth/2
2390 !
2400 ! Determine the number of arrays necessary to hold the plot file data.
2410 ! The maximum size of an RMB array is 32767. If the number of words of
2420 ! data is greater than 32767 then multiple arrays must be used to hold
2430 ! all of the data.
2440 !
2450 ! Compute the dimensions of Plt_arry1 and Plt_arry2 that are required
2460 ! to hold the plot data.
2470 Arry1_sz=Nmbr_wrds MOD 32767
2480 Arry2_sz=INT(Nmbr_wrds/32767)
2490 !
2500 ! Re-dimension Plt_arry1 to the correct size
2510 REDIM Plt_arry1(1:Arry1_sz)
2520 !
2530 ! If the number of words of data is less than 32767 then use one array
2540 IF Arry2_sz=0 THEN
2550   ENTER @Ldisc;Plt_arry1(*)           ! Read the plot data from the file
2560 ELSE
2570   ! Use 2 arrays to read data
2580   ENTER @Ldisc;Plt_arry2(*),Plt_arry1(*)  ! Read the plot data from file
2590 END IF
2600 !
2610 ASSIGN @Ldisc TO *                        ! Close plot file
2620 !
2630 ! Output the data to the hardcopy device
2640 IF Arry2_sz=0 THEN                        ! Only one array <32767 words
2650   OUTPUT @Prntpltdvc;Plt_arry1(*)
2660 ELSE                                     ! Data > 32767 so send 2 arrays
2670   OUTPUT @Prntpltdvc;Plt_arry2(*),Plt_arry1(*)
2680 END IF
2690 RETURN
2700  !
2710  !*************************************************************************
2720  !
2730 Frm_fd:! Send a form feed (page eject) command to the hardcopy device
2740 IF Outputdvc=Plttr THEN                   ! Plotter output
2750   OUTPUT @Prntpltdvc USING "#,K";"PG;"
2760 ELSE                                     ! For printer output. The
2770   ! printer first must exit HPGL mode before sending form feed command
2780   OUTPUT @Prntpltdvc USING "#,K";Hpgl_exit$&Form_feed$
2790 END IF
2800 RETURN
2810  !
2820 END
```

Example 7F: Reading ASCII Disk Files to the Instrument Controller's Disk File

Another way to access the analyzer's test results is to store the data onto a disk file from the analyzer. This operation generates an ASCII file of the analyzer data in a CITIFILE format. A typical file generated by Example 7F is shown below:

