Errata

Title & Document Type: HP 8719A, 8720A/B Microwave Network Analyzers HP-IB Programming Guide

Manual Part Number: 08719-90030

Revision Date: 1989-05-01

HP References in this Manual

This manual may contain references to HP or Hewlett-Packard. Please note that Hewlett-Packard's former test and measurement, semiconductor products and chemical analysis businesses are now part of Agilent Technologies. We have made no changes to this manual copy. The HP XXXX referred to in this document is now the Agilent XXXX. For example, model number HP8648A is now model number Agilent 8648A.

About this Manual

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

Support for Your Product

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

www.tm.agilent.com

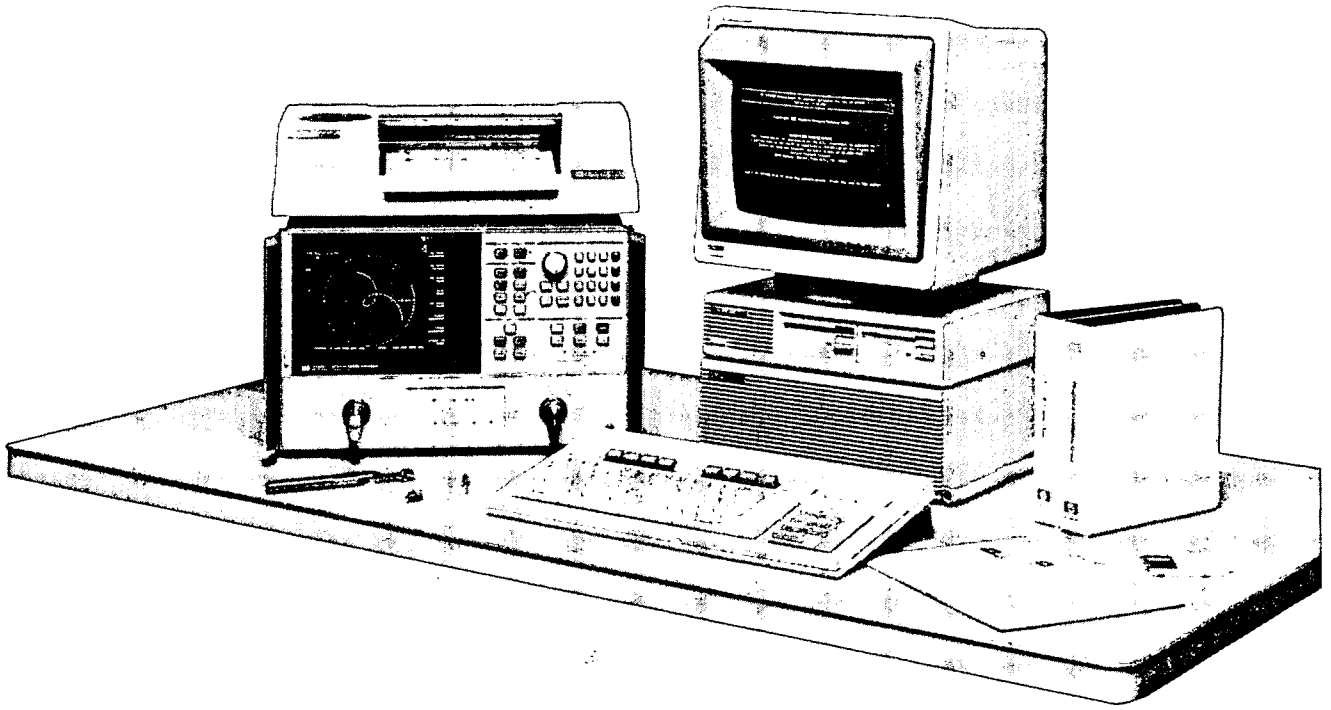
Search for the model number of this product, and the resulting product page will guide you to any available information. Our service centers may be able to perform calibration if no repair parts are needed, but no other support from Agilent is available.

Agilent Technologies

HP-IB Programming Guide

HEWLETT PACKARD

For the HP 8719A and 8720A/B Microwave Network Analyzers with the HP 9000 series 200/300 desktop computer (BASIC)



Introduction

This programming note is an introduction to remote operation of the HP 8720A, 8720B, and 8719A network analyzers using an HP 9000 series 200 or 300 computer. It is a tutorial introduction, using BASIC programming examples to demonstrate remote operation. The examples are on the Example Programs disk (part number 08720-20003), included with the network analyzer operating manual. This document is associated with the HP 8700 family *HP-IB Quick Reference*. The *HP-IB Quick Reference* provides complete programming information in a very concise format. Included in the *HP-IB Quick Reference* are both functional and alphabetical lists of HP-IB commands. The instrument *Quick Reference* lists HP-IB commands, along with its softkey menu explanations.

The Hewlett-Packard computers specifically addressed are the HP 9000 series 200 and 300 computers, operating with BASIC 2.0 with AP2—1, or BASIC 3.0 or higher. This includes the 216 (9816), 217 (9817), 220 (9920), 226 (9826), 236 (9836), 310 and 320 computers.

The reader should become familiar with the operation of the network analyzer before controlling it over HP-IB. Also, this document is not intended to teach BASIC programming or to discuss HP-IB theory except at an introductory level. See page 2 for documents better suited to these tasks.

For more information

For more information concerning the operation of the instrument, refer to its:

User's Guide
Quick Reference
Operating Manual

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

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

For more information concerning HP-IB, see:

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

Table of Contents

Required equipment

To run the examples, the following equipment is required:

1. HP 8700 family microwave network analyzer.
2. HP 9000 series 200 or 300 computer with enough memory to hold BASIC, needed binaries, and at least 64 kBytes of program space. In addition, 512 kBytes are needed for BASIC 3.0 or higher operating systems, with the binaries suggested in step 2 in the section *Powering up the system*. A disk drive (e.g. HP 9122) is required to load BASIC if no internal disk drive is available.
3. HP BASIC 2.0 with AP2—1, or BASIC 3.0 or higher.
4. HP 10833A/B/C/D HP-IB cables to interconnect the computer, the analyzer, and any peripherals.

Optional equipment

1. Calibration kit.
2. Test port return cables.
3. The bandpass filter supplied with the instrument (part number 0955-0446) for use as a test device in the example measurement programs.
4. HP 7440A ColorPro plotter, an HP 2225A ThinkJet printer, or an HP 9122 or HP 9153 CS80 disk drive. See the "General Information" section of the manual for a more complete list of compatible peripherals.

Powering up the system

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

2. **Turn on the computer and load the BASIC operating system.** For BASIC 2.0, load AP2—1 if available. If BASIC 3.0 or higher is used, load the following BASIC binary extensions: HPIB, GRAPH, IO, KBD, and ERR. Depending on the disk drive, a binary such as CS80 may be also be required.

3. **Turn the network analyzer on.** To verify the instrument HP-IB address, press **[LOCAL]**, *[SET ADDRESSES]*, and *[ADDRESS: ANALYZER]*. If the address has been changed from 16, the default value, return it to 16 while performing the examples in this document by pressing **[1] [6] [x1]** and then presetting the instrument. Make sure the instrument is in either *[USE PASS CONTROL]* or *[TALKER/LISTENER]* mode, as indicated under the **[LOCAL]** key. These are the only modes in which the network analyzer will accept commands over HP-IB.

4. **On the computer, type the following:** `OUTPUT 716;"PRES;"` **[EXECUTE]** (or **[RETURN]**) This will preset instrument. If Preset does not occur, there is a problem. First check all HP-IB addresses and connections: most HP-IB problems are caused by an incorrect address and bad or loose HP-IB cables.

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

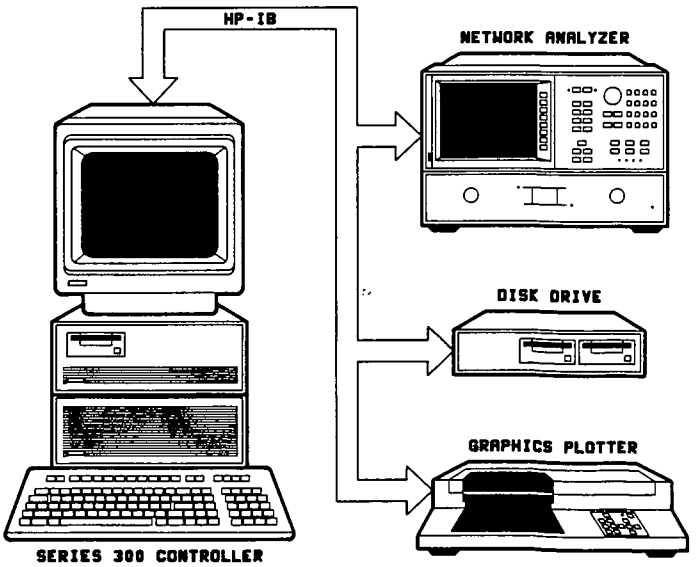


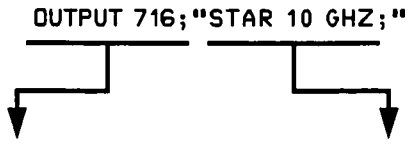
Figure 1. HP-IB connections in a typical setup

Basic Instrument Control

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

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

The start frequency is set to 10 GHz. The construction of the command is:



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

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

The `STAR 10 GHZ;` command performs the same function as pressing **[START]** and keying in **10 [G/n]**. `STAR` is the root mnemonic for the start key, `10` is the data, and `GHZ` are the units. The root mnemonics are derived from the equivalent key label where possible, otherwise from the common name for the function. The *Quick Reference* lists all the root mnemonics, and all the different units that are accepted.

