Programmer's Guide

HP 8702D
Lightwave Component Analyzer

HEWLETT®
PACKARD

HP Part No. 08702-90042
Printed in USA
April 1997

Hewlett-Packard Company
Lightwave Operations
1400 Fountaingrove Parkway
Santa Rosa, CA 95403-1799,
USA
(707) 577-1400

Safety Symbols.
CAUTION

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

WARNING

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

⚠ The instruction manual symbol. The product is marked with this warning symbol when it is necessary for the user to refer to the instructions in the manual.

The laser radiation symbol. This warning symbol is marked on products which have a laser output.

∿ The AC symbol is used to indicate the required nature of the line module input power.

| The ON symbols are used to mark the positions of the instrument power line switch.

❍ The OFF symbols are used to mark the positions of the instrument power line switch.

CE The CE mark is a registered trademark of the European Community.

The CSA mark is a registered trademark of the Canadian Standards Association.

ISM1-A This text denotes the instrument is an Industrial Scientific and Medical Group 1 Class A product.

Typographical Conventions.
The following conventions are used in this book:

Key type for keys or text located on the keyboard or instrument.

Softkey type for key names that are displayed on the instrument's screen.

`Display type` for words or characters displayed on the computer's screen or instrument's display.

User type for words or characters that you type or enter.

Emphasis type for words or characters that emphasize some point or that are used as place holders for text that you type.

The HP 8702D—At a Glance

The *HP 8702D Programmer's Guide* shows you how to program the instrument. The programming syntax conforms to the *IEEE 488.2 Standard Digital Interface for Programmable Instrumentation*.

Chapter 1, "Writing Programs" provides information on addressing the instrument, debugging programs, and other important tasks. Each programming command is documented in Chapter 3, "Language Reference". Chapter 4, "Graphics Language Reference" documents graphics commands for drawing objects on the display. Refer to Chapter 2, "Examples" for a list of practical examples using common measurement tasks.

1

Writing Programs

Writing Programs

This chapter provides a general introduction to programming the HP 8702D. It covers topics such as command syntax, addressing the instrument, initializing the instrument, and returning measurement results to the computer.

Before you begin programming, you should first become proficient at making manual measurements as explained in the *HP 8702D User's Guide*. Once you learn the techniques and softkey presses needed to make a measurement, you can refer to Table 5-4, "Keys versus Programming Commands," on page 5-26 to locate the equivalent programming command.

NOTE:

The HP 8702D conforms to IEEE 488.1 and IEC-625 standards for interfacing instruments.

What you'll find in this chapter

General Information

The HP 8702D occupies two HP-IB addresses: the instrument itself and the display. The display address is derived from the instrument address as described in “Drawing Graphics on the Display” on page 1-33. These addresses are stored in short-term, non-volatile memory and are not affected when you press PRESET or cycle the power. The default address for the HP 8702D is device 16, and the display address is device 17.

There is also an address for the system controller. This address refers to the controller when the HP 8702D is being used in pass-control mode. This is the address that control is passed back to when the HP 8702D-controlled operation is complete.

You can change the HP-IB address from the front panel as described in “To change the HP-IB address” on page 1-8.

HP-IB status lights

When the HP 8702D is connected to the HP-IB, the front-panel HP-IB status lights indicate the current status of the HP 8702D. These lights have the following definitions:

R = Remote operation

L = Listen mode

T = Talk mode

S = Service request (SRQ) asserted by the HP 8702D

Remote mode and front-panel lockout

Whenever the instrument is controlled by a computer, the front-panel HP-IB remote status light, R, is turned on, and the instrument keys are disabled. Press the LOCAL key to restore front panel control of the instrument.

Consult the documentation for your programming environment to determine which commands are used to put an instrument in the remote and local lockout modes. These are not HP 8702D commands; they control HP-IB control lines and do not send any characters to the HP 8702D.

Initialize the instrument at the start of every program

It is good practice to initialize the instrument at the start of every program. This ensures that the bus and all appropriate interfaces are in a known state. HP BASIC provides a CLEAR command which clears the interface buffer and also resets the instrument's parser. (The parser is the program that reads the instructions that you send.) Whenever the instrument is under remote programming control, it should be in the single measurement acquisition mode. This is automatically accomplished when the RST is used. The RST command initializes the instrument to the factory preset state:

```
CLEAR 716
OUTPUT 716;"RST"
```

Notice in the example above, that the commands are sent to an instrument address of 716. This indicates address 16 on an interface with select code 7. Pressing the PRESET key does not change the HP-IB address. Pressing the PRESET key resets the instrument to either the factory preset state or a user-defined preset state.

Rules for writing commands

The HP 8702D accepts letters, changing lowercase to uppercase, numbers, decimal points, +, –, semicolons, carriage returns and line feeds. Leading zeros, spaces, carriage returns, and unnecessary terminators are ignored, except when inserted between a command and a numerical or string argument. If the analyzer does not recognize a character as appropriate, it generates a syntax error message and recovers at the next terminator.

For example, the CHAN command must be entered as CHAN1 or CHAN2. Inserting a space in the syntax (CHAN 1) results in an error. However, a space must be entered between the MARK2 command and its argument. For example, MARK2 1.4GHZ.

Units and terminators

The HP 8702D outputs data in basic units and assumes these basic units when it receives an input (unless the input is otherwise qualified). The basic units and allowable expressions can be found below. Both uppercase and lowercase letters are acceptable.

Table 1-1. Terminator Codes

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

Terminators are used to indicate the end of a command. This allows the HP 8702D to recover to the next command in the event of a syntax error. The semicolon (;) is the recommended command terminator. The line-feed character (LF) and the HP-IB EOI line can also be used as terminators. Again, the HP 8702D will ignore the carriage-return character (CR).

Table 1-2. HP-IB Interface Capabilities

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

To select the communication mode

1. Press the LOCAL key.
2. Press *TALKER/LISTENER* so that this softkey is underlined.

To change the HP-IB address

1. Press the LOCAL key.
2. Press *SET ADDRESSES.*
3. Press *ADDRESS: 8702.*
4. Use the front-panel knob or keys to enter the new HP-IB address.

Selecting the HP-IB Device Mode

Three different device modes are possible for the HP 8702D:

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

Performing an instrument preset does not affect the selected bus mode, although the bus mode will return to talker/listener mode if the line power is cycled.

Talker/listener mode

This is the mode that is normally used for remote programming of the HP 8702D. In talker/listener mode, the HP 8702D and all peripheral devices are controlled from an external instrument controller. The controller can command the HP 8702D to talk and other devices to listen. The HP 8702D and peripheral devices cannot talk directly to each other unless the computer sets up a data path between them. This mode allows the HP 8702D to act as either a talker or a listener, as required by the controlling computer for the particular operation in progress.

While in this mode, the HP 8702D can make a plot or print using the `OUTPPLOT;` or `OUTPPRIN;` commands. The HP 8702D will wait until it is addressed to talk by the system controller and then dump the display to a plotter/printer that the system controller has addressed to listen. Use of the commands `PLOT;` and `PRINALL;` require control to be passed to another controller.

System-controller mode

Do not attempt to use this mode for programming. This mode allows the HP 8702D to control peripherals directly in a stand-alone environment (without an external controller). This mode can only be selected manually from the HP 8702D's front panel. It can only be used if no active computer or instrument controller is connected to the system via HP-IB. If an attempt is made to set the HP 8702D to the system-controller mode when another controller is connected to the interface, the following message is displayed on the HP 8702D's display screen: `"CAUTION: ANOTHER SYSTEM CONTROLLER ON HP-IB"`.

The HP 8702D must be set to the system-controller mode in order to access peripherals from the front panel. In this mode, the HP 8702D can directly control peripherals (for example, plotters, printers, disk drives, and power meters) and the HP 8702D may plot, print, store on disk or perform power meter functions.