```
CITIFILE A.01.00
 #NA VERSION 8703B.04.13
NAME DATA
VAR FREQ MAG 11
DATA S[1,1] RI
SEG_LIST_BEGIN
SEG 100000000 200000000 11
SEG_LIST_END
BEGIN
8.30566E-1,-1.36749E-1
8.27392E-1,-1.43676E-1
8.26080E-1,-1.52069E-1
8.25653E-1,-1.60003E-1
8.26385E-1,-1.68029E-1
8.26507E-1,-1.77154E-1
8.26263E-1,-1.87316E-1
8.26721E-1,-1.97265E-1
8.2724E-1,-2.07611E-1
8.28552E-1,-2.19940E-1
8.29620E-1,-2.31109E-1
END
```

This data file is stored by the analyzer under remote control or manually from the front panel. See your analyzer’s user’s guide for more details on manual operation. This program performs the same steps that are required to manually store a file from front panel.

This program stores a file in the same manner as an operator would store a file onto the analyzer's internal disk drive from the front panel.

This example explains the process of storing the data from the analyzer to a file on the internal disk drive. There is also a program to read the data from the file into a data array for further processing or reformatting to another file type. The internal drive will store in the same format that is present on the disk. A new disk may be formatted in either LIF or DOS. For the example, the assumption has been made that the format transformation has already taken place, and there is a file that can be read record by record, from which data can be retrieved.

The goal of this example is to recover an array of stimulus frequency along with the trace-data values. CITIFILES contain the real and imaginary values of each data point. Some further transformation will be required to obtain magnitude values, for example.

The disk file contents for this example are shown above. This file contains more information than will be used in this example. The file is accessed and the records read from the file and printed on the controller display to observe the actual file contents. The file pointer is reset and the records are then read and interpreted for their data contents.

The first six records are skipped for this example. The seventh record contains the stimulus-frequency values and the number of points in the trace. These values are read from the record. The frequency increment, or point spacing, is calculated and used later in the frequency-data calculations for a point. Two more records are skipped and the next is the first record representing data values. The data values are read in a loop until the values for the number of points have been recovered from the file. The data values are tabulated and printed out on the controller display.

The following is an outline of the program's processing sequence:

- An I/O path is assigned to the analyzer.
- The system is initialized.
- A string is dimensioned to hold a file record.
- The analyzer operating state is set.
- The internal drive is selected for storage (only ASCII data is stored).
- A file name is entered and the data stored into it.
- The operator is prompted to move the disk to the controller disk drive.
- The disk file is read and the contents displayed.
- The file pointer is rewound.
- The file contents are converted to trace data.
- The frequency and complex-data pair is displayed for each point.
- The analyzer is restored to continuous-sweep mode.
- The analyzer is returned to local control and the program ends.

CAUTION Do not mistake the line switch for the disk eject button. If the line switch is mistakenly pushed, the instrument will be turned off, losing all settings and data that have not been saved.