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

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

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

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

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

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

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

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

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

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

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

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

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

The debug mode can be used to aid in trouble-shooting systems. When debug mode is on, incoming HP-IB commands scroll across the display. To turn the mode on manually, press **[LOCAL]**, *[HP-IB DIAG ON]*. To turn it on over HP-IB, execute:

```
OUTPUT 716;"DEBUON;"
```

Command interrogate

Suppose the operator has changed the power level from the front panel. The computer can find out the new power level using the command interrogate function. If a question mark is appended to the root of a command, the value of that function is output. For instance, `POWE -20 DB;` sets the output power to −20 dB, and `POWE?;` outputs the current RF output power at the test port. For example, type **SCRATCH [EXECUTE]**, to clear old programs, and **EDIT [EXECUTE]** to get into the edit mode. Then type in:

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

Run the program. The computer will display the test port power level in dBm. The preset test port power level is −10 dBm. Change the power level by pressing **[LOCAL]**, **[MENU]**, *[POWER]* and then entering **[-] [1] [5] [x1]**. Run the program again. Note that the test port power can range from −65 to −10 dBm, in 5 dB steps. If a command to set the power level to a value which is not an increment of 5 dB is sent, the analyzer will set the next closest power level. For example, sending `POWE -12 DB` will result in the test port power level being set to −10 dBm.

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

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

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

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

For instance, rewrite line 10 as:

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

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

Another example is to rewrite line 10 as:

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

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

Held commands

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

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

Operation complete

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

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

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

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

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

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

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

Preparing for HP-IB control

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

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

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

```
10  ABORT 7
```
This halts all bus action and gives active control to the computer.

```
20  CLEAR 716
```
This clears all HP-IB errors, resets the HP-IB interface, clears syntax errors. It does not affect the status reporting system.

```
30  OUTPUT 716;"PRES;"
```
Preset the instrument. This clears the status reporting system, as well as resetting all of the front panel settings, except the HP-IB mode and the HP-IB addresses.

```
40  END
```

Running this program brings the instrument to a known state, ready to respond to HP-IB control.

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

```
REMOTE 716
```

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

The local key can also be disabled with the sequence:

```
REMOTE 716
LOCAL LOCKOUT 7
```

Now no front panel keys will respond at all. The network analyzer can be returned to local mode temporarily with:

```
LOCAL 716
```

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

```
LOCAL 7
```

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

Measurement Programming

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

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

Set up the instrument

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

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

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

Calibrate

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

There are several ways to calibrate the instrument. The simplest is to stop the program and have the operator perform the calibration from the front panel. Alternatively, the computer can be used to guide the operator through the calibration, as discussed in Example 2A and 2B, *S11 1-port calibration* and *Full 2-port calibration.* The last option is to transfer calibration data from a previous calibration back into the instrument, as discussed in Example 6C, *Reading calibration data.*

Connect device

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

Take data

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

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

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

Post process data

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

Transfer data:

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

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

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

Basic Programming Examples

Example 1: Setting up a basic measurement

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

This example illustrates how a basic measurement can be set up. The sequence will be to first select the desired S-parameter, the measurement format, and then the frequency range. Performing calibrations is described later.

Since the network analyzer has a frequency resolution of 100 kHz, it is required that all of the data points in the sweep be at some integer multiple of 100 kHz (130.1 MHz, 2.0011 GHz, 19.9985 GHz for example). Therefore, the actual frequencies that are set may be slightly different from those specified by the operator in lines 70 and 80. By interrogating the analyzer to determine the actual values of the start and stop frequencies, the computer can keep track of the actual frequencies. Note that if you are using a network analyzer which has 1 Hz frequency resolution, (Option 001), the actual frequencies set by the analyzer will be identical to those specified using the `START` and `STOP` commands in all cases, and it is not necessary to interrogate the analyzer to determine the actual start and stop frequencies.

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

```
10  ABORT 7
20  CLEAR 716                          Prepare for HP-IB control.
30  OUTPUT 716;"PRES;"                 Preset the network analyzer.
40  OUTPUT 716;"CHAN1; S11;            Make channel 1 the active channel, and measure
    LOGM;"                             S11, displaying its magnitude in dB.
50  OUTPUT 716;"CHAN2; S11;            Make channel 2 the active channel, and measure
    PHAS;"                             the phase of S11 on it.
60  OUTPUT 716;"DUACON;"               Tell the analyzer to display both channels
                                       simultaneously.
70  INPUT "ENTER START                 Input a start frequency.
    FREQUENCY
    (GHz):",F_start
80  INPUT "ENTER STOP                  Input a stop frequency.
    FREQUENCY (GHz):",F_stop
90  OUTPUT 716;"STAR";                 Set the start frequency to F_start.
    F_start;"GHZ;"
100 OUTPUT 716;"STOP";                 Set the stop frequency to F_stop.
    F_stop;"GHZ;"
110 OUTPUT 716;"STAR?;"                Interrogate the value of the start frequency.
120 ENTER 716;Start_freq               Read the value into the variable Start_freq.
130 OUTPUT 716;"STOP?;"                Interrogate the value of the stop frequency.
140 ENTER 716;Stop_freq                Read the value into the variable Stop_freq.
150 DISP Start_freq,                   Show the current start and stop frequencies.
    Stop_freq
160 END
```

Running the program

The program will set up a measurement of S_{11}, log magnitude on channel 1, and S_{11}, phase on channel 2, and turn on the dual channel display mode. When prompted for start and stop frequencies, enter any value in GHz from 0.13 (130 MHz) to 13.5 GHz or 20 GHz. These will be entered, and the actual frequencies that the analyzer is set to are then sent back to the computer and displayed. Try entering frequencies with more than 100 kHz resolution, and note the value actual by set.

Performing a measurement calibration

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

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

Calibration kits

The calibration kit definition tells the network analyzer what standards to use 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 an S11 1-port calibration is one calibration step. All of the shorts that can be used for this calibration step make up the class, which is called class S11B. For the 7 mm and the 3.5 mm cal kits, class S11B has only one standard in it. For type-N cal kits, class S11B has two standards in it: male and female shorts.

When doing an S11 1-port calibration in 7 or 3.5 mm, selecting *[SHORT]* automatically measures the short because there is only one standard in the class. When doing the same calibration in type-N, selecting *[SHORTS]* brings up a second menu, allowing the user to select which standard in the class is to be measured. The sex listed refers to the test port: if the test port is female, then the user selects the female short option.

Doing an S11 1-port calibration over HP-IB is very similar. In 7 or 3.5 mm, sending `CLASS11B` will automatically measure the short. In type-N, sending `CLASS11B` brings up the menu with the male and female short options. To select a standard, use `STANA` or `STANB`. The `STAN` command is appended with the letters A through G, corresponding to the standards listed under softkeys 1 through 7, softkey 1 being the topmost softkey.

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

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

Full 2-port calibrations

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

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

Example 2A: S_{11} 1-port calibration

To demonstrate coordinating a calibration over HP-IB, the following program does an S11 1-port calibration, using the HP 85052B 3.5 mm calibration kit. This program simplifies the calibration for the operator by giving explicit directions on the network analyzer display, and allowing the user to continue the program by pressing any key on the network analyzer front panel.