Pass-control mode

This mode allows the computer to control the HP 8702D via HP-IB (as with the talker/listener mode), but also allows the HP 8702D to take control of the interface in order to plot, print, or access a disk. During an HP 8702D-controlled peripheral operation, the host computer is free to perform other internal tasks (for example, data or display manipulation) while the HP 8702D is controlling the bus. After the HP 8702D-controlled task is completed, the HP 8702D returns control to the system controller.

In pass-control mode, the HP 8702D can request control from the system controller and take control of the bus if the controller addresses it to take control. This allows the HP 8702D to take control of printers, plotters, and disk drives on an as-needed basis. The HP 8702D sets event-status-register bit 1 when it needs control of the interface, and the HP 8702D will transfer control back to the system controller at the completion of the operation. It will pass control back to its controller address, specified by ADDRCONT.

To change the HP-IB device mode

1. Press the LOCAL key.
2. Press *SET ADDRESSES.*

Making Measurements

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

1. Setting up the instrument
2. Calibrating the test setup
3. Connecting the device under test
4. Taking the measurement data
5. Post-processing the measurement data
6. Transferring the measurement data

Step 1. Setting up the instrument

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

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

You can quickly set up an entire instrument state, using the save/recall registers and the learn string. The learn string is a summary of the instrument state compacted into a string that the computer reads and retransmits to the analyzer. Refer to “Using the Learn String” on page 2-45.

Step 2. Calibrating the test setup

After you have defined an instrument state, you should perform a measurement calibration. Although it is not required for the measurement of O/E and E/O devices, a measurement calibration is required to obtain useful measurements.

The following list describes several methods to calibrate the analyzer:

- Stop the program and perform a calibration from the analyzer's front panel.
- Use the computer to guide you through the calibration, as discussed in "S11 1-Port Measurement Calibration" on page 2-8 and "Full 2-Port Measurement Calibration" on page 2-13.
- Transfer the calibration data from a previous calibration back into the analyzer, as discussed in "Using Instrument States" on page 2-52.

Step 3. Connecting the device under test

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

Step 4. Taking the measurement data

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

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

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

Step 5. Post-processing the measurement data

Figure 1-1 on page 1-22 shows the process functions used to affect the data after you have made an error-corrected measurement. These process functions have parameters that can be adjusted to manipulate the error-corrected data prior to formatting. They do not affect the analyzer's data gathering. The most useful functions are trace statistics, marker searches, electrical-delay offset, time domain, and gating.

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

Step 6. Transferring the measurement data

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

Reading Data

Output queue

Whenever a output-data command is received, the HP 8702D puts the data into the output queue (or buffer) where it is held until the system controller outputs the next read command. The queue, however, is only one event long; the next output-data command will overwrite the data already in the queue. Therefore, it is important to read the output queue immediately after every interrogation or data request from the HP 8702D.

Command interrogate

All instrument functions can be interrogated to find the current ON/OFF state or value. For instrument state commands, append the question mark character (`?`) to the command to interrogate the state of the functions. Suppose the operator has changed the power level from the HP 8702D's front panel. The computer can ascertain the new power level using the HP 8702D's command-interrogate function. If a question mark is appended to the root of a command, the HP 8702D will output the value of that function. For instance, `POWE 7 DB;` sets the source power to 7 dB, and `POWE?;` outputs the current RF source power at the test port. When the HP 8702D receives `POWE?;`, it prepares to transmit the current RF source power level. This condition illuminates the HP 8702D front-panel talk light (T). In this case, the HP 8702D transmits the output power level to the controller.

ON/OFF commands can be also be interrogated. The reply is a one (1) if the function is ON or a zero (0) if it is OFF. For example, if a command controls an active function that is underlined on the HP 8702D display, interrogating that command yields a one (1) if the command is underlined or a zero (0) if it is not. As another example, there are nine options on the format menu and only one option is underlined at a time. Only the underlined option will return a one (1) when interrogated. For instance, send the command string `DUAC?;` to the HP 8702D. If dual-channel display is switched ON, the HP 8702D will return a one (1) to the instrument controller.

Similarly, to determine if phase is being measured and displayed, send the command string `PHAS?;` to the HP 8702D. In this case, the HP 8702D will return a one (1) if phase is currently being displayed. Since the command only applies to the active channel, the response to the `PHAS?;` query depends on which channel is active.

Output syntax

The following three types of data are transmitted by the HP 8702D in ASCII format:

- response to interrogation
- certain output commands
- ASCII floating-point (form 4) array transfers

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

When multiple numbers are sent, the numbers are separated by commas.

Table 1-3. Form 4 (ASCII) Data-Transfer Character Definitions

Character(s)*	Definition
Sign	– for negative, blank for positive.
3 digits	Digits to the left of the decimal point.
Decimal point	
15 digits	Digits to the right of the decimal point.
E	Exponent notation.
Sign	– for negative, + (or blank) for positive.
Exponent	Two digits for the exponent.

* The items in this column are separated by commas and delimited (terminated) with a line feed character (LF).

Marker data

The HP 8702D offers several options for outputting trace-related data. Trace information can be read out of the HP 8702D in several different formats. Data can be selectively read from the trace using the markers, or the entire trace can be read by the controller. If only specific information is required (such as a single point on the trace or the result of a marker search), the marker output command can be used to read the information.

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

As an alternative, the HP 8702D can be programmed to choose the stimulus value by using the MARKER SEARCH function. Maximum, minimum, target value, or bandwidths search can be automatically determined with MARKER SEARCH. To continually update the search, switch the marker tracking ON. The trace-maximum search will remain activated until:

- The search is switched OFF
- The tracking is switched OFF
- All markers are switched OFF

Marker data can be output to a controller with a command to the HP 8702D. This set of commands causes the HP 8702D to transmit three numbers: marker value 1, marker value 2, and marker stimulus value. In log-magnitude display mode we get the log magnitude at marker 1, zero, and the marker frequency. Refer to “Units as a Function of Display Format” on page 1-17 for a complete listing of all the possibilities for values 1 and 2. The three possibilities for the third parameter are:

- frequency
- time (as in time domain)
- CW time

Table 1-4. Units as a Function of Display Format

Display Format	Marker Mode	OUTPMARK		OUTPFORM		Marker Readout	
		Value 1	Value 2	Value 1	Value 2	Value	AUX Value
LOG MAG		dB	†	dB	†	dB	†
PHASE		degrees	†	degrees	†	degrees	†
DELAY		seconds	†	seconds	†	seconds	†
SMITH CHART	LIN MKR	lin mag	degrees	real	imag	lin mag	degrees
	LOG MKR	dB	degrees	real	imag	dB	degrees
	Re/Im	real	imag	real	imag	real	imag
	R + jX	real ohms	imag ohms	real	imag	real ohms	imag ohms
	G + jB	real Siemens	imag Siemens	real	imag	real Siemens	imag Siemens
POLAR	LIN MKR	lin mag	degrees	real	imag	lin mag	degrees
	LOG MKR	dB	degrees	real	imag	dB	degrees
	Re/Im	real	imag	real	imag	real	imag
LIN MAG		lin mag	†	lin mag	†	lin mag	†
REAL		real	†	real	†	real	†
SWR		SWR	†	SWR	†	SWR	†

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

† Value 2 is not significant in this format, though it is included in data transfers.

Array-data formats

The HP 8702D can transmit and receive arrays in the HP 8702D's internal binary format, as well as four different numeric formats. The current format is set with the FORM command. This command does not affect learn-string transfers, calibration-kit string transfers, or non-array transfers, such as command interrogate, or output marker values. A transmitted array will be output in the current format, and the HP 8702D will attempt to read incoming arrays according to the current format. Each data point in an array is a pair of numbers, usually a real/imaginary pair. The number of data points in each array is the same as the number of points in the current sweep.