NOTE If the command `EXTMDATOON` is used, it will override all of the other save options (such as `EXTMFORMON`). Because this type of data is only intended for computer manipulation, the file contents of a `EXTMDATOON` (data only) save cannot be recalled and displayed on the analyzer.

Running the Program

The analyzer is initialized and the operating range re-defined to an 11-point trace from 100 to 200 MHz. This setup gives a restricted range to be evaluated when the ASCII data file (CITIFILE) is read in from the controller. The operator is prompted for a 5-character filename to use for storing the data. The analyzer is setup for external storage and stores the data file. Once the “pass control/storage/return control” operation is complete, the operator is prompted to place the disk in the controller disk drive and press **Return.** The disk is then read and the records contained in the file are printed on the controller display. A prompt

appears, `Press return to continue`, which allows viewing of the file contents. Once **Return,** is pressed, the data records are read and decoded and a table of the stimulus frequency and the data values are printed.

BASIC Program Listing

```
10  ! This program shows how to store an ASCII data file in CITIFILE format
20  ! and retrieve the data with the controller. The disk is written in the
30  ! analyzer system and then moved to the controller disk and the data
40  ! accessed.
50  !
60  ! EXAMP7F Reading ASCII Disk Files to the Instrument Controller's Disk File
70  !
80  ASSIGN @Nwa TO 716                  ! Assign an I/O path for the analyzer
90  !
100 CLEAR SCREEN
110 ABORT 7                             ! Generate an IFC (Interface Clear)
120 CLEAR @Nwa                          ! SDC (Selected Device Clear)
130 OUTPUT @Nwa;"OPC?;PRES;"            ! Preset the analyzer and wait
140 ENTER @Nwa;Reply                    ! Read the 1 when complete
150 !
160 DIM Record$[80]                     ! String to read the disk records
170 !
180 ! Set up analyzer
190 OUTPUT @Nwa;"STAR100MHZ;"           ! Start frequency 100 MHz
200 OUTPUT @Nwa;"STOP 200MHZ"           ! Stop frequency 200 MHz
210 OUTPUT @Nwa;"POIN11;"               ! Trace length 11 points
220 OUTPUT @Nwa;"OPC?;SING;"            ! Single sweep and wait
230 ENTER @Nwa;Reply                    ! Read in the 1 when complete
240 !
250 ! Program disk storage operation
260 !
270 OUTPUT @Nwa;"INTD;"                 ! Select internal disk file
280 OUTPUT @Nwa;"EXTMFORMON;"           ! Store formated data
290 OUTPUT @Nwa;"EXTMDATOON;"           ! Store data file only
300 INPUT "Enter data file name (5 chars)",File_name$ ! Get file name
310 File_name$=UPC$(File_name$)         ! File names are uppercase
320 OUTPUT @Nwa;"TITF1""";File_name$;""";" ! Title for save reg 1
330 OUTPUT @Nwa;"SAVUASCI;"             ! Save as ASCII file
340 !
350 OUTPUT @Nwa;"STOR1;"                ! Store data to disk file
360 OUTPUT @Nwa;"OPC?;WAIT;"            ! Wait until store is complete
370 ENTER @Nwa;Reply
380 !
390 ! File storage is complete
400 !
410 INPUT "Place disk in controller disk drive, then press Return",A$
420 !
430 ! Read data file information
440 !
450 ASSIGN @File TO File_name$&"D1:,1400"  ! Open an I/O path for file
460 Record_cnt=1                        ! Counter to count records
470 !
480 PRINT CHR$(12);                     ! Formfeed to clear display
490 PRINT "Contents of data file"       ! Show contents of the data file
500 Readfile:   !
510 ON END @File GOTO End_file          ! Test for end of file and exit
520 ENTER @File;Record$                 ! Read ASCII record
```