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

```
10  ABORT 7
20  CLEAR 716                          Prepare for HP-IB control.
30  OUTPUT 716;"CALK35MM;              This is the minimum instrument set up: the 3.5 mm
    MENUOFF;CLES;ESE 64;"              cal kit is selected, the softkey menu is turned off,
                                       and the status reporting system is set up so that bit
                                       6, User Request, of the event status register, is
                                       summarized by bit 5 of the status byte. This allows
                                       us to detect a key press with a serial poll. Refer to
                                       Appendix A.
40  OUTPUT 716;"CALIS111;"             Open the calibration by calling the S11 1-port
                                       calibration.
50  CALL Waitforkey("CONNECT           Now ask for the open, and wait for the operator.
    OPEN AT PORT 1")                   The Waitforkey subroutine will not return until the
                                       operator presses a key on the front panel of the
                                       network analyzer.
60  OUTPUT 716;"OPC?;                  There is only one choice in this class, so the CLASS
    CLASS11A;"                         command is OPC'able. Using the OPC? command
                                       causes the program to wait until the standard has
                                       been measured before continuing. This is very
                                       important, because the prompt to connect the next
                                       standard should only appear after the first standard
                                       is measured.
70  ENTER 716;Reply                    Wait until the standard is measured.
80  CALL Waitforkey("CONNECT           Ask for a short, and wait for the operator to connect
    SHORT AT PORT 1")                  it.
90  OUTPUT                             Measure the short.
    716;"OPC?;CLASS11B;"
100 ENTER 716;Reply                    Wait for the standard to be measured.
110 CALL Waitforkey("CONNECT           Have the operator connect the lowband load and
    LOWBAND LOAD AT PORT 1")           wait for his reply.
120 OUTPUT 716;"CLASS11C:              There is more than one standard in the loads class,
    OPC?; STANC;"                      so we must identify the specific standard within that
                                       class. The lowband load is the third softkey
                                       selection from the top in the menu, so select a
                                       lowband load as the standard using the command
                                       STANC.
130 ENTER 716;Reply                    Wait for the standard to be measured.
140 CALL Waitforkey("CONNECT           Have the operator connect the sliding load, and wait
    SLIDING LOAD AT PORT 1")           for his reply.
150 OUTPUT 716;"STANB;"                The sliding load selection is the second softkey from
                                       the top in the menu, so we select a sliding load as
                                       the standard using the command STANB.
160 FOR I=1 TO 5                       It will require five different positions of the sliding
                                       load to properly characterize the directivity error
                                       term.
170 CALL Waitforkey("SET SLIDE         Send the prompt to set the slide, and wait for the
    IN POSITION")                      operator.
```

```
180 OUTPUT 716;"SLIS"
```
Measure the load in its current position.

```
190 NEXT I
```
Repeat the process until the sliding load has been measured in five different positions.

```
200 OUTPUT 716;"SLID;"
```
Tell the analyzer that the sliding load calibration has been completed.

```
210 OUTPUT 717;"PG;"
```
The PG command sent to the display clears the user graphics, removing the last prompt.

```
220 DISP "COMPUTING
    CALIBRATION
    COEFFICIENTS"
230 OUTPUT
    716;"DONE;OPC?;SAV1;"
```
Affirm the completion of the calibration, and save the calibration.

```
240 ENTER 716;Reply
```
Wait until the instrument is finished calculating the calibration coefficients before allowing the program to go on.

```
250 DISP "DONE"
260 OUTPUT 716;"MENUON;"
```
The calibration is completed, so turn the soft key menu back on.

```
270 END
280 SUB Waitforkey(Lab$)
```
This subroutine displays the passed message on the network analyzer display, and waits for the operator to press a key. It assumes that bit 6, User Request, of the event status register has been enabled.

```
290 DISP Lab$
```
First, display a message on the computer in case the operator has returned to the computer keyboard.

```
300 OUTPUT 717;"PG;PU;PA390,
    3600;PD;LB";Lab$;",
    PRESS ANY KEY WHEN
    READY;"
```
This statement writes on the network analyzer display. PG (page) clears old user graphics. PU (pen up) prevents anything from being drawn. PA390,3600; moves the logical pen to just above the message area on the display. PD (pen down) enables drawing. LB (label) writes the message on the display. The label command is terminated with an ETX symbol, which is **[CTRL] [C]** (pressed simultaneously) on the keyboard.

```
310 CLEAR 716
```
Clear the message line.

```
320 OUTPUT 716;"ESR?;"
```
Clear the latched User Request bit so that old key presses will not trigger a measurement.

```
330 ENTER 716;Estat
340 Stat=SPOLL(716)
```
Now wait for a key press to be reported.

```
350 IF NOT BIT(Stat,5) THEN
    GOTO 340
360 SUBEND
```

Running the program

The program assumes that the test port being calibrated is a 3.5 mm, either male or female. The prompts appear just above the message line on the network analyzer display. Pressing any key on the instrument front panel continues the program and measures the standard. The program will display DONE when the measurement calibration is complete.

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

Example 2B: Full 2-port measurement calibration

The following example shows how to perform a full 2-port measurement calibration using the HP 85052D calibration kit. The main difference between this example and example 2A is that in this case, the calibration process allows removal of both the forward and reverse error terms, so that all four S-parameters of the device under test can be measured. Also, since the HP 85052D calibration kit is used, a broadband load will be used instead of the sliding load used in Example 2A. Use of the broadband load results in a more convenient calibration, since only one measurement is required for the load calibration, as opposed to five measurements when the sliding load is used.

This example program is stored on the Example Programs disk as **IPG2B**.

```
10  ABORT 7
20  CLEAR 716                          Prepare for HP-IB control.
30  OUTPUT 716;"CALK35MM;              This is the minimum instrument set up: the 3.5 mm
    MENUOFF;CLES;ESE 64;"              cal kit is selected, the softkey menu is turned off,
                                       and the status reporting system is set up so that bit
                                       6, User Request, of the event status register, is
                                       summarized by bit 5 of the status byte. This allows
                                       us to detect a key press with a serial poll. Refer to
                                       Appendix A.
40  OUTPUT 716;"CALIFUL2;"             Open the calibration by calling for a full 2-port
                                       calibration.
50  OUTPUT 716;"REFL;"                 Open the reflection calibration subsequence.
60  CALL Waitforkey("CONNECT           Now ask for the open, and wait for the operator.
    OPEN AT PORT 1")                   The Waitforkey subroutine will not return until the
                                       operator presses a key on the front panel.
70  OUTPUT 716;"OPC?;                  There is only one choice in this class, so the CLASS
    CLASS11A;"                         command is OPC'able. Using the OPC? command
                                       causes the program to wait until the standard has
                                       been measured before continuing. This is very
                                       important, because the prompt to connect the next
                                       standard should only appear after the first standard
                                       is measured.
80  ENTER 716;Reply                    Wait until the standard is measured.
90  CALL Waitforkey("CONNECT           Ask for a short, and wait for the operator to connect
    SHORT AT PORT 1")                  it.
100 OUTPUT 716;"OPC?;                  Measure the short.
    CLASS11B;"
110 ENTER 716;Reply                    Wait for the standard to be measured.
120 CALL Waitforkey("CONNECT           Have the operator connect the broadband load, and
    BROADBAND LOAD AT PORT             wait for his reply.
    1")
130 OUTPUT 716;"CLASS11C;              There is more than one standard in the loads class,
    OPC?;STANA;"                       so we must identify the specific standard within that
                                       class. The broadband load selection is the first
                                       softkey from the top in the menu, so we select a
                                       broadband load as the standard using the command
                                       STANA.
140 ENTER 716;Reply                    Wait for the standard to be measured.
150 CALL Waitforkey("CONNECT           Ask for the open for port 2, and wait for the
    OPEN AT PORT 2")                   operator.
160 OUTPUT                             Measure the open.
    716;"OPC?;CLASS22A;"
170 ENTER 716;Reply                    Wait until the standard is measured.
```

```
180 CALL Waitforkey("CONNECT
    SHORT AT PORT 2")
190 OUTPUT
    716;"OPC?;CLASS22B;"
200 ENTER 716;Reply
210 CALL Waitforkey("CONNECT
    BROADBAND LOAD AT PORT
    2")
220 OUTPUT 716;"CLASS22C;
    OPC?;STANA;"
230 ENTER 716;Reply
240 OUTPUT 716;"REFD;"
250 DISP "COMPUTING
    REFLECTION CALIBRATION
    COEFFICIENTS"
260 OUTPUT 716;"TRAN;"
270 CALL Waitforkey("CONNECT
    THRU [PORT 1 TO PORT 2]")
280 DISP "MEASURING FORWARD
    TRANSMISSION"
290 OUTPUT 716;"OPC?;FWDT;"
300 ENTER 716;Reply
310 OUTPUT 716;"OPC?;FWDM;"
320 ENTER 716;Reply
330 DISP "MEASURING REVERSE
    TRANSMISSION"
340 OUTPUT 716;"OPC?;REVT;"
350 ENTER 716;Reply
360 OUTPUT 716;"OPC?;REVM;"
370 ENTER 716;Reply
380 OUTPUT 716;"TRAD;"
390 CALL Waitforkey("ISOLATE
    TEST PORTS")
400 OUTPUT 716;"ISOL;"
410 !OUTPUT 716;"OMII;"
420 !GOTO 500
430 DISP "MEASURING REVERSE
    ISOLATION"
440 OUTPUT 716;"OPC?;REVI;"
450 ENTER 716;Reply
460 DISP "MEASURING FORWARD
    ISOLATION"
470 OUTPUT 716;"OPC?;FWDI;"
480 ENTER 716;Reply
490 OUTPUT 716;"ISOD;"
500 OUTPUT 717;"PG;"
```

Ask for a short, and wait for the operator to connect it.

Measure the short.

Wait for the standard to be measured.

Have the operator connect the broadband load, and wait for his reply.

Measure the broadband load.

Wait for the standard to be measured.

Close the reflection calibration subsequence.

Open the transmission calibration subsequence.

Measure forward transmission.

Measure forward load match.

Measure reverse transmission.

Measure reverse load match.

Close the transmission calibration subsequence.

Ask operator to isolate the test ports.

Open the isolation calibration subsequence.

Omit isolation calibration if it is not desired.

Skip the isolation calibration steps.

Measure reverse isolation.

Measure forward isolation.

Close the isolation calibration subsequence.