The five formats are described below:

1. The HP 8702D's internal binary format, 6 bytes-per-data point. The array is preceded by a four-byte header. The first two bytes represent the string `"#A"`, the standard block header. The second two bytes are an integer representing the number of bytes in the block to follow. FORM 1 is best applied when rapid data transfers, not to be modified by the computer nor interpreted by the user, are required.
2. IEEE 32-bit floating-point format, 8 bytes-per-data point. The data is preceded by the same header as in FORM 1. Each number consists of a 1-bit sign, an 8-bit biased exponent, and a 23-bit mantissa. FORM 2 is the format of choice if your computer supports single-precision floating-point numbers.
3. IEEE 64-bit floating-point format, 16 bytes-per-data point. The data is preceded by the same header as in FORM 1. Each number consists of a 1-bit sign, an 11-bit biased exponent, and a 52-bit mantissa. This format may be used with double-precision floating-point numbers. No additional precision is available in the HP 8702D data, but FORM 3 may be a convenient form for transferring data to your computer.
4. ASCII floating-point format. The data is transmitted as ASCII numbers, as described in "Output syntax" on page 1-15. There is no header. The HP 8702D uses FORM 4 to transfer data that is not related to array transfers (for example, marker responses and instrument settings).
5. PC-DOS 32-bit floating-point format with 4 bytes-per-number, 8 bytes-per-data point. The data is preceded by the same header as in FORM 1. The byte order is reversed to comply with PC-DOS formats. *If you are using a PC-based controller, FORM 5 is the most effective format to use.*

The HP 8702D terminates each transmission by asserting the EOI interface line with the last byte transmitted. Table 1-5 on page 1-19 offers a comparative overview of the five array-data formats.

Table 1-5. HP 8702D/Option 011 Array-Data Formats

Format Type	Type of Data	Bytes per Data Value	Bytes per Data Point[a]	Bytes per 201 Point Trace	Total Bytes with Header
1	Internal Binary	3	6	1206	1210
2	IEEE 32-bit Floating-Point	4	8	1608	1612
3	IEEE 64-bit Floating-Point	8	16	3216	3220
4	ASCII Numbers	24	50	10,050	10,050[b]
5	PC-DOS 32-bit Floating-Point	4	8	1608	1612

a. There are two data values (real and imaginary) for every data point.
b. No header is used in form 4.

Trace-data transfers

Transferring trace data from the HP 8702D using an instrument controller can be divided into three steps:

1. allocating an array to receive and store the data
2. commanding the HP 8702D to transmit the data
3. accepting the transferred data

Data residing in the HP 8702D is always stored in pairs for each data point (to accommodate real/imaginary pairs). The real value is first, followed by the imaginary value. Hence, the receiving array has to be two elements wide, and as deep as the number of points in the array being transferred. Memory space for the array must be declared before any data can be transferred from the HP 8702D to the computer. When reading logarithmic amplitude and phase, save the first value of each data point pair and discard the second value.

As mentioned earlier, the HP 8702D 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), because the format determines what type of data is transferred. “Data Transfer Using Markers” on page 2-18 illustrates an ASCII transfer using Form 4. For more information on the various data formats, refer to “Array-data formats” on page 1-18. For information on the various types of data that can be obtained (raw data, error-corrected data, etc.), refer to “Data levels” on page 1-23.

Frequency-related arrays

Frequency-related values are calculated for the HP 8702D displays. The only data available to the programmer are the start and stop frequencies, or center and span frequencies, of the selected frequency range.

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

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

$$F = Start\ frequency + (n-1) \times Span/(Points-1)$$

In most cases, this is an easy solution for determining the related frequency value that corresponds with a data point. This technique is illustrated in "Data Transfer Using Markers" on page 2-18.

When using log sweep or a list-frequency sweep, the points are not evenly spaced over the frequency range of the sweep. In these cases, the frequencies can be read directly out of the instrument with the `OUTPLIML` command. "Data Transfer Using Frequency-Array Information" on page 2-28 demonstrates this technique.

Executing `OUTPLIML;` reports the limit-test results by transmitting:

- the stimulus point tested
- a number indicating the limit-test results
- the upper test limit at the stimulus point (if available)
- the lower test limit at the stimulus point (if available)

The numbers used to indicate the limit-test results are:

- a negative one (–1) for no test
- a zero (0) for fail
- a positive one (1) for pass

If there are no limits available, the HP 8702D transmits zeros.

This data is very useful when testing with limit lines. It provides a method of obtaining accurate frequency-stimulus values to correspond with the trace data. The other limit-related values may be discarded and the stimulus values used with the trace-data points.

Data-Processing Chain

This section describes the manner in which the HP 8702D processes measurement data. It includes information on data arrays, common output commands, data levels, the learn string, and the calibration kit string.

Data arrays

Figure 1-1 on page 1-22 shows the different kinds of data available within the instrument:

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

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

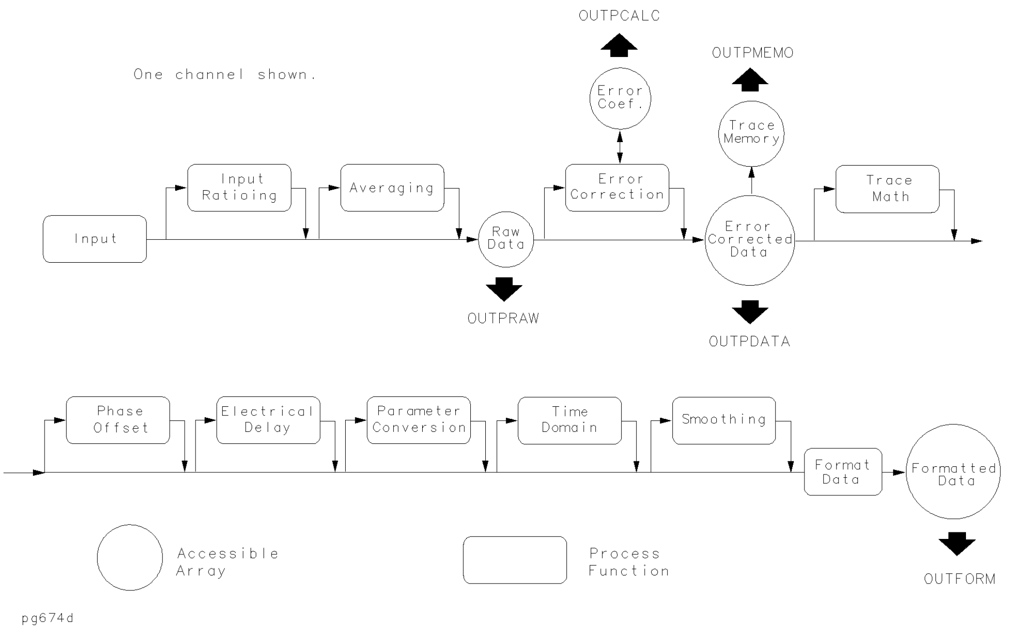


Figure 1-1. The data-processing chain

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

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

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

Data levels

Different levels of data can be read out of the instrument. Refer to the data-processing chain in Figure 1-1 on page 1-22.

The following list describes the different types of data that are available from the HP 8702D.

Raw data The basic measurement data, reflecting the stimulus parameters, IF averaging, and IF bandwidth. If a full 2-port measurement calibration is activated, there are actually four raw arrays kept: one for each raw S-parameter. The data can be output to a controller with the commands `OUTPRAW1`, `OUTPRAW2`, `OUTPRAW3`, `OUTPRAW4`. Normally, only raw 1 is available, and it holds the current parameter. If a 2-port measurement calibration is active, the four arrays refer to S_{11}, S_{21}, S_{12}, and S_{22} respectively. This data is represented in real/imaginary pairs.