```
530 PRINT Record_cnt,Record$              ! print record on display
540 Record_cnt=Record_cnt+1               ! Increment record counter
550 GOTO Readfile                         ! Read next record
560 !
570 End_file: !                           ! Reached the end of file
580 PRINT “End of File. “;Record_cnt-1;” Records found”
590 INPUT “Press Return to continue”,A$
600 PRINT CHR$(12);                       ! Formfeed to clear display
610 !
620 ! Read file data into arrays
630 !
640 RESET @File                           ! Rewind file pointer to begining
650 FOR I=1 TO 6
660   ENTER @File;Record$                 ! Skip first six records
670 NEXT I
680 ENTER @File;Record$                   ! Read frequency data record
690 Record$=Record$[POS(Record$,” “)+1]   ! skip SEG to first space + 1
700 Startf=VAL(Record$)                   ! Read start frequency
710 Record$=Record$[POS(Record$,” “)+1]   ! Skip to next space + 1
720 Stopf=VAL(Record$)                    ! Read stop frequency
730 Record$=Record$[POS(Record$,” “)+1]   ! Skip to next space +1
740 Num_points=VAL(Record$)               ! Read the number of points
750 PRINT “ Number of points in file “;Num_points
760 PRINT                                 ! White space
770 !
780 Freq_inc=(Stopf-Startf)/(Num_points-1) ! Compute frequency increment
790 !
800 ALLOCATE Array(Num_points,2)          ! Allocate array from Num_points
810 ENTER @File;Record$                   ! Skip SEG_LIST_END record
820 ENTER @File;Record$                   ! Skip BEGIN record
830 !
840 ! Read in the data array
850 PRINT “Freq (MHz)  Data 1    Data 2”  ! Table header for data array
860 FOR I=1 TO Num_points                 ! Read in array entries
870   ENTER @File;Record$                 ! Read in the record of 2 entries
880   !
890   Array(I,1)=VAL(Record$)             ! Read first data value
900   Data$=Record$[POS(Record$,”,”)+1]   ! Skip to comma and next value
910   Array(I,2)=VAL(Data$)               ! Read second data value
920   !
930   Freq=Startf+(Freq_inc*(I-1))        ! Compute stimulus value for array
940   Freq=Freq/1.E+6                     ! Convert frequency to MHz
950   !
960   PRINT Freq,Array(I,1),Array(I,2)    ! Print data array values
970 NEXT I                                ! Read next array data points
980 !
990  OUTPUT @Nwa;”CONT;”                  ! Restore continuous sweep
1000 OUTPUT @Nwa;”OPC?;WAIT;”             ! Wait for analyzer to finish
1010 ENTER @Nwa;Reply                     ! Read the 1 when complete
1020 LOCAL @Nwa                           ! Release GPIB control
1030 END
```

Limit Line and Data Point Special Functions

The analyzer has special functions in the area of limit testing and in the detection of min/max data points within limit segments. The information in this section will teach you how to use these limit line and data point special functions. The following topics are included:

- Overview
- Constants Used Throughout This Document
- Output Limit Test Pass/Fail Status Per Limit Segment
- Output Pass/Fail Status for All Segments
- Output Minimum and Maximum Point Per Limit Segment
- Output Minimum and Maximum Point For All Segments
- Output Data Per Point
- Output Data Per Range of Points
- Output Limit Pass/Fail by Channel

Overview

The limit line and data point special functions are available as remote commands only. Each command is overviewed in the following table.

Table 8-10. Limit Line and Data Point Special Functions Commands

Action	Mnemonic	Syntax	?	Description
MIN/MAX DATA DETECTION PER LIMIT SEGMENT				
Min/max recording	MINMAX<ON|OFF>	2	1,0	Enables/disables min/max recording per segment. Min and max values are recorded per limit segment.
Max values	OUTPAMAX	1		Outputs max values for all limit line segments. OUTPAMAX values and OUTPAMIN values are both output using OUTPSEGAM.
Min values	OUTPAMIN	1		Outputs min values for all limit line segments. OUTPAMIN values and OUTPAMAX values are both output using OUTPSEGAM.
Min/max values	OUTPSEGAM	1		Outputs limit test min/max for all segs. Outputs the segment number, max stimulus, max value, min stimulus, and min value for all active segments.†
Min/max value	OUTPSEGM	1		Outputs limit test min/max for a specified segment. See "SEL" on page 2-67.†