The PG command sent to the display clears the user graphics, removing the last prompt.

```
510 DISP "COMPUTING
    CALIBRATION
    COEFFICIENTS"
520 OUTPUT 716;"OPC?;SAV2;"
530 ENTER 716;Reply
```

Wait until the analyzer calculates the calibration coefficients before allowing the program to go on.

```
540 DISP "DONE"
550 OUTPUT 716;"MENUON;"
```

The calibration is completed, so turn the soft key menu back on.

```
560 END
570 SUB Waitforkey(Lab$)
```

This subroutine displays the passed message on the network analyzer, and waits for the operator to press a key. It assumes that bit 6, User Request, of the event status register has been enabled.

```
580 DISP Lab$
```

First, display a message on the computer in case the operator has returned to the computer keyboard.

```
590 OUTPUT 717;"PG;PU;PA390,
    3600;PD;LB";Lab$;",
    PRESS ANY KEY*;"
```

This statement writes on the network analyzer display. PG (page) clears old user graphics. PU (pen up) prevents anything from being drawn. PA390,3600; moves the logical pen to just above the message area on the display. PD (pen down) enables drawing. LB (label) writes the message on the display. The label command is terminated with an ETX symbol, which is **[CTRL] [C]** (pressed simultaneously) on the keyboard.

```
600 CLEAR 716
```

Clear the message line.

```
610 OUTPUT 716;"ESR?;"
```

Clear the latched User Request bit so that old key presses will not trigger a measurement.

```
620 ENTER 716;Estat
630 Stat=SPOLL(716)
```

Now wait for a key press to be reported.

```
640 IF NOT BIT(Stat,5) THEN
    GOTO 270
650 SUBEND
```

Running the program

The program assumes that the test ports being calibrated are 3.5 mm, either male or female, and that the HP 85052D 3.5 mm economy calibration kit is to be used (no sliding loads). The prompts appear just above the message line. Pressing any key on the front panel continues the program and measures the standard. The program will display DONE when the measurement calibration is complete.

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

Data transfer from analyzer to computer

Using markers to obtain trace data at specific points

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

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

```
OUTPUT 716;"MARK1  1.56 GHZ;"
```

This places marker one at 1.56 GHz. If the markers are in continuous mode, the marker value will be linearly interpolated from the two nearest points if 1.5600 GHz was not sampled. This interpolation can be prevented by putting the markers into discrete mode. The key sequence for this is **[LOCAL]**, **[MKR]**, *[MARKER MODE MENU]*, *[MARKERS:DISCRETE]*. To do it over HP-IB, execute:

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

After executing this, note that the marker is may no longer be precisely on 1.56 GHz. (This depends on the start and stop frequencies).

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

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

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

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

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

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

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

Table 1. Units as a Function of Display Format

DISPLAY FORMAT	MARKER MODE	OUTPMARK value 1, value 2	OUTPFORM value 1, value 2	MARKET READOUT** value, aux value
LOG MAG		dB,*	dB,*	dB,*
PHASE		degrees,*	degrees,	degrees,*
DELAY		seconds,*	seconds,*	seconds,*
SMITH	LIN MKR	lin mag, degrees	real, imag	lin mag, degrees
CHART				
	LOG MKR	dB, degrees	"	dB, degrees
	Re/Im	real, imag	"	real, imag
	R + jX	real, imag ohms	"	real, imag ohms
	G + jB	real, imag Siemens	"	real, imag Siemens
POLAR	LIN MKR	lin mag, degrees	real, imag	lin mag, degrees
	LOG MKR	dB, degrees	"	dB, degrees
	Re/Im	real, imag	"	real, imag
LIN MAG		lin mag,*	lin mag,*	lin mag,*
REAL		real,*	real,*	real,*
SWR		SWR,*	SWR,*	SWR,*

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

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

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

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

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

Trace transfer

Getting trace data with a 200/300 series computer can be broken down into three steps:

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

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

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

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

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

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

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

Running the program

The first number of the result is a trace value in dB, and the second is zero. Put a marker at 130 MHz, which was the first point transmitted, to see that the values displayed by the computer agree. Keep in mind that no matter how many digits are displayed, the network analyzer is specified to measure magnitude to a resolution of .001 dB, phase to a resolution of .01 degrees, and group delay to a resolution of 1 psec.

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

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

$$F = Start\_frequency + (N-1) \times Span/(Points-1)$$

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, zeros are transmitted.

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

```
40  DIMDat(1:11,1:2),Stim(1:11)
```

And type in:

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

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

Data levels

Different levels of data can be read out of the instrument. Referring to the data processing chain in Figure 2, there is available:

- Raw data. The basic measurement data, reflecting the stimulus parameters, IF averaging, and IF bandwidth. If a full 2-port measurement calibration is on, there are actually four raw arrays kept: one for each raw S-parameter. The data is read out with the commands `OUTPRAW1`, `OUTPRAW2`, `OUTPRAW3`, `OUTPRAW4`. Normally, only raw 1 is available, and it holds the current parameter. If a 2-port calibration is on, the four arrays refer to S11, S21, S12, and S22 respectively. This data is in real/imaginary pairs.

- Error Corrected data. This is the raw data with error correction applied. The array is for the currently measured parameter, and is in real/imaginary pairs. The error corrected data is read out with `OUTPDATA`. `OUTPMEMO` reads the trace memory if available, which is also error corrected data. Note that neither raw nor error corrected data reflect such post-processing functions as electrical delay offset, trace math, or time domain gating.

- Formatted data. This is the array of data actually being displayed. It reflects all post-processing functions such as electrical delay or time domain, and the units of the array read out depends on the current display format. See Table 1 for the various units as a function of display format.

- Calibration coefficients. The results of a calibration are arrays of calibration coefficients which are used in the error correction routines. Each array corresponds to a specific error term in the error model. The *Quick Reference* details which error coefficients are used for specific calibration types, and which arrays those coefficients are to be found in. Not all calibration types use all 12 arrays. The data is stored as real/imaginary pairs.

Formatted data is the most generally useful data, being the same information an operator sees on the display. However if the post processing is unneeded or unwanted, as may be the case with smoothing, error corrected data is more desirable. Error corrected data also gives you the opportunity to put the data into the instrument and apply post-processing at a later time.

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

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

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

Data formats

As stated earlier, the network analyzer can transmit data over HP-IB in five different formats. Until now, we have been using form 4, an ASCII data transfer. Another option is to use form 3, which is the IEEE 32 bit floating point format. In this mode, each number takes only 8 bytes instead of 24. This mode is particularly attractive since data is stored internally in the 200/300 series computer with the IEEE 64 bit floating point format, removing the need for any re-formatting by the computer. The fifth format is FORM5, in which the data is transferred in 32 bit floating point MS-DOS compatible format, commonly used by Personal Computers.

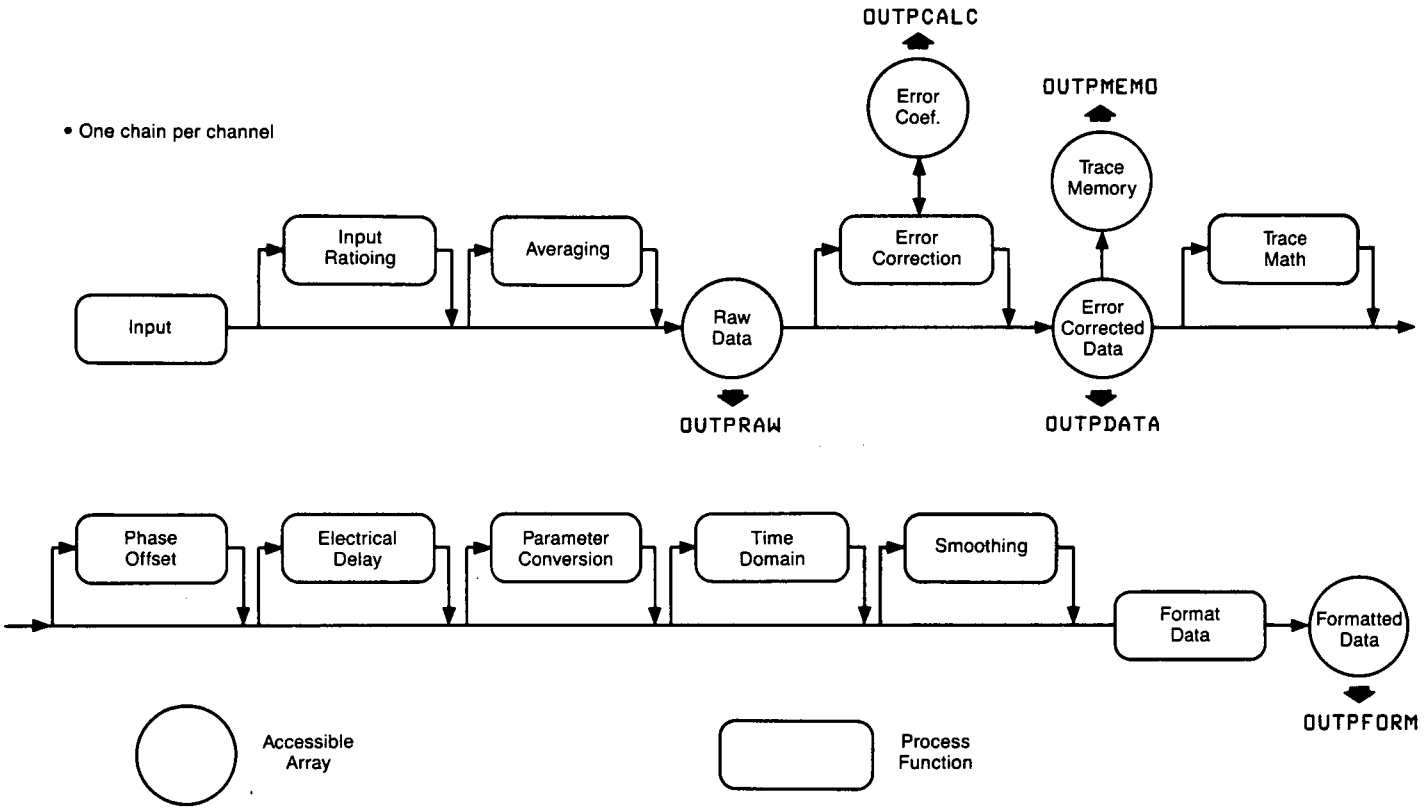


Figure 2. Data processing chain

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

Example program 3B illustrates data transfer using form 3, in which data is transmitted in the IEEE 32 bit floating point format.