Error-corrected data This is the raw data with error-correction applied. The array represents the currently measured parameter, and is stored in real/imaginary pairs. The error-corrected data can be output to a controller with the `OUTPDATA;` command. The `OUTPMEMO;` command reads the trace memory, if available. The trace memory also contains error-corrected data. Note that neither raw nor error-corrected data reflect such post-processing functions as electrical-delay offset, 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 and time domain. The units of the array output depend on the current display format. Refer to Table 1-4, “Units as a Function of Display Format,” on page 1-17 for the various units defined as a function of display format.

Calibration coefficients

The results of a measurement calibration are arrays containing calibration coefficients. These calibration coefficients are then used in the error-correction routines. Each array corresponds to a specific error term in the error model. The *HP 8702D User's Guide* details which error coefficients are used for specific calibration types, as well as the arrays those coefficients can be found in. Not all calibration types use all 12 arrays. The data is stored as real/imaginary pairs.

Generally, formatted data is the most useful of the four data levels, because it is the same information the operator sees on the display. However, if post-processing is unnecessary (possibly in cases involving smoothing), error-corrected data may be more desirable. Error-corrected data also affords the user the opportunity to input the data to the HP 8702D and apply post-processing at another time.

Learn string and calibration-kit string

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

The learn string can be output to a controller with the `OUTPLEAS;` executable, which commands the HP 8702D to start transmitting the binary string. The string has a fixed length for a given firmware revision. It can not be more than 3000 bytes in length. The array has the same header as in Form 1.

The calibration kit is a set of key characteristics of the calibration standards used to increase the calibration accuracy. There are default kits for several different connector types. There is also space for a user-defined calibration kit. The command `OUTPCALK` outputs the currently active calibration kit as a binary string in Form 1. As with the learn string, the calibration-kit string has a fixed length for a given firmware revision. It can not be longer than 1000 bytes.

Controlling Command Execution

Some HP 8702D commands require relatively longer execution times due to measurement sweeps or other processes and, occasionally, there is a need to know when certain HP 8702D operations have been completed. There is an operation-complete function (OPC) that allows a synchronization of programs with the execution of certain key commands. Table 1-6 on page 1-26 lists all the OPC-compatible commands for the instrument.

Synchronized program execution with OPC command

Issue an `OPC;` or `OPC?;` prior to an OPC-compatible command. The status byte or ESR operation-complete bit will then be set after the execution of the OPC-compatible command. For example, issuing `OPC;SING;` causes the OPC bit to be set when the single sweep is finished. Issuing `OPC?;`, in place of `OPC;`, causes the HP 8702D to output a one (1) when the command execution is complete.

Addressing the HP 8702D to talk after issuing `OPC?;` will not cause an `addressed to talk without selecting output` error, but the HP 8702D will halt the computer by not transmitting the one (1) until the command has completed. For example, executing `OPC?;PRES;`, and then immediately interrogating the HP 8702D, causes the bus to halt until the instrument preset is complete and the HP 8702D outputs a one (1).

As another example, consider the timing of sweep completion. Send the command string `SWET 3 S;OPC?;SING;` to the HP 8702D. This string sets the HP 8702D sweep time to 3 seconds, and then waits for completion of a single sweep to respond with a one (1). The computer is programmed to read the one (1) response from the HP 8702D as the completion of the single sweep. The program then waits until the sweep is completed before continuing operation. At this point a valid trace exists and the trace data can be read into the computer.

OPC-compatible commands delay execution of other commands

The HP 8702D cannot process other commands while executing OPC-compatible commands. Once a OPC-compatible command is received, the HP 8702D reads new commands into the input buffer, but it will not begin the execution of any commands until the completion of the OPC-compatible command. When the 15-character input buffer is full, the HP 8702D holds off the bus until it is able to process the commands in the buffer.

Table 1-6. OPC-Compatible Commands

CHAN	HARM	REFD
CLASS[a]	INSM	RESPDONE
CLEARALL	ISOD	REV[a]
DATI	MANTRIG	RST
DONE	NOOP	SAV
EDITDONE	NUMG	SAVC
EXTTOFF	PRES	SAVE
EXTTON	RAID	SAVEREG
EXTTPOIN	RAIRESP[a]	SING
FREQOFF	RAIISOL[a]	STAN
FWDI[a]	RECA	TRAD
FWDM[a]	RECAREG	WAIT

a. The CLASS commands are OPC-compatible if there is only one standard in the class.

Calibrating for Measurements

Measurement calibration over HP-IB follows the same command sequence as a calibration from the front panel. Since different cal kits can have a different number of standards in a given class, any automated calibration sequence is valid only for a specific cal kit. Table 1-7 on page 1-28 indicates the relationship between calibration and classes. Table 1-8 on page 1-29 describes the calibration arrays.

To calibrate the instrument

1. Select a calibration kit, such as 50 ohm type N (`CALKN50;` over HP-IB). Load a calibration standard definition from disk or enter the coefficients for O/E and E/O measurements.
2. Select a calibration type, such as S_{11} 1-port (`CALIS111;` over HP-IB).
3. Call each class used by the calibration type, such as `FORWARD: OPEN` (`CLASS11A` over HP-IB). During a 2-port calibration, the reflection, transmission, and isolation subsequences must be opened before the classes in the subsequence are called, and then closed at the end of each subsequence.
4. If a class has more than one standard in it, select a standard from the menu presented (`STANA` to `STANG` over HP-IB).

   The `STANA` to `STANG` commands are all held commands because they trigger a sweep. If a class has only one standard in it, which means that it will trigger a sweep when called, the class command will be held also.
5. If, during a calibration, two standards are measured to satisfy one class, the class must be closed with `DONE;`.
6. Declare the calibration done, such as with `DONE 1-PORT CAL` (`SAV1;` over HP-IB).

Table 1-7. Relationship Between Calibration and Classes

Class	Response	Response and Isolation	S11 1-port	S22 1-port	One path 2-port	Full 2-port	TRL*/LRM*	Response and Match E/O	Response and Match O/E
Reflection[a]:					•	•	•		
S11A, RE FW MTCH			•		•	•	•	•	•
S11B, LN FW MTCH			•		•	•	•	•	•
S11C, LN FW TRAN			•		•	•	•	•	•
S22A, LN RV MTCH				•		•	•		•
S22B, LN RV TRAN				•		•	•		•
S22C, LN RV TRAN				•		•	•		•
Transmission:					•	•	•		
Forward match					•	•	•		•
Forward trans					•	•	•	•	•
Reverse match						•	•		
Reverse trans						•	•		
Isolation:[a.]					•	•	•		
Forward					•	•	•	•	•
Reverse						•	•		
Response	•								
Response and isolation:									
Response		•							
Isolation		•							

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

Table 1-8. Calibration Arrays[a] [b]

Array	Response	Response and Isolation	Response and Match E/O	Response and Match O/E	1-port	2-port[c]	TRL*/LRM*
1	ER or ET	E_X (ED)[d]	E_{DF}	E_{DR}	E_D	E_{DF}	E_{DF}
2		E_T (ER)	E_{SF}	E_{SR}	E_S	E_{SF}	E_{SF}
3			E_{RF}	E_{RR}	E_R	E_{RF}	E_{RF}
4			E_{XF}	E_{XF}		E_{XF}	E_{XF}
5			E_{TF}	E_{LF}		E_{LF}	E_{LF}
6				E_{TF}		E_{TF}	E_{TF}
7						E_{DR}	E_{DR}
8						E_{SR}	E_{SR}
9						E_{RR}	E_{RR}
10						E_{XR}	E_{XR}
11						E_{LR}	E_{LR}
12						E_{TR}	E_{TR}

a. Meaning of first subscript: D=directivity , S=source match , R=reflection tracking, X=crosstalk, L=load match , T=transmissio n tracking.
b. Meaning of second subscript: F=forward, R=reverse.
c. One path, 2-port cal duplicates arrays 1 to 6 in arrays 7 to 12.
d. Response and isolation corrects for crosstalk and transmission tracking in transmission measurements, and for directivity and reflection tracking in reflection measurements.