Segment	SELSEG<num>	3	<num>	Selects segment number for the OUTPSEGF and OUTPSEGM commands to report on. <num> can range from 1 to 18.†
OUTPUT TRACE DATA BY SELECTED POINTS				
Last point	SELMAXPT<num>	3	<num>	Selects the last point number in the range of points that the OUTPDATR command will report. <num> can range from 0 to the number of points minus 1.
First point	SELMINPT<num>	3	<num>	Selects the first point number in the range of points that the OUTPDATR command will report. <num> can range from 0 to the number of points minus 1.
Specify point	SELPT<num>	3	<num>	Selects the single point number that the OUTPDATP command will report. <num> can range from 0 to the number of points minus 1.
Data: point	OUTPDATP	1		Outputs a single trace data value indexed by point. (See "SELPT" on page 2-67.)
Data: range	OUTPDATR	1		Outputs trace data for range of points. (See "SELMINPT" and "SELMAXPT" on page 2-67.)
† For the definition of a limit segment, see "Example Display of Limit Lines" on page 8-127.				

Table 8-10. Limit Line and Data Point Special Functions Commands

Action	Mnemonic	Syntax	?	Description
LIMIT TEST STATUS BY CHANNEL				
Limit test: ch1	OUTPLIM1	1		Outputs status* of limit test for channel 1.
Limit test: ch2	OUTPLIM2	1		Outputs status* of limit test for channel 2.
LIMIT TEST STATUS BY SEGMENT				
Segment	SELSEG<num>	3	<num>	Selects the segment number for the OUTPSEGF and OUTPSEGM commands to report on. <num> can range from 1 to 18.†
Limit test status	OUTPSEGAF	1		Outputs the segment number and its limit test status* for all active segments.†
Limit test status	OUTPSEGF	1		Outputs the limit test status* for a specified segment. See "SELSEG" on page 2-67.†
LIMIT TEST STATUS BY POINT				
Fail report	OUTPFAIP	1		This command is similar to OUTPLIMF except that it reports the number of failures first, followed by the stimulus and trace values for each failed point in the test (note: use command LIMITEST<ON> to function properly).

† For the definition of a limit segment, see "Example Display of Limit Lines" on page 8-127.

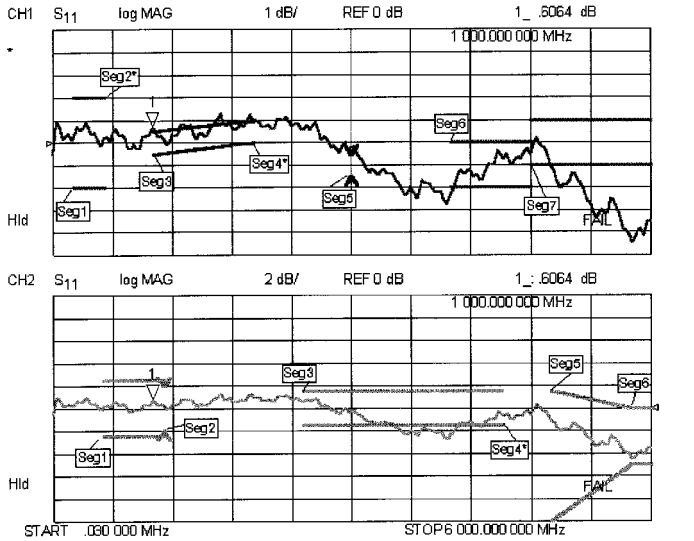
* Values returned for limit test status are: 1 (PASS), 0 (FAIL), –1 (NO_LIMIT)

Example Display of Limit Lines

The features that output data by limit segment are implemented based on the current definition of a limit segment. The actual limit lines formed by the limit table almost never have a 1-for-1 relationship with the segment numbers in the limit edit table. Out of 18 segments in the limit table, you can create 18 limit lines if (a) all limit segments are contiguous and (b) the last segment extends to the stop frequency. Otherwise, terminating a segment requires a single point which means that constructing a limit line requires two entries (segments) of the limit table. Thus you have a minimum of 9 lines available and those lines will not be referenced by sequential segment numbers.

The following figure is an example of a screen print of limit lines set up on the two instrument channels. The limit line examples shown are of Flat Line, Slope Line and Single Point Limits.

Figure 8-2. Limit Segments Versus Limit Lines



cg65d

Limit Segments

The values in the table below were used to create the limit lines in the figure.

Table 8-11. Limit Segment Table for Figure 7-3

Segment Num.	Stimulus (Frequency)	Upper Limit (dB)	Lower Limit (dB)	Limit Type
Channel 1				
1	200 MHz	2	–2	Flat Line (FL)
2*	500 MHz	2	–2	Single Point (SP)
3	1000 MHz	0.5	–0.5	Slope LIne (SL)
4*	2000 MHz	1	0	Single Point (SP)
5	3000 MHz	–0.5	–1.5	Single Point (SP)
6	4000 MHz	0	–2	Flat Line (FL)
7	4800 MHz	1	–1	Flat Line (FL)
Channel 2				
1	500 MHz	2.5	–2.5	Flat Line (FL)
2	1100 MHz	2	–2	Single Point (SP)
3	2500 MHz	1.5	–1.5	Flat Line (FL)
4*	4500 MHz	1.5	–1.5	Single Point (SP)
5	5000 MHz	1.5	–10	Slope Line (SL)
6	5800 MHz	0	–5	Slope Line (SL)

Table 8-11. Limit Segment Table for Figure 7-3

Segment Num.	Stimulus (Frequency)	Upper Limit (dB)	Lower Limit (dB)	Limit Type
* No test limit-segment is created.				

Note that if a single point limit is used to terminate slope lines, no test limit-segment is created. (See Figure 8-2: CH1, Seg4.) Also, if a single point limit is used to terminate a flat line, no test limit-segment is created. (See Figure 8-2: CH1, Seg2.) However, if the single point limit used to terminate the flat line limit has different limit values, a single-point test limit-segment is created. (See Figure 8-2: CH2, Seg2.)