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

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

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

Running the program

Preset the instrument and run the program. The computer displays 3,216 and the trace values at 130 MHz. The number 3,216 comes from 201 points, 2 values per point, 8 bytes per value. Note that this transfer is much faster than a form 4 transfer: more than twice as fast.

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

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

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

BASIC offers two options to those who want to dimension an array with a variable expression, such as the number of points in the sweep. One is the REDIM statement, available with AP2—1 or the MAT binary, which redimensions a given array to any size less than or equal to its originally dimensioned size. The other option is to ALLOCATE the array just before using it, and DEALLOCATE when it's no longer needed. ALLOCATE works exactly like DIM, except that when you deallocate, the memory space is returned to general use and you can re-use the variable name. All of the following examples use ALLOCATE.

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

```
70  ENTER @Dt;Hdr,Lgth
80  ALLOCATE
    Dat(1:Lgth/16,1:2)
```

This guarantees that the receiving array is the correct size. In form 3, each number is 8 bytes, and there are two numbers per point, so we divide Lgth by 16 to get number of points.

```
90  ENTER @Dt;Dat(*)
100 DISP Dat(Lgth/16,1)
```

Display the last number read in.

```
110 END
```

Set the number of points to 51 and run the program: this time no errors are generated. Set the number of points to 401, and run the program again. Move a marker to the last point on the trace, and check to see that the last point read in was the last point on the trace, as expected. There are two other formats available. Form 2 is not used with 200/300 computers, and form 1 is a special high speed transfer. Form 1 is a condensed transfer format that is useful if data is being transferred out of the network analyzer for direct storage and later re-transmission to the network analyzer. Example 3C gives an example of a data transfer using form 1.

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

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

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

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

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

```
180 ASSIGN @Disk TO *          Close the file.
190 DEALLOCATE Dat(*)          Release the memory for the data array.
200 PURGE "TESTDATA"           And purge the data file.
210 END
```

Running the program

Preset the N, and run the program. When the program pauses press **[LOCAL]**, change the trace and press return. When the data is re-loaded, it will be formatted and displayed as the current trace. Note that this form of data transfer is even faster than the transfer using form 3.

Advanced Programming Examples

Using list frequency mode

The network 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 mode allows you 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 to ensure device performance in an efficient manner. By only sampling specific points, measurement time is reduced, since additional time is not spent measuring device performance at frequencies which are of no concern.

The following examples illustrate the use of the frequency list mode to perform arbitrary frequency testing. Example 4A lets you construct a table of list frequency segments which is then loaded into the frequency list table. Each segment stipulates a start and stop frequency, and the number of data points to be taken over that frequency range. Example 4B lets you select a specific segment to "zoom-in" on. A single instrument can be ready to measure several different devices, each with its own frequency range, using a single calibration performed with all of the segments active. When a specific device is connected, the operator selects the appropriate segment for that device. List frequency segments can be overlapped, but the total number of points in all the segments cannot exceed 1601 points.

Example 4A: Setting up a list frequency sweep

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

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

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

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

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

```
10  ABORT 7
20  CLEAR 716                          Prepare the network analyzer for HP-IB control
30  OUTPUT 716;"EDITLIST;"             Activate the frequency list edit mode.
40  FOR I=1 to 30                      Setup a FOR NEXT loop.
50  OUTPUT 716;"SDEL;"                 Delete any existing segments.
60  NEXT I
70  INPUT "Number of                   Find out how many segments to expect
    segments?",Numb
80  ALLOCATE Table(1:Numb,1:3)         Create a table to hold the segments. Keep start
                                       frequency, stop frequency, and number of points.
90  PRINTER IS 1                       Make sure we print on the screen.
100 OUTPUT 2;CHR$(255)&"K";            Clear the screen.
110 PRINT USING "10A,10A,10A,          Print the table header.
    20A";"SEGMENT","START" ,
    "STOP","NUMBER OF
    POINTS"
120 FOR I=1 TO Numb                    Read in each segment.
130 GOSUB Loadpoin                     Loadpoin (line 300) reads in the start frequency,
                                       stop frequency, and number of points for segment I.
                                       Since Loadpoin is a subroutine, I is used as a
                                       global variable.
```

```
140 NEXT I
150 LOOP
160 INPUT "DO YOU WANT TO
    EDIT? Y OR N" ,An$
170 EXIT IF An$="N"
180 INPUT "ENTRY NUMBER?",I
190 GOSUB Loadpoin
200 END LOOP
210 OUTPUT 716;"EDITLIST"
220 FOR I=1 TO Numb
230 OUTPUT 716;"SADD;STAR";
    Table(I,1);"GHZ;"
240 OUTPUT 716;"STOP";
    Table(I,2);"GHZ;"
250 OUTPUT 716;"POIN",
    Table(I,3),";"
260 OUTPUT 716;"SDON;"
270 NEXT I
280 OUTPUT
    716;"EDITDONE;LISFREQ;"
290 STOP
300 Loadpoin: !
310 INPUT "START FREQUENCY?
    (GHZ)",Table(I,1)
320 INPUT "STOP FREQUENCY?
    (GHZ)",Table(I,2)
330 INPUT "NUMBER OF
    POINTS?",Table(I ,3)
340 IF Table(I,3)=1 THEN
    Table(I,2)=Table(I,1)
350 PRINT TABXY(0,I+1);I;
    TAB(10);Table(I,1);
    TAB(20);Table(I ,2);
    TAB(30),Table
360 RETURN
370 END
```

Use the LOOP, EXIT IF, END LOOP structure to loop and edit the table until editing is done. This sets up a loop with the exit point in the middle of the loop rather than the beginning (as with WHILE, END WHILE,) or the end (as with REPEAT, UNTIL.)

Edit the table. Editing is re-entering the entire segment. The old segment values are left in place if return is pressed without typing anything.

Exit the edit loop if editing is finished. Execution is continued at line 210.

For editing, get the entry number.

And have Loadpoin re-enter the values.

Begin the table entry by opening the list frequency table for editing. The table must be empty, or these segments will write over the old ones.

Loop for each segment.

Enter the segment values.

Declare the segment done.

Close the table, and turn on the list frequency mode.

Enter in a segment.

Enter the segment values.

If only one point in the segment, make the stop frequency equal to the start frequency to avoid ambiguity.

Print the segment out. Because of the TABXY, this will print over old segments editing is being done.

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 instrument, and list frequency mode turned on. During editing, simply pressing return leaves an entry at the old value.

Any segments already in the frequency list table will be deleted by the program. If this is not desired, delete lines 40 thru 60. New segments will be entered on top of the old ones.

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

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

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

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

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

```
230 ENTER 716;Table(I,1)
240 OUTPUT
    716;"STOP;OUTPACTI;"
250 ENTER 716;Table(I,2)
260 OUTPUT
    716;"POIN;OUTPACTI;"
270 ENTER 716;Table(I,3)
280 IF I=18 THEN INPUT "HIT
    RETURN FOR MORE",A$
290 IMAGE
    4D,6X,2D.9D,3X,2D.9D,3X,4D
300 PRINT USING 290;I;
    Table(I,1)/1.E+9;
    Table(I,2)/1.E+9;
    Table(I,3)
310 RETURN
320 END
```

Read the start frequency into the list table.

Make the stop frequency active, and output its value.

Read the stop frequency value.

Make the number of points active, and output its value.

Read the number of points.

Stop printing when 17 segments have been listed on the display, this allows the operator to examine the first 17 segments before they are scrolled off the computer display by addition segments (remember, there are up to 30 segments).

Specify the print format and margins for the list frequency table.

Print out the segment parameters for the I th segment.

Running the program

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

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

Using limit lines to perform PASS/FAIL tests

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

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

Example 5A: Setting up limit lines

The purpose of this example is to show how to create a limit table and transmit it.