Debugging Programs

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

To start debugging mode

1 Press the LOCAL key.

2 Press *HP-IB DIAG ON OFF* so that *ON* its highlighted.

Understanding File Names

Disk files created by the analyzer consist of a state name of up to eight characters, such as FILTER, appended with up to two characters. In LIF format, the file name is FILTERXX. In DOS format, the filename is FILTER.XX. The first appended character is the file type, telling the kind of information in the file. The second appended character is a data index, used to distinguish files of the same type. Data and calibration files are Form 3 data (without a header) which can be read off the disk. The other files are not meant to be decoded. Table 1-9 on page 1-32 lists the appended characters and their meanings.

Table 1-9. Disk File Names

Appended Character 1	Definition	Appended Character 2	Definition
I	Instrument state		
G	Graphics	1	Display graphics
		0	Graphics index
D	Error corrected data	1	Channel 1
		2	Channel 2
R	Raw data	1 to 4	Channel 1, raw arrays 1 to 4
		5 to 8	Channel 2, raw arrays 1 to 4
F	Formatted data	1	Channel 1
		2	Channel 2
M	Memory trace	1	Channel 1
		2	Channel 2
P	Instrument state appendix		
C	Cal kit	K	
1	Cal data, channel 1	0	Stimulus state
		1 to 9	Coefficients 1 to 9
		A	Coefficient 10
		B	Coefficient 11
		C	Coefficient 12
2	Cal data, channel 2	0 to C	Same as channel 1

Drawing Graphics on the Display

You can use the HP 8702D's screen as a graphics display for showing connection diagrams or custom instructions to an operator. For programming purposes, consider the display as a separate device; it has its own HP-IB address. The display's address is configured automatically based on the address of the HP 8702D.

The display's default HP-IB address is 17. If you change the HP 8702D's default address, you can determine the display's new address with the following instructions. If the HP 8702D address is an even number, add 1 to the address. If the HP 8702D address is an odd number, subtract 1 from the address.

The display accepts a subset of Hewlett-Packard Graphics Language (HP-GL) commands. These commands are documented in Chapter 4, "Graphics Language Reference". Some HP-GL commands are accepted but ignored. These commands are documented in Table 1-10 on page 1-34.

When using the graphics commands, you will need to specify positions on the display. The origin (0, 0) is located in the display's lower left corner.

Graticule:

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

Complete Display (includes annotation and softkeys):

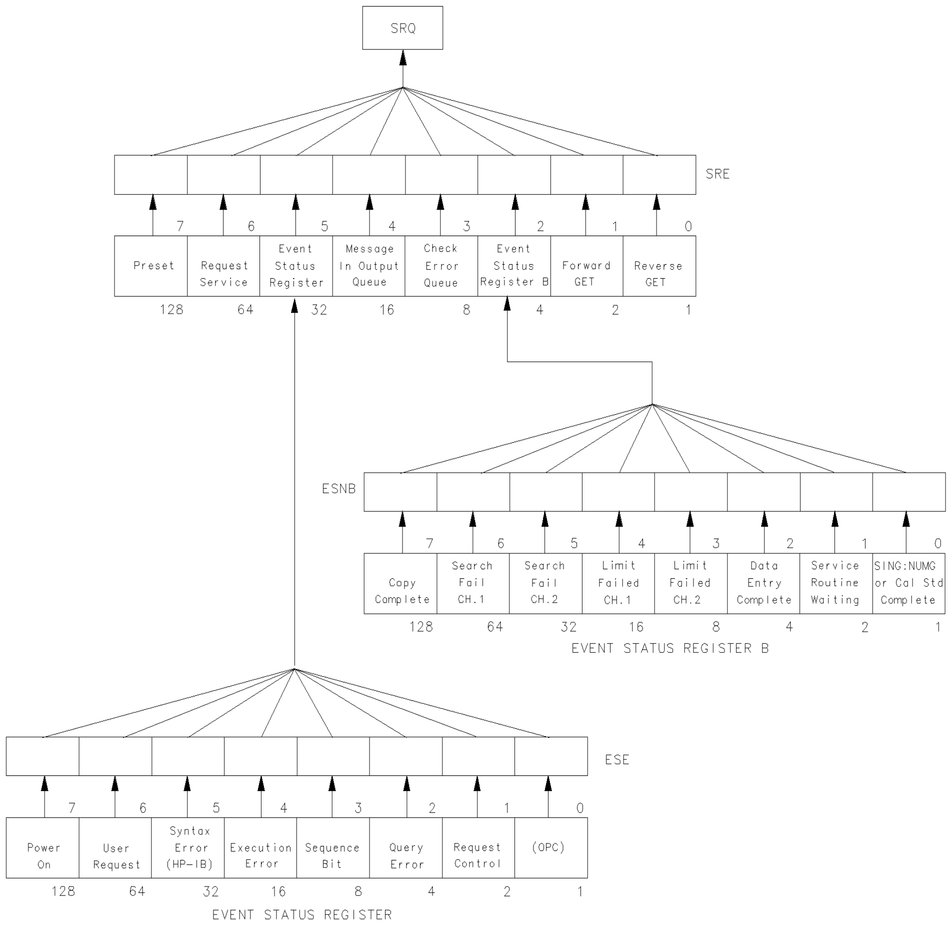
- length: 0 to 5850
- height: 0 to 4095

Table 1-10. Ignored but Accepted HP-GL Commands

Command	Description
IM	Input service request mask
IP	Input P1,P2 scaling points
IW	Input window
OC	Output current pen position
OE	Output error
OI	Output identity
OS	Output status
SL	Character slant
SR	Relative character size

Monitoring the Instrument

Almost every program that you write will need to monitor the HP 8702D for its operating status and errors. This includes querying execution or command errors and determining whether or not measurements have been completed. Several status registers and queues are provided to accomplish these tasks. In this section, you'll learn how to enable and read these registers. You'll also learn about reading the error messages in the error queue.



pg6151d

Figure 1-2. Status reporting structure

The Status Byte

The analyzer has a status-reporting mechanism that reports information about specific analyzer functions and events. The status byte (consisting of summary bits) is the top-level register. Each bit reflects the condition of another register or queue. If a summary bit is set (equals 1), the corresponding register or queue should be read to obtain the status information and clear the condition. Reading the status byte does not affect the state of the summary bits. The summary bits always reflect the condition of the summarized queue or register. The status byte can be read by a serial poll or by using the command `OUTPSTAT`. When using this command, the sequencing bit can be set by the operator during the execution of a test sequence. `OUTPSTAT` does not automatically put the instrument in remote mode, thus giving the operator access to the analyzer's front-panel functions.

The status byte:

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

Any bit in the status byte can generate a service request

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

The status byte summarizes two queues

The status byte also summarizes two queues: the output queue and the error queue. (The error queue is described in "Error Messages" on page 1-39.) When the analyzer outputs information, it puts the information in the output queue where it resides until the controller reads it. The output queue is only one event long. Therefore, the next output request will clear the current data. The summary bit is set whenever there is data in the output queue.

Table 1-11. Bits in the Status Byte

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

The Event-Status Register and Event-Status-Register B

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

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

1 The event-status register is read, clearing the latched bit.

2 The summary bit in the status byte is disabled.

3 The event-status-register bit is disabled.

4 The status registers are cleared with the `CLES;` command.

5 An instrument preset is performed.

Service requests generated when there are error messages or when the instrument is waiting for the Group Execute Trigger (GET) command are cleared by:

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

Table 1-12. Bits in the Event-Status Register

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

Table 1-13. Bits in the Event-Status Register B

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

Error Messages

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

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

Response to IEEE-488 Universal Commands

Abort The HP 8702D responds to the abort message (IFC) by halting all listener, talker, and controller functions.