Output Results

Table 8-12 shows the output of the OUTPSEGAM test (min/max of all active segments); note that the segments with asterisks (*) from Table 8-11 have no output in Table 8-12.

Table 8-12. Example Output: OUTPSEGAM (min/max of all segments)

Channel 1 Segment	Freq. at Minimum Value (Hz)	Minimum Value (dB)	Freq. at Maximum Value (Hz)	Maximum Value (dB)
1	480027600	–0.1268225	330028350	0.9590923
3	1140024300	–0.09223454	1680021600	1.258809
5	3000015000	–0.2199298	3000015000	–0.2199298
6	4020009900	–2.203248	4770006150	–0.2444123
7	5820000900	–4.473375	4860005700	0.23913
Channel 2 Segment				
1	780026100	–0.2838693	990025050	0.6258904
2	1110024450	0.2364199	1110024450	0.2364199
3	3960010200	–2.745585	2640016800	0.888033
5	5790001050	–4.136453	5010004950	–1.064739
6	5820000900	–4.472594	6000000000	–3.501008

Constants Used Throughout This Document

NOTE The logic values attached to pass and fail indicators were chosen to be consistent with the current logic used in the standard OUTPLIML and OUTPLIMF commands.

Table 8-13. Pass/Fail/No_Limit Status Constants

Status Definition	Status Indicator
PASS	1
FAIL	0
NO_LIMIT	–1

Table 8-13 is an interpretation of the Pass/Fail/No_Limit status constants. These constants are used to identify the Pass/Fail/No_Limit state on the data strings if status is returned.

Table 8-14. Min/Max Test Constants

String	Stimulus Value	Data Value
NO_DATA	0	–1000

Table 8-14 is an interpretation of the min/max test constants. If the selected segment has no associated limit, the NO_DATA string is generated, which reports a stimulus value of 0 and a data value of –1000.

Output Limit Test Pass/Fail Status Per Limit Segment

Two commands allow you to query the pass/fail test status on a limit segment basis.

- SELSEG<num> will select the segment.
- OUTPSEGF will return the status of the limit test for that segment: 1 (PASS), 0 (FAIL) or –1 (NO_LIMIT) if no limit exists for the selected segment number. Due to the non-sequential numbering of actual limit line segments on the screen, some segment numbers will have no associated limits and will thus return –1 (NO_LIMIT).

  Under the following conditions, OUTPSEGF will issue the following errors:

  If the limit testing is off: “30: Requested Data Not Currently Available.” To clear the error message, turn the limit test on.

  If the limit table is empty: “204: Limit Table Empty” (this is a new message). To clear the error message, enter a limit table.

  In both cases, the error is issued and the command responds with –1 (NO_LIMIT).

The argument for SELSEG<num> is limited by the maximum number of segments allowed in the limit table, which is currently 22. The minimum value for the argument is 1. If the user inputs a number that is outside this range, the active entry limits are invoked, causing the analyzer to return the status for limit 22.

Example:

Sending SELSEG3 and OUTPSEGF may return the following:

1 (segment number 3 passed)

NOTE The output is ASCII. Currently, the formatting for integer numbers appears to append a trailing space.

Output Pass/Fail Status for All Segments

The GPIB command OUTPSEGAF will return the number of segments being reported, followed by pairs of data consisting of the segment number and its status. A segment is reported only if it has an associated limit. The output is only valid if limit test is on. See "Output Limit Test Pass/Fail Status Per Limit Segment" on page 8-130.

Example:

Sending OUTPSEGAF may return the following:

3
1, 0
3, 1
5, 0

For an explanation of these results, see Table 8-15.

NOTE A new Line Feed character $<^{L}_{F>}$ is inserted after the number of segments and after each data pair.

Table 8-15. Example Output: OUTPSEGAF (pass/fail for all segments)

Segments Reported	Segment Number	Status	Status Definition
3			
	1	0	FAIL
	3	1	PASS
	5	0	FAIL

Table 8-15 is an interpretation of the data returned by the command OUTPSEGAF. For clarification, status definition is also included.

Example Program of OUTPSEGAF Using BASIC

The following program is not included on the “Programming Examples” CD-ROM:

```
10 OUTPUT 716; "outpsegaf;"

20 ENTER 716; Numsegs

30 PRINT "Receiving status for"; Numsegs; "segments."

40 IF Numsegs>0 THEN

50    FOR I=1 TO Numsegs

60      ENTER 716; Segnum, Pf

70      PRINT USING "DD, 2X, 8A"; Segnum, Pf

80    NEXT I
```

The example program shows how the OUTPSEGAF command can be used to request the number of active segments and their status. Notice that each segment result must use a new enter command as a line feed terminates each segment data result.

Output Minimum and Maximum Point Per Limit Segment

The command MINMAX<ON|OFF> toggles a feature which records the minimum and maximum data points in all active limit segments. Note that limit testing need not be turned on.

The command OUTPSEGM will report the min/max data for the segment previously selected by SELSEG. The data is returned in a comma delimited string with the segment number, minimum point stimulus, minimum trace value, maximum point stimulus and maximum trace value.

Under the following conditions, OUTPSEGM will issue the following errors:

- If the min/max testing is off: “30: Requested Data Not Currently Available.” To clear the error message, turn the min/max testing on.
- If the limit table is empty: “204: Limit Table Empty” (this is a new message). To clear the error message, enter a new limit table.

  When the above error conditions occur, there is no data to report, thus no output is generated.

  If the selected segment has no associated limit, the NO_DATA string is generated, which reports a stimulus value of 0 and a data value of –1000.

Example:

Sending SELSEG3 and OUTPSEGM may return the following:

3, 1.900000000E+09, –9.900000E–01, 2.123456789E+09, 2.123456E+00

For an explanation of these results, see Table 8-16.

Table 8-16. Example Output: OUTPSEGM (min/max per segment)

Segment	Min Pt Stimulus (Frequency)	Min Pt Value (dB)	Max Pt Stimulus (Frequency)	Max Pt Value (dB)
3	1.9 GHz	–.99	2.12 GHz	2.12

Table 8-16 is an interpretation of the min/max data returned using the SELSEG and OUTPSEGM commands.

NOTE A new Line Feed character <$^{L}_{F>}$ is inserted after the segment number and after each data pair.

Output Minimum and Maximum Point for All Segments

Three GPIB commands allow the user to dump the min-or-max or min-and-max values for all active segments:

- OUTPSEGAM: outputs min and max data for each active segment.
- OUTPAMIN: outputs the min data for each active segment.
- OUTPAMAX: outputs the max data for each active segment.

The OUTPSEGAM output consists of:

- The total number of segments being reported.
- The following data for each segment:

  segment number
  min stimulus
  min value
  max stimulus
  max value

Example:

Sending OUTPSEGAM may return the following:

5,

1, 1.900000000E+09, –9.900000E–01, 2.123456789E+09, 2.123456E+00

3, 2.300000000E+09, –10.00000E–01, 2.600000000E+09, 3.100000E+00

5, 3.200000000E+09, –10.00000E–01, 3.400000000E+09, 3.100000E+00

7, 4.300000000E+09, –10.00000E–01, 4.700000000E+09, 3.100000E+00

8, 5.000000000E+09, –10.00000E–01, 5.400000000E+09, 3.100000E+00

For an explanation of these results, see Table 8-17.

NOTE A new line feed character <L_F> is inserted after the segment number and after each data pair.

Table 8-17. Example Output: OUTPSEGAM (min/max for all segments)

Segments Reported	Segment Number	Min Pt Stimulus (Frequency)	Min Pt Value (dB)	Max Pt Stimulus (Frequency)	Max Pt Value (dB)
5					
	1	1.9 GHz	–.99	2.12 GHz	2.12
	3	2.3 GHz	–1.0	2.6 GHz	3.1
	5	3.2 GHz	–1.0	3.4 GHz	3.1
	7	4.3 GHz	–1.0	4.7 GHz	3.1
	8	5.0 GHz	–1.0	5.4 GHz	3.1

Table 8-17 is an interpretation of the min/max data returned using the OUTPSEGAM command.

Example Program of OUTPSEGAM Using BASIC

The following program is not included on the Programming Examples CD-ROM:

```
10 Minmax:                !