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

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

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

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

```
10  ABORT 7
20  CLEAR 716                          Prepare for HP-IB control
30  FOR I =1 to 15
40  OUTPUT                             Delete any existing limits.
    716;"EDITLIML;SDEL;"
50  NEXT I
60  INPUT "Number of                   Find out how many limits to expect
    limits?",Numb
70  ALLOCATE Table(1:Numb,1:3)         Create a table to hold the limits. It will contain
                                       stimulus value (frequency), upper limit value, and
                                       the lower limit value.
80  ALLOCATE Limtype$(Numb)[2]         Create a string array to indicate the limit types.
90  PRINTER IS 1                       Make sure we print on the screen.
100 OUTPUT 2;CHR$(255)&"K";            Clear the screen.
110 PRINT USING "10A,
    20A,15A,20A";"SEG",
    "UPPER (dB)", "LOWER
    (dB)","TYPE"Print the
    table header.
120 FOR I=1 TO Numb                    Read in each segment.
130 GOSUB Loadlimit                    Loadlimit (line 330) reads in the stimulus value
                                       (frequency), upper value, lower value, and the limit
                                       type for limit I. Since Loadlimit is a subroutine, I
                                       is used as a global variable.
140 NEXT I
150 LOOP                               Use the LOOP, EXIT IF, END LOOP structure to
                                       loop and edit the table until the operator indicates
                                       that editing is no longer desired. This structure sets
                                       up a loop with the exit point in the middle of the
                                       loop rather than at the beginning (as with WHILE,
                                       END WHILE,) or at the end (as with REPEAT,
                                       UNTIL.)
160 INPUT "DO YOU WANT TO              Let the operator edit the table. Editing is actually re-
    EDIT? Y OR N",An$                  entering the entire limit. The old limit values are left
                                       in place if the operator presses return without
                                       typing anything.
170 EXIT IF An$="N"                    Exit the edit loop if editing is finished. Execution is
                                       continued at line 210.
180 INPUT "ENTRY NUMBER?",I            For editing, get the entry number.
```

```
190 GOSUB Loadlimit
```
And have `Loadlimit` re-enter the values.

```
200 END LOOP
```

```
210 OUTPUT 716;"EDITLIML;"
```
Begin the table entry by opening the limit table for editing. The limit table must be empty, or these limits will just be added on top of the old ones.

```
220 FOR I=1 TO Numb
```
Loop for each limit.

```
230 OUTPUT 716;"SADD;
    LIMS";Table(I,1);"GHZ;"
```
Enter the stimulus value.

```
240 OUTPUT 716;"LIMU";
    Table(I,2);"DB;"
```
Enter the upper limit value.

```
250 OUTPUT 716;"LIML",
    Table(I,3),"DB;"
```
Enter the lower limit value.

```
260 IF Limtype$(I)="FL" THEN
    OUTPUT 716;"LIMTFL;"
```
Set flat limit type.

```
270 IF Limtype$(I)="SL" THEN
    OUTPUT 716;"LIMTSL;"
```
Set sloped limit type.

```
280 IF Limtype$(I)="SP" THEN
    OUTPUT 716;"LIMTSP;"
```
Set point limit type.

```
290 OUTPUT 716;"SDON;"
```
Declare the limit done.

```
300 NEXT I
```

```
310 OUTPUT 716;"EDITDONE;
    LIMILINEON; LIMITESTON;"
```
Close the table, display the limits, and activate limit testing.

```
320 STOP
```

```
330 Loadlimit: !
```
Enter in a segment.

```
340 INPUT "STIMULUS VALUE?
    (GHZ)",Table(I,1)
```

```
350 INPUT "UPPER LIMIT VALUE
    (DB)?",Table(I,2)
```

```
360 INPUT "LOWER LIMIT VALUE
    (DB)?",Table(I,3)
```
Enter the limit values.

```
370 INPUT "LIMIT TYPE"
    (FL=FLAT, SL=SLOPED,
    SP=POINT)",Limtype$(I)
```
Enter the limit type.

```
380 PRINT TABXY(0,I+1);I;
    TAB(10);Table(I ,1);
    TAB(30);Table(I,2);
    TAB(45),Table(I,3),
    TAB(67); Limtype$(I)
```
Print the limit values out. Because of the `TABXY`, this will print over old limits if a limit is being edited.

```
390 RETURN
```

```
400 END
```

Running the program

The program displays the limit table as it is entered. During editing, the displayed table is updated as each line is edited. The table is not reordered. At the completion of editing, the table is entered, and limit testing mode is turned on. During editing, simply pressing return leaves an entry at the old value.

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

Example 5B: Performing PASS/FAIL tests while tuning

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

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

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

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

In this example, the requirement that several sweeps in a row must pass is used in order to give confidence that the limit test pass was not a fluke due to the device settling or the operator tuning during the sweep. Upon running the program, the number of passed sweeps for qualification is entered. For very slow sweeps, a small number such as 2 is appropriate. For very fast sweeps, where the device needs time to settle after tuning and the operator needs time to get away from the device, as many sweeps as 6 or more sweeps might be appropriate.

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

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

```
10  ABORT 7
20  CLEAR 716                          Prepare for remote control.
30  INPUT "Number of tests             Find out how many sweeps must pass before the
    for qualification?",Qual           device is considered to have passed the limit test.
40  DISP "TUNE DEVICE"                 Tell operator to begin tuning.
50  Reap=0                             Reap is a counter holding how many sweeps have
                                       passed the limit test.
60  OUTPUT 716;"OPC?;SING;"            Take a sweep. When it is done, limit test will have
                                       occurred.
70  ENTER 716;Reply                    Wait for the end of the sweep.
80  OUTPUT 716;"ESB?;"                 Check to see if the fail bit is set.
90  ENTER 716;Estat
100 IF BIT(Estat,4) THEN               If the fail bit for channel one is set, reset the number
                                       of sweeps passed counter.
110 IF Reap<>0 THEN BEEP               If sweeps had been passing, warn the operator that
    1200,.05                           the device is now failing.
120 Reap=0
130 GOTO 40
140 END IF                             If the fail bit was not set, tell the operator.
150 BEEP 2500,.01
160 Reap=Reap+1                        Increment the sweeps passed counter.
170 DISP "STOP TUNING"                 Encourage the operator to stop tuning the device.
180 IF Reap<Qual THEN GOTO 60          If not enough sweeps have passed, loop.
190 DISP "DEVICE PASSED!"              The device has passed.
```

```
200 FOR I=1 TO 10
210 BEEP 1000,.05
220 BEEP 2000,.01
230 NEXT I
240 INPUT "HIT RETURN FOR
    NEXT DEVICE",Dum$
250 GOTO 40
260 END
```

Warble, telling the operator the device has passed, using an audible signal.

Wait for the next device.

Running the program

Set up a limit table on channel 1 for a specific device either manually, or using the program in Example 5A. The recommended device is the bandpass filter supplied with the instrument (HP Part No. 0955-0446). Run the program, and enter the number of passed sweeps desired for qualification. After entering the qualification number, connect the filter. When a sweep passes, the computer beeps. When enough sweeps in a row pass to qualify the device, the computer beeps at the operator, and then asks for a new device. For the bandpass filter, the suggested limits are as follows:

Seg	Stimulus (GHz)	Upper (dB)	Lower (dB)	Type
1	8.0	−70	−200	FL
2	9.0	−70	−200	SP
3	9.4	−60	−200	SL
4	10.0	−3	−200	SP
5	10.2	0	−3	FL
6	10.3	0	−3	SP
7	10.5	−3	−200	SL
8	11.1	−60	−200	SP
9	11.5	−70	−200	FL
10	12.5	−70	−200	SP

These are only suggestions. Your filter may vary slightly, and the limits may need to be modified to allow the filter to pass. The program assumes a response calibration (thru calibration) or full 2-port calibration has been performed prior to running the program. Try causing the filter to fail by loosening the cables connecting the filter, and then retightening them.

Storing and recalling instrument states

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

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.

Example 6A: 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. The learn string is read out with `OUTPLEAS`, and put back into the instrument with `INPULEAS`. The string itself is in form 1, and is no longer than 3000 bytes long.

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

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

Running the program

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

Using the learn string obtained from an HP 8702A

Assuming that State$ has an HP 8702A learn string, it can be input to an HP 8720B/8719A in the following way:

```
10  DIM State$ "3000].
20                                      Obtain the HP 8720A learn string in State$.
30  Eol$ = ``;''
40  ASSIGN @ ANA TO 716; EOL            Syncs EOI with a ; as last character
    Eol$ END
50  OUTPUT @ ANA; ``INPULEAS'';
    State$
60  End
```

Example 6B: Coordinating disk storage

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

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

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

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

Running the program

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

Example 6C: Reading calibration data

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

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

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

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

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

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

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

```
170 FOR I=1 TO Numb
180 OUTPUT 716 USING "K,ZZ";
    "OUTPCALC",I
190 ENTER @Dt;Hdr,Lgth
200 FOR J=1 TO Poin
210 ENTER @Dt;
    Cal(I,J,1),Cal(I,J,2)
220 NEXT J
230 NEXT I
240 INPUT "HIT RETURN TO RE-
    TRANSMIT
    CALIBRATION",Dum$
250 OUTPUT 716;Calt$,";"
260 FOR I=1 TO Numb
270 DISP "TRANSMITTING
    ARRAY: ",I
280 OUTPUT 716 USING "K,ZZ";
    "FORM3;INPUCALC",I
290 OUTPUT @Dt;Hdr,Lgth
300 FOR J=1 TO Poin
310 OUTPUT @Dt;Cal(I,J,1),
    Cal(I,J,2)
320 NEXT J
330 NEXT I
340 OUTPUT 716;"SAVC;"
350 OUTPUT 716;"CONT;"
360 DISP "DONE"
370 END
```

Loop once for each calibration coefficient.

Request the calibration coefficient. The K transmits OUTPCALC literally, and ZZ transmits I as two digits, using a leading zero if needed.

Read the header.

Since we are not filling the entire array, we have to read each point individually.

The calibration data is now all in the computer.

Begin the calibration re-transmission by declaring what calibration type is about to be loaded.

Now load each calibration coefficient.