Device Clear The HP 8702D responds to the device clear commands (DCL, SDC) by clearing the input and output queues, and clearing any HP-IB errors. The status registers and the error queue are unaffected.

Local The HP 8702D will go into local mode if: the local command (GTL) is received, the remote line is unasserted, or the front-panel LOCAL key is pressed. Changing the HP 8702D's HP-IB status from remote to local does not affect any of the front-panel functions or values.

Local Lockout If the HP 8702D receives the local-lockout command (LLO) while it is in remote mode, it will disable the entire front panel except for the line power switch. A local-lockout condition can only be cleared by releasing the remote line, although the local command (GTL) will place the instrument temporarily in local mode.

Parallel Poll The HP 8702D does not respond to parallel-poll configure (PPC) or parallel-poll unconfigure (PPU) messages.

Pass Control If the HP 8702D is in pass-control mode, is addressed to talk, and receives the take-control command (TCT) from the system, it will take active control of the bus. If the HP 8702D is not requesting control, it will immediately pass control to the system controller's address.

Remote The HP 8702D will go into remote mode when the remote line is asserted and the HP 8702D is addressed to listen. While the HP 8702D is held in remote mode, all front-panel keys (with the exception of LOCAL) are disabled. Changing the HP 8702D's HP-IB status from remote to local does not affect any front-panel settings or values.

Serial Poll

The HP 8702D will respond to a serial poll with its status byte, as defined in the “The Status Byte” on page 1-36. To initiate the serial-poll sequence, address the HP 8702D to talk and issue a serial-poll enable command (SPE). Upon receiving this command, the HP 8702D will return its status byte. End the sequence by issuing a serial-poll disable command (SPD). A serial poll does not affect the value of the status byte, and it does not set the instrument to remote mode.

Trigger

In hold mode, the HP 8702D responds to device trigger by taking a single sweep. If a one-path, 2-port measurement calibration is active, the HP 8702D will set the waiting-for-Group-Execute-Trigger bits in the status byte. If waiting-for-forward-GET is set, the HP 8702D will assume the device is connected for forward measurement and take a sweep when GET is received. Similarly, if the waiting-for-reverse-GET bit is set, the HP 8702D will assume the device is connected for reverse measurement. The HP 8702D responds only to selected-device trigger (SDT). This means that it will not respond to group execute-trigger (GET) unless it is addressed to listen. The HP 8702D will not respond to GET if it is not in hold mode.

2

Examples

Programming Examples

The examples documented in this chapter can be found on the disk that was provided with your instrument. The example disk includes versions of these programs for each of the following languages:

- HP BASIC (LIF format)
- Microsoft's QuickBASIC (DOS format)

  Microsoft® is a U.S. registered trademark of Microsoft Corp.
- Microsoft's QuickC (DOS format)

For the HP BASIC language, examples are written for the HP 82335B interface card.

For the QuickBASIC and QuickC languages, examples are provided for both the HP 82335B Interface and Command Library card and the National Instruments card.

What you'll find in this chapter

Preparing Measurement Settings

File Name EXAMP1A.BAS

Description Use the same command sequence used for manually setting up measurements for remotely setting up measurements via HP-IB. There is no required order, as long as you set the desired frequency range, number of points, and power level prior to performing a measurement calibration.

This example sets the following parameters:

- reflection log magnitude on channel 1
- reflection phase on channel 2
- dual channel display mode
- frequency range from 100 MHz to 500 MHz

The following is the program's algorithm:

- The system is initialized.
- The analyzer is adjusted to measure return loss on channel 1 and display it in log magnitude.
- The analyzer is adjusted to measure return loss on channel 2 and display the phase.
- The dual-channel display mode is activated.
- The system operator is prompted to enter the frequency range of the measurement.
- The displays are autoscaled.
- The analyzer is released from remote control and the program ends.

This example program initializes the analyzer and the operator is queried for the measurement's start and stop frequencies. The analyzer is set up to display the S_{11} reflection measurement as a function of log magnitude and phase over the selected frequency range. The displays are autoscaled and the program ends.

```
10  ! This program selects the S-parameter to be measured, the display
20  ! format and then sets the specified start and stop frequencies.
30  ! The analyzer display is then autoscaled.
40  !
50  ! EXAMP1A
60  !
70  ASSIGN @Nwa TO 716                   ! Assign an I/O path for the analyzer
80  !
90  CLEAR SCREEN
100 ! Initialize the system
110 ABORT 7                              ! Generate an IFC (Interface Clear)
120 CLEAR @Nwa                           ! SDC (Selected Device Clear) analyzer
130 OUTPUT @Nwa;”OPC?;PRES;”             ! Preset the analyzer and wait
140 ENTER @Nwa;Reply                     ! Read in the 1 returned
150 !
160 ! Set up measurement and display
170 OUTPUT @Nwa;”CHAN1;”                 ! Channel 1
180 OUTPUT @Nwa;”S11;”                   ! Return Loss measurement
190 OUTPUT @Nwa;”LOGM;”                  ! Log magnitude display
200 !
210 OUTPUT @Nwa;”CHAN2;”                 ! Channel 2
220 OUTPUT @Nwa;”S11;”                   ! Return Loss measurement
230 OUTPUT @Nwa;”PHAS;”                  ! Phase display
240 !
250 OUTPUT @Nwa;”DUACON;”                ! Dual channel display
260 !
270 ! Request start and stop frequency
280 INPUT “ENTER START FREQUENCY (MHz):”,F_start
290 INPUT “ENTER STOP FREQUENCY (MHz):”,F_stop
300 !
310 ! Program the analyzer settings
320 OUTPUT @Nwa;”STAR”;F_start;”MHZ;”    ! Set the start frequency
330 OUTPUT @Nwa;”STOP”;F_stop;”MHZ;”     ! Set the stop frequency
340 !
350 ! Autoscale the displays
360 OUTPUT @Nwa;”CHAN1;AUTO;”            ! Autoscale channel 1 display
370 OUTPUT @Nwa;”CHAN2;AUTO;”            ! Autoscale channel 2 display
380 !
390 OUTPUT @Nwa;”OPC?;WAIT;”             ! Wait for the analyzer to finish
400 ENTER @Nwa;Reply                     ! Read the 1 when complete
410 LOCAL @Nwa                           ! Release HP-IB control
420 END
```

Verifying Measurement Settings

File Name EXAMP1B.BAS

Description This example shows how to read analyzer settings into your program. Information on the command formats and operations is located in Chapter 1, "Writing Programs". Appending a question mark (?) to a command that sets an analyzer parameter will return the value of that setting from the analyzer to the controller. Parameters that are set as ON or OFF when queried will return a one (1) if active or a zero (0) if OFF. Parameters are returned in ASCII format, Form 4. This format varies in length from 1 to 24 characters-per-value. In cases of marker or other multiple responses, these values are separated by commas.

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

- The system is initialized.
- The number of points in the trace is queried and printed on the controller display.
- The start frequency is queried and printed on the controller display.
- The averaging function state is queried and printed on the controller display.
- The analyzer is released from remote control and the program ends.