20 Mm:  IMAGE DD,":",2X,D.DDDE,2X,SD.DDDE,2X,D.DDDE,2X,SD.DDDE

30      PRINT "TESTING: OUTPSEGAM: min/max points for each segment"

40      OUTPUT 716;"minmaxon;"

50      OUTPUT 716;"outpsegam;"

60      ENTER 716;Numsegs

70      PRINT "receiving data for";Numsegs;"segments"

80      FOR I=1 TO Numsegs

90          ENTER 716;Segnum,Minstim,Minval,Maxstim,Maxval

100         PRINT USING Mm; Segnum, Minstim, Minval,Maxstim,
            Maxval

110     NEXT I
```

Output Data Per Point

The GPIB command OUTPDATP returns the value of the selected point using FORM4 (ASCII). The point is selected using the SELPT command. This returns the last point if the selected point is out of range. Otherwise, it uses the same format as that used by the marker value command. These formats are as follows:

Table 8-18. Example Output: OUTPDATP (data per point)

Display Format	Marker Mode	Marker Readout Format	Example Returns	
Log Mag		dB*	–3.521 (dB)	9.7E–39*
Phase		degrees*	157.8 (Deg)	5.3E–15*
Delay		seconds*	0.5068x10–9	0*
Smith Chart	LIN MKR	lin mag, degrees		
	LOG MKR	dB, degrees		
	Re/Im	real, imag		
	R + jX	real, imag ohms	10.37 Ω	9.399 Ω
	G + jB	real, imag Siemens		
POLAR	LIN MKR	lin mag, degrees	0.6667	157.8 (Deg)
	LOG MKR	dB, degrees	–3.521 (dB)	157.8 (Deg)
	Re/Im	real, imag	–0.6173	0.2518
LIN MAG		lin mag *	0.6667	0*
REAL		real *		
SWR		SWR *	5.001	0*
* Value is insignificant, but is included in data transfers.				

The commands in the following example are sent while using the format command LOGM.

Example:

Sending SELPT5 and OUTPDATP may return the following:

–3.513410E+00, 0.00915E+15 (Note that the second number is insignificant.)

Output Data Per Range of Points

The GPIB command OUTPDATR returns the value of the selected points using FORM4 (ASCII). This ASCII format requires many data bytes per point for transfer. For a large number of points, it may be faster to make trace data dumps (OUTPDATA) using a binary format. The range of points is selected using the SELMINPT and SELMAXPT commands (select minimum point, select maximum point of desired point range). These commands return the last max point if the selected points are out of range. Only the SELMAXPT will be returned if the selected minimum point is greater than the selected maximum point.

The commands in the following example are sent while using the format command LOGM.

Example:

Sending SELMINPT5, SELMAXPT7 and OUTPDATR may return the following:

3.880465E–01, 0.000039E–01

1.901648E–01, 1.363988E+11

5.57587E–01, 1.258655E+30 (Note that the second number is insignificant.)

For an explanation of these results see Table 8-19.

NOTE A new line feed character <L_F> is inserted after the segment number and after each data pair.

Table 8-19. Example Output: OUTPDATPR (data per range of points)

Point	Value	Value*
5	0.3880465	0.000039E–01
6	0.1901648	1.363988E+11
7	0.557587	1.258655E+30
* These values are insignificant.		

Table 8-19 is an interpretation of the min/max data per range of points returned using the SELMINPT5, SELMAXPT7 and OUTPDATR commands.

Output Limit Pass/Fail by Channel

The GPIB commands OUTPLIM1 and OUTPLIM2 output the status of the limit test for channel 1 and channel 2, respectively.

These commands return the values 1 (PASS), 0 (FAIL), or –1 (NO_LIMIT) if limit testing is disabled. Currently, the results of limit testing can be retrieved by reading a bit in the status register.

Example:

Sending OUTPLIM1 or OUTPLIM2 (channel 1 or channel 2) may return the following:

1 (PASS), 0 (FAIL), or if limit test not enabled then –1 (NO_LIMIT).

Index

Index

Index