All of the calibration data has been loaded.

End the sequence by activating the calibration.

Trigger a sweep so the calibration becomes active.

Running the program

Before executing the program, perform a calibration.

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

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

Miscellaneous Programming Examples

Controlling peripherals

The purpose of this section is to demonstrate how to coordinate printers, plotters, and disk drives.

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

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

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 network analyzer to the plotter or printer. The computer sets up the path by addressing the network analyzer to talk and the plotter to listen and then placing the bus into data mode. The network analyzer will then make the plot or print. When it is finished, it asserts the End or Identify (EOI) control line on HP-IB.

This program makes a plot using the talker/listener mode. It is stored on the Example Programs disk as **IPG7A**.

```
10  OUTPUT 716;"OUTPPLOT;"
```
Command the plot using the talker/listener mode plot command. For a printer, use `OUTPPRIN;`.

```
20  SEND 7;UNL LISTEN 5 TALK 16
    DATA
```
Use the HP-IB control commands to establish a data path to the plotter. `SEND 7` sends bus control commands. `UNL` clears out the last data path. `LISTEN 5` tells the device at address 5, the default address for a plotter, to accept the data. For printing, substitute the address 1, the default for a printer, and change `"OUTPPLOT;"` in line 10 to `"OUTPPRIN;"`. `TALK 16` tells the HP 8720A to talk; that is, transmit the contents of its output queue. When `DATA` is executed, the bus changes from command to data mode, and the plot is made.

```
30  DISP "PLOTTING"
```
This statement serves the dual purpose of informing the user of the state of the program and preventing interrogation of status register 7 immediately after the `SEND` statement, when the register state is unstable.

```
40  STATUS 7,7;Stat
```
Now wait for the instrument to assert the EOI line, indicating the end of transmission. The `STATUS` command accesses the status registers for the interfaces installed on the computer. In this case, we access interface 7 (HP-IB), register 7, HP-IB status. The value of the register is placed in the variable `Stat`. We are specifically interested in bit 11, which is assigned to the EOI line.

```
50  IF NOT BIT(Stat,11) THEN
    GOTO 40
```
If bit 11 is not set, then the EOI line is not being asserted, so loop and check again.

```
60  DISP "DONE"
```

The network analyzer has asserted EOI to indicate that it has finished with the plot.

```
70  END
```

Running the program

The network analyzer will go into remote, and make the plot. During the plot, the computer will display the message `PLOTTING.` 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 command `PSOFTON` and disabled with `PSOFTOFF`.

When the plot is completed, the network analyzer asserts the EOI line on HP-IB. The computer detects this and displays the DONE message. The network analyzer will go on asserting EOI until some other activity on the bus causes it to clear the line.

If a problem arises with the plotter, such as no pen or paper, the network analyzer cannot detect the situation because it only has a one-way path of communication. Hence, the instrument will attempt to continue plotting until the operator intervenes and aborts the plot by pressing a front panel key. Pressing a front panel key aborts the plot, causes the warning message "CAUTION: PLOT ABORTED", asserts EOI, and hence frees the computer. Because of possible malfunctions, it is generally advisable to use pass control mode, which allows two way communication between the plotter and the network analyzer.

Example 7B: Operation using pass control mode

If the network analyzer is in pass control mode and receives a command telling it to plot, print, or store/load to disk, it sets bit 1 in the event status register to indicate that it needs control of the bus. If the computer then uses the HP-IB control command to pass control, the network analyzer will take control of the bus, and access the peripheral. When the network analyzer no longer requires control, it will pass control back to the computer.

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

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

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

10	`OUTPUT 716;"CLES;ESE2;"`	Clear the status reporting system, and enable the Request Active Control bit in the event status register.
20	`OUTPUT 716;"USEPASC;PRINALL;"`	Put the network analyzer in pass control mode, and request a print.
30	`Stat=SPOLL(716)`	Get the status byte.
40	`IF NOT BIT(Stat,5) THEN GOTO 30`	If the network analyzer is not requesting control, loop and wait.
50	`SEND 7;TALK 16 CMD 9`	This is the bus command to pass active control to device 16. With BASIC 3.0 or higher, or 2.0 with extensions 2.1, the command `PASS CONTROL 716` can be used instead.
60	`DISP "PRINTING"`	
70	`STATUS 7,6;Hpib`	To determine when the print is finished, watch for return of active control. The `STATUS` command loads the interface 7 (HP-IB) register 6, the computer's status with respect to HP-IB, into the variable `Hpib`. Bit 6 tells if the computer is the active controller: it will be set when control is returned.
80	`IF NOT BIT(Hpib,6) THEN GOTO 70`	If control has not returned, loop and wait.
90	`DISP "DONE"`	Control has returned.
100	`END`	

Running the program

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

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

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

Example 8: Creating a user interface

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

User graphics

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

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

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

Front panel control

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

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

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

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

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

`10`	`INTEGER Hdr,Lgth,Keyc`	Declare variables to hold the header and the key code.
`20`	`ASSIGN @Dt TO 716;FORMAT`	Define an IO path with formatting off, to receive the form 3 trace data for plotting.
`30`	`OUTPUT 716;"HOLD;AUTO;CLES;ESE 64;POIN?;"`	Prepare the instrument. `HOLD;AUTO;` freezes and scales the trace for plotting. `CLES;ESE 64;` clears the status reporting system and enables the User Request bit in the event status register. Lastly, `POIN?;` requests the number of points.
`40`	`ENTER 716;Poin`	Read in the number of points.
`50`	`GINIT`	Initialize the graphics functions in the computer.
`60`	`PLOTTER IS 717,"HPGL"`	Specify the network analyzer display as the plotting device.
`70`	`OUTPUT 717;"CS;"`	Turn off the measurement display.

```
80  Cx=55
```
Initialize the x position of the center of the rectangle.

```
90  Cy=60
```
Initialize the y position of the center of the rectangle.

```
100 S=20
```
Set the size of the rectangle.

```
110 REPEAT
```
The REPEAT, UNTIL structure sets up a loop that keeps repeating until the condition specified in the UNTIL statement is found to be true. The condition is checked at the end of the loop. In this case, loop and redraw the rectangle until **[ENTRY OFF]** has been pressed.

```
120 LINE TYPE 4
```
Select a dashed line for the rectangle.

```
130 GCLEAR
```
Clear the graphics area.

```
140 IF Cx>160 THEN Cx=160
```
Prevent box from going off the screen.

```
150 IF Cx<-17 THEN Cx=-17
```
Note that these values are linked to the increments set in lines 270/310 and 320!

```
160 IF Cy>115 THEN Cy=115
170 IF Cy<-15 THEN Cy=-15
```

```
180 VIEWPORT Cx-S,Cx+S,Cy-
    S,Cy+S
```
Define the area of the rectangle, which will become the plotting area for the grid and trace.

```
190 WINDOW 0,Poin-1,0,1
```
Define the units along the edges of the rectangle. In this case, the horizontal edge has as many units as points in the sweep, and the vertical edge is simply unity.

```
200 FRAME
```
Draw the rectangle around the plotting area.

```
210 Stat=SPOLL(716)
```
Read the status byte.

```
220 IF NOT BIT(Stat,5) THEN
    GOTO 170
```
If bit 5 is not set, a key has not been pressed, so loop and wait.

```
230 OUTPUT 716;"ESR?;"
```
A key press has occurred, so read the event status register in order to clear the latched bit.

```
240 ENTER 716;Estat
```
Read in the register value, but do nothing with it.

```
250 OUTPUT 716;"KOR?;"
```
Now read in the key or knob count.

```
260 ENTER 716;Keyc
```

```
270 IF Keyc=26 THEN Cy=Cy+5
```
Key 26 is the up key, so shift the rectangle up.

```
280 IF Keyc=18 THEN Cy=Cy-5
```
Key 18 is the down key, so shift the rectangle down.

```
290 IF Keyc<0 THEN
```
If the keycode was negative, then it is a knob count.

```
300 Knb=Keyc
```
Decode the knob count into the variable Knb.

```
310 IF Knb<-64 THEN
    Knb=Knb+32768
```
If the count is less then -64, add 32768 (2^{15}) to recover the knob count. If the count is more than -64, then no decoding is needed.

```
320 Cx=Cx-Knb*3
```
Shift the rectangle according the knob count, multiplying the knob count to make the rectangle move farther.

```
330 END IF
```

```
340 UNTIL Keyc=34
```
This is the end of the REPEAT, UNTIL structure. Leave the loop only when key 34, **[ENTRY OFF]** has been pressed.

```
350 GRID (Poin-1)/10,.1
```
[ENTRY OFF] has been pressed, so draw the grid and the trace. This statement draws a grid with 10 divisions on each axis.

```
360 LINE TYPE 1
```
Use a solid line for the trace.

```
370 OUTPUT
    716;"FORM3;OUTPFORM;"
380 ENTER @Dt;Hdr,Lgth
390 ALLOCATE Dat(1:Poin,1:2)
400 ENTER @Dt;Dat(*)
410 OUTPUT 716;"SCAL?;"
420 ENTER 716;Scal
430 OUTPUT 716;"REFV?;"
440 ENTER 716;Ref
450 OUTPUT 716;"REFD?;"
460 ENTER 716;Refp
470 Bot=Rev-Refp*Scal
480 Full=10*Scal
490 MOVE 0,(Dat(1,1)-Bot)/Full
500 FOR I=1 TO Poin-1
510 DRAW I,(Dat(I,1)-Bot)/Full
520 NEXT I
530 END
```

Now get the trace data.