The analyzer is preset. The preset values are returned for the number of points, the start frequency, and the state of the averaging function. The analyzer is released from remote control and the program ends.

```
10  ! This program performs some example queries of analyzer
20  ! settings. The number of points in a trace, the start frequency
30  ! and, if averaging is turned on, are determined and displayed.
40  !
50  ! EXAMP1B
60  !
70  ASSIGN @Nwa TO 716                   ! Assign an I/O path for the analyzer
80  !
90  CLEAR SCREEN
100 ! Initialize the system
110 ABORT 7                              ! Generate an IFC (Interface Clear)
120 CLEAR @Nwa                           ! SDC (Selected Device Clear)
130 OUTPUT @Nwa;”OPC?;PRES;”             ! Preset the analyzer and wait
140 ENTER @Nwa;Reply                     ! Read in the 1 returned
150 !
160 ! Query analyzer parameters
170 OUTPUT @Nwa;”POIN?;”                 ! Read in the default trace length
180 ENTER @Nwa;Num_points
190 PRINT “Number of points “;Num_points
200 PRINT
210 !
220 OUTPUT @Nwa;”STAR?;”                 ! Read in the start frequency
230 ENTER @Nwa;Start_f
240 PRINT “Start Frequency “;Start_f
250 PRINT
260 !
270 OUTPUT @Nwa;”AVERO?;”               ! Averaging on?
280 ENTER @Nwa;Flag
290 PRINT “Flag =”;Flag;”   “;
300 IF Flag=1 THEN                       ! Test flag and print analyzer state
310   PRINT “Averaging ON”
320 ELSE
330   PRINT “Averaging OFF”
340 END IF
350 !
360 OUTPUT @Nwa;”OPC?;WAIT;”             ! Wait for the analyzer to finish
370 ENTER @Nwa;Reply                     ! Read the 1 when complete
380 LOCAL @Nwa                           ! Release HP-IB control
390 END
```

S11 1-Port Measurement Calibration

File Name EXAMP2A.BAS

Description This section shows you how to coordinate a measurement calibration over HP-IB. You can use the following sequence for performing either a manual measurement calibration, or a remote measurement calibration via HP-IB:

1 Select the calibration type.
2 Measure the calibration standards.
3 Declare the calibration done.

The actual sequence depends on the calibration kit and changes slightly for full 2-port measurement calibrations, which are divided into three calibration sub-sequences.

Calibration Kits

The calibration kit tells the analyzer what standards to expect at each step of the calibration. The set of standards associated with a given calibration is termed a “class”. For example, measuring the short during an S_{11} 1-port measurement calibration is one calibration step. All of the shorts that can be used for this calibration step make up the class, which is called class S11B. For the 7-mm and 3.5-mm cal kits, class S11B uses only one standard. For Type-N cal kits, class S11B contains two standards: male and female shorts.

When doing an S_{11} 1-port measurement calibration, use a 7-mm or 3.5-mm calibration kit. Selecting *SHORT* automatically measures the short because the class contains only one standard. When doing the same calibration in Type-N, selecting *SHORT* brings up a second menu, allowing the operator to select which standard in the class is to be measured. The sex listed refers to the test port. If the test port is female, then the operator selects the female short option.

Doing an S_{11} 1-port measurement calibration over HP-IB is very similiar. When using a 7-mm or 3.5-mm calibration kit, sending CLASS11B will automatically measure the short. In Type-N, sending CLASS11B brings up the menu with the male and female short options. To select a standard, use STANA or STANB. The STAN command is appended with the letters A through G, corresponding to the standards listed under softkeys 1 through 7, softkey 1 being the topmost softkey.

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

Example

This example shows you how to coordinate an S_{11} 1-port measurement calibration over HP-IB, using the HP 85032B 50Ω Type-N calibration kit.

1. Set up the desired instrument state.
2. Run the program.
3. Connect the standards as prompted.
4. Press ENTER on the controller keyboard to measure the standard.

NOTE

Some computers may have a RETURN hardkey as the activating switch for carriage return/line feed. Throughout this section, the carriage-return/line-feed hardkey is represented by ENTER.

Information on selecting calibration standards can be found in Chapter 1, “Writing Programs”.

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

- The system is initialized.
- The appropriate calibration kit is selected.
- The softkey menu is deactivated.
- The S_{11} 1-port measurement calibration sequence is run.
- The S_{11} 1-port measurement calibration data is saved.
- The softkey menu is activated.
- The analyzer is released from remote control and the program ends.

Running the Program

This program assumes the following test port characteristics:

- 50 ohm
- Type-N connectors
- Female

The prompts appear just above the message line on the analyzer's and the controller's display. Pressing ENTER on the controller continues the program and measures the standard. The program displays a message when the measurement calibration is complete.

```
10  ! This program performs a 1-port calibration on the HP 8702D.
20  ! It guides the operator through a 1-port calibration
30  ! using the HP 85032B 50 ohm type N calibration kit.
40  !
50  ! The routine Waitforkey displays a message on the instrument’s
60  ! display and the console, to prompt the operator to connect the
70  ! calibration standard. Once the standard is connected, the
80  ! ENTER key on the computer keyboard is pressed to continue.
90  !
100 ! EXAMP2A
110 !
120 ASSIGN @Nwa TO 716                   ! Assign an I/O path for the analyzer
130 !
140 CLEAR SCREEN
150 ! Initialize the system
160 ABORT 7                              ! Generate an IFC (Interface Clear)
170 CLEAR @Nwa                           ! SDC (Selected Device Clear)
180 !
190 OUTPUT @Nwa;”CALKN50;”               ! Select CAL kit type
200 OUTPUT @Nwa;”MENUOFF;”               ! Turn softkey menu off.
210 !
220 OUTPUT @Nwa;”CALIS111;”              ! S11 1 port CAL initiated
230 !
240 CALL Waitforkey(“CONNECT OPEN AT PORT 1”)
250 OUTPUT @Nwa;”CLASS11A;”              ! Open reflection CAL
260 OUTPUT @Nwa;”OPC?;STANB;”            ! Select the second standard, B
270 ENTER @Nwa;Reply                     ! Read in the 1 returned
280 !
290 CALL Waitforkey(“CONNECT SHORT AT PORT 1”)
300 OUTPUT @Nwa;”CLASS11B;”              ! Short reflection CAL
310 OUTPUT @Nwa;”OPC?;STANB;”            ! Select the second standard, B
320 ENTER @Nwa;Reply                     ! Read in the 1 returned
330 !
340 CALL Waitforkey(“CONNECT LOAD AT PORT 1”)
350 OUTPUT @Nwa;”OPC?;CLASS11C;”         ! Reflection load CAL
360 ENTER @Nwa;Reply                     ! Read in the 1 returned
370 !
380 OUTPUT 717;”PG;”                     ! Clear the analyzer display
390 !
400 DISP “COMPUTING CALIBRATION COEFFICIENTS”
410 !
420 OUTPUT @Nwa;”DONE;”                  ! Finished with the CAL cycle
430 OUTPUT @Nwa;”OPC?;SAV1;”             ! Save the ONE PORT CAL
440 ENTER @Nwa;Reply                     ! Read in the 1 returned
450 !
460 DISP “S11 1-PORT CAL COMPLETED. CONNECT TEST DEVICE.”
470 OUTPUT @Nwa;”MENUON;”                ! Turn on the softkey menu
480 !
490 OUTPUT @Nwa;”OPC?;WAIT;”             ! Wait for the analyzer to finish
500 ENTER @Nwa;Reply                     ! Read the 1 when complete
510 LOCAL @Nwa                           ! Release HP-IB control
520 !
530 END
540 !
550 ! *************************** Subroutines ******************************
560 !
```

```
570 Waitforkey:  ! Prompt routine to read a keypress on the controller
580 SUB Waitforkey(Lab$)
590   !   Position and display text on the analyzer display
600   OUTPUT 717;"PG;PU;PA390,3700;PD;LB";Lab$;", PRESS ENTER WHEN READY_;"
610   !
620   DISP Lab$&"  Press ENTER when ready"; ! Display prompt on console
630   INPUT A$                              ! Read ENTER key press
640   !
650   OUTPUT 717;"PG;"                      ! Clear analyzer display
660 SUBEND
```

Full 2-Port Measurement Calibration

File Name EXAMP2B.BAS

Description The following example shows how to perform a full 2-port measurement calibration using the HP 85032B calibration kit. The main difference between this example and the S_{11} 1-Port Measurement Calibration example is that the full 2-port measurement calibration process allows removal of both the forward- and reverse-error terms, so that all four S-parameters of the device under test can be measured. PORT 1 is a female test port and PORT 2 is a male test port.