Get the header information.

Define the receiving array.

And read in the data.

Instead of scaling the data in this program, interrogate the scale factor.

Similarly, use the value at the reference position to decide where to draw the trace.

Interrogate the current reference position being used.

Calculate the value of the bottom grid line.

And define the full scale span across the grid.

Go to the first point on the trace without drawing anything.

And draw all the rest of the points in the trace.

The trace is drawn, so end the program.

Running the program

Before running the program, set the instrument up to make a measurement. Because of memory limitations, the network analyzer will not accept a graphics dump of a trace of greater than 201 points.

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

Appendix A: Status Reporting

The network analyzer has a status reporting mechanism that gives information about specific functions and events inside the instrument. The status byte is an 8 bit register with each bit summarizing the state of one aspect of the instrument. For example, the error queue summary bit will always be set if there are any errors in the queue. The value of the status byte can be read with the `SPOLL(716)` statement. This command does not automatically put the instrument in remote mode, thus giving the operator access to the 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 that is set when the instrument is issuing a service request over HP-IB, and a bit that is set when network analyzer has data to send out over HP-IB. See Figure A.1 for a definition of the status registers.

Example A1: Using the error queue

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

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

10	`DIM Err$[50]`	Prepare a string to hold the error message.
20	`Stat=SPOLL(716)`	Use the serial poll statement to read the status byte into the variable `Stat`. Serial poll is an HP-IB function dedicated specifically to getting the status byte of an instrument quickly, and does not cause the network analyzer to go into remote.
30	`IF NOT BIT(Stat,3) THEN GOTO 20`	If the error queue summary bit is not set, we loop until it gets set.
40	`OUTPUT 716;"OUTPERRO;"`	If the error queue has something in it, we instruct the network analyzer to output the error number and the error message. This communication will put it in remote mode.
50	`ENTER 716;Err,Err$`	`Err` holds the error number, `Err$` the error message.
60	`PRINT Err,Err$`	
70	`LOCAL 716`	Return to local mode so that the front panel is available to the operator.
80	`BEEP 600,.01`	Give an audible signal that there is a problem.
90	`GOTO 20`	
100	`END`	

Running the program

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

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

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

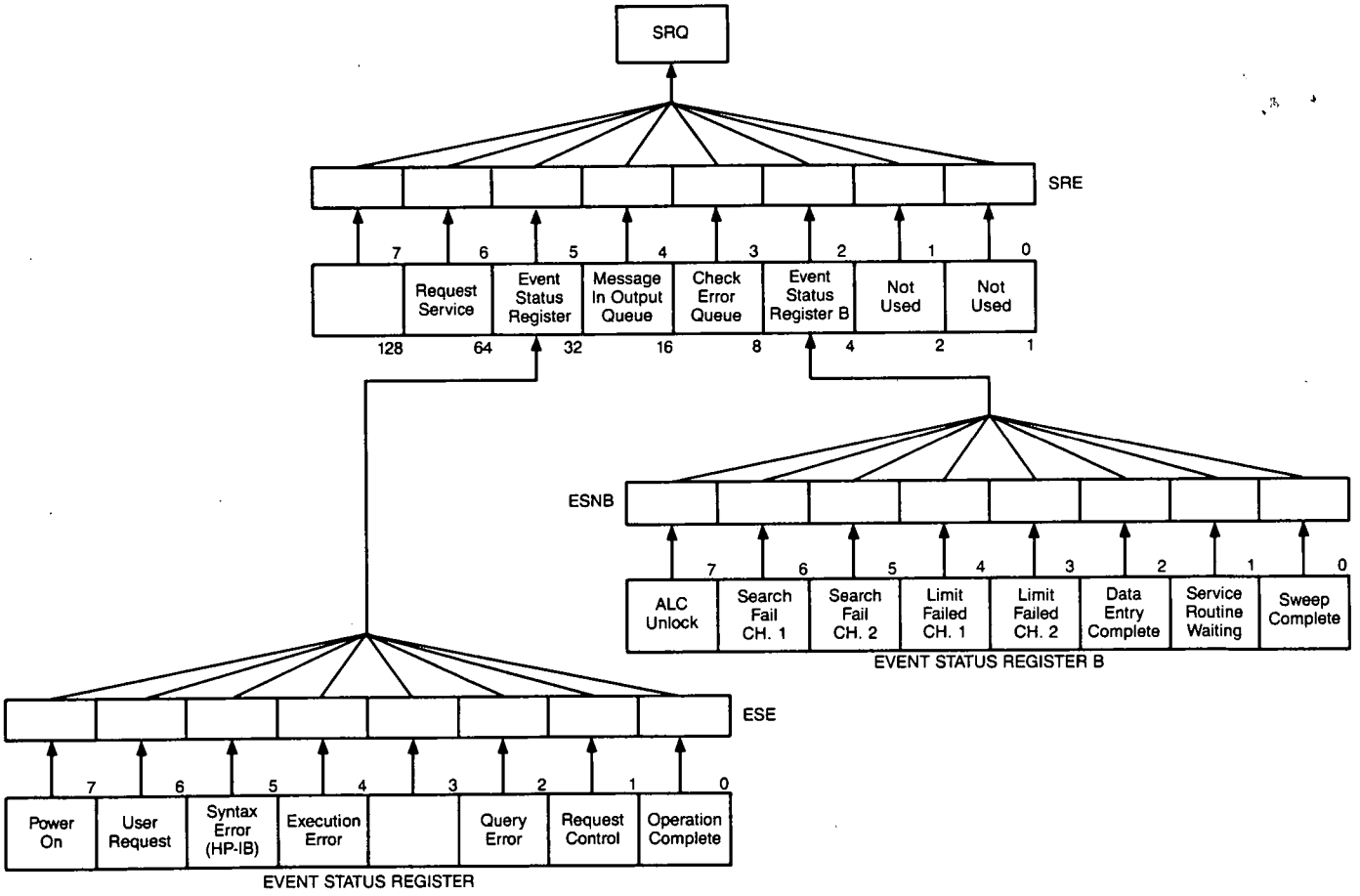


Figure A.1. Status reporting system

Example A2: Using the status registers

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

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

```
10  CLEAR 716
20  OUTPUT 716;"ESR?;"
30  ENTER 716;Estat
40  IF NOT BIT(Estat,6) THEN
    GOTO 20
50  OUTPUT 716;"KOR?;"
60  ENTER 716;Keyc
70  IF Keyc≥0 then
    PRINT "KEY ";
80  IF Keyc<-400 THEN
    Keyc=Keyc+32768
90  PRINT "CODE =",Keyc
100 GOTO 20
110 END
```

Clear out any old conditions.

Read out the event status register.

If the user request bit of the event status register is not set, loop back.

If the user request bit has been set, there has been some front panel activity, and we read out the key code. The reply to KOR?; includes the knob count if the knob was turned. The information comes as a negative number, and has to be decoded.

If the code was positive, we know it was a key press rather than a knob turn, and print the leader KEY. By placing a semicolon after the statement, we suppress the carriage return, line feed, allowing the code to be printed on the same line.

If the keycode is negative, it represents a knob count. If it isn't less than −400, then the count is a clockwise rotation and needs no modification. However, if the count is less than −400, we have to add 32,768 (2'15) to get the counter-clockwise count.

Print the decoded key code.

Wait for the next key press.

Running the program

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

Example A3: Generating interrupts

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

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

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

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

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

10	`OUTPUT 716;"CLES; ESE 32; SRE 32;"`	Clear the status reporting system, and then enable bit 5 of the event status register, and bit 5 of the status byte so that an SRQ will be generated on a syntax error.
20	`ON INTR 7 GOTO Err`	Tell the computer where to branch it gets the interrupt.
30	`ENABLE INTR 7;2`	Tell the 200/300 series to enable an interrupt from interface 7 (HP-IB) when bit 1 (value 2, the SRQ bit) of the interrupt register is set. If there is more than one instrument on the bus capable of generating an SRQ, it is necessary to use serial poll to determine which device has issued the SRQ. In this case, we assume the HP 8720A did it. A branch to `Err` will disable the interrupt, so the return from `Err` re-enables it.
40	`GOTO 40`	Do nothing loop.
50	`Err:!`	
70	`OUTPUT 716;"ESR?"`	The interrupt has come in! Read the register to clear the bit.
80	`ENTER 716;Estat`	
90	`PRINT "SYNTAX ERROR DETECTED"`	
100	`ENABLE INTR 7`	
110	`GOTO 30`	
120	`END`	

Running the program

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

```
OUTPUT 716;"STIP 2 GHZ;"
```

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

The SRQ can be cleared by reading the event status register and hence clearing the latched bit, or by clearing the enable registers with `CLES`. The syntax error message on the

display can only be cleared by `CLEAR 7` or `CLEAR 716`. `CLEAR 7` is not commonly used because it clears every device on the bus.

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

```
CLEAR 716
OUTPUT 716;"STOP 100 GHZ;"
```

The stop frequency is set to 20 GHz (or 13.5 GHz), without generating a syntax error.

May 1, 1989

08719-90030

HEWLETT
PACKARD