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

- The system is initialized.
- The appropriate calibration kit is selected.
- The softkey menu is deactivated.
- The 2-port calibration sequence is run.
- The operator is prompted to choose or skip the isolation calibration.
- The softkey menu is activated.
- The analyzer is released from remote control and the program ends.

Running the Program

The program assumes the following situation:

- test ports are Type-N
- PORT 1 is a female test port
- PORT 2 is a male test port
- HP 85032B 50Ω Type-N calibration kit is used

The prompts appear just above the message line on the analyzer's display, as well as the computer's console. Pressing ENTER on the computer keyboard continues the program and measures the standard. You have the option of omitting the isolation portion of the measurement calibration. If you perform the isolation portion of the calibration, averaging is automatically employed to insure a valid calibration. The program will display a message when the measurement calibration is complete.

```
10  ! This program performs a full 2-port measurement calibration.
20  ! It guides the operator through a full 2-port calibration
30  ! using the HP 85032B 50 ohm type N calibration kit.
40  ! The routine Waitforkey displays a message on the instrument’s
50  ! display and the console to prompt the operator to connect the
60  ! calibration standard. Once the standard is connected, the
70  ! ENTER key on the computer keyboard is pressed to continue.
80  !
90  ! EXAMP2B
100 !
110 ASSIGN @Nwa TO 716                    ! Assign an I/O path to the analyzer
120 !
130 CLEAR SCREEN
140 ! Initialize the analyzer
150 ABORT 7                               ! Generate an IFC (Interface Clear)
160 CLEAR @Nwa                            ! SDC (Selected Device Clear)
170 !
180 OUTPUT @Nwa;”CALKN50;MENUOFF;”     ! Select CAL kit type and turn off menu
190 !
200 OUTPUT @Nwa;”CALIFUL2;”               ! Full 2 port CAL
210 !
220 OUTPUT @Nwa;”REFL;”                   ! Reflection CAL
230 !
240 CALL Waitforkey(“CONNECT OPEN AT PORT 1”)
250 OUTPUT @Nwa;”CLASS11A;”              ! S11 open CAL
260 OUTPUT @Nwa;”OPC?;STANB;”             ! Select the second standard, B
270 ENTER @Nwa;Reply                      ! Read in the 1 returned
280 !
290 CALL Waitforkey(“CONNECT SHORT AT PORT 1”)
300 OUTPUT @Nwa;”CLASS11B;”               ! S11 short CAL
310 OUTPUT @Nwa;”OPC?;STANB;”             ! Select the second standard, B
320 ENTER @Nwa;Reply                      ! Read in the 1 returned
330 !
340 CALL Waitforkey(“CONNECT LOAD AT PORT 1”)
350 OUTPUT @Nwa;”OPC?;CLASS11C;”         ! S11 load CAL
360 ENTER @Nwa;Reply                      ! Read in the 1 returned
370 !
380 CALL Waitforkey(“CONNECT OPEN AT PORT 2”)
390 OUTPUT @Nwa;”CLASS22A;”              ! S22 open CAL
400 OUTPUT @Nwa;”OPC?;STANA;”             ! Select the first standard, A
410 ENTER @Nwa;Reply                      ! Read in the 1 returned
420 !
430 CALL Waitforkey(“CONNECT SHORT AT PORT 2”)
440 OUTPUT @Nwa;”CLASS22B;”               ! S22 short CAL
450 OUTPUT @Nwa;”OPC?;STANA;”             ! Select the first standard, A
460 ENTER @Nwa;Reply                      ! Read in the 1 returned
470 !
480 CALL Waitforkey(“CONNECT LOAD AT PORT 2”)
490 OUTPUT @Nwa;”OPC?;CLASS22C;”         ! S22 load CAL
500 ENTER @Nwa;Reply
510 !
520 DISP “COMPUTING REFLECTION CALIBRATION COEFFICIENTS”
530 !
540 OUTPUT @Nwa;”REFD;”                   ! Reflection portion complete
550 !
560 OUTPUT @Nwa;”TRAN;”                   ! Transmission portion begins
```

```
570 !
580 CALL Waitforkey(“CONNECT THRU [PORT1 TO PORT 2]”)
590 DISP “MEASURING FORWARD TRANSMISSION”
600 OUTPUT @Nwa;”OPC?;FWDT;”              ! Measure forward transmission
610 ENTER @Nwa;Reply                      ! Read in the 1 returned
620 !
630 OUTPUT @Nwa;”OPC?;FWDM;”              ! Measure forward load match
640 ENTER @Nwa;Reply                      ! Read in the 1 returned
650 !
660 DISP “MEASURING REVERSE TRANSMISSION”
670 OUTPUT @Nwa;”OPC?;REVT;”              ! Measure reverse transmission
680 ENTER @Nwa;Reply                      ! Read in the 1 returned
690 !
700 OUTPUT @Nwa;”OPC?;REVM;”              ! Measure reverse load match
710 ENTER @Nwa;Reply                      ! Read in the 1 returned
720 !
730 OUTPUT @Nwa;”TRAD;”                   ! Transmission CAL complete
740 !
750 INPUT “SKIP ISOLATION CAL? Y OR N.”,An$
760 IF An$=”Y” THEN
770   OUTPUT @Nwa;”OMII;”                 ! Skip isolation cal
780   GOTO 940
790 END IF
800 !
810 CALL Waitforkey(“ISOLATE TEST PORTS”)
820 !
830 OUTPUT @Nwa;”ISOL;”                   ! Isolation CAL
840 OUTPUT @Nwa;”AVERFACT10;”             ! Average for 10 sweeps
850 OUTPUT @Nwa;”AVEROON;”               ! Turn on averaging
860 DISP “MEASURING REVERSE ISOLATION”
870 OUTPUT @Nwa;”OPC?;REVI;”              ! Measure reverse isolation
880 ENTER @Nwa;Reply                      ! Read in the 1 returned
890 !
900 DISP “MEASURING FORWARD ISOLATION”
910 OUTPUT @Nwa;”OPC?;FWDI;”              ! Measure forward isolation
920 ENTER @Nwa;Reply                      ! Read in the 1 returned
930 !
940 OUTPUT @Nwa;”ISOD;AVEROOFF;”          ! Isolation complete averaging off
950 OUTPUT 717;”PG;”                      ! Clear analyzer display prompt
960 !
970 DISP “COMPUTING CALIBRATION COEFFICIENTS”
980 OUTPUT @Nwa;”DONE;”                   ! End the CAL sequence
990 OUTPUT @Nwa;”OPC?;SAV2;”              ! Save THE TWO PORT CAL
1000 ENTER @Nwa;Reply                     ! Read in the 1 returned
1010!
1020 DISP “DONE WITH FULL 2-PORT CAL. CONNECT TEST DEVICE.”
1030 OUTPUT @Nwa;”MENUON;”                ! Turn softkey menu on
1040 !
1050 OUTPUT @Nwa;”OPC?;WAIT;”             ! Wait for the analyzer to finish
1060 ENTER @Nwa;Reply                     ! Read the 1 when complete
1070 LOCAL @Nwa                           ! Release HP-IB control
1080!
1090 END
1100!
1110! ************************* Subroutines *******************************
1120!
```

```
1130 SUB Waitforkey(Lab$)
1140   ! Position and display prompt on the analyzer display
1150  OUTPUT 717;”PG;PU;PA390,3700;PD;LB”;Lab$;”, PRESS ANY KEY WHEN READY_;”
1160   !
1170   DISP Lab$&”   Press ENTER when ready”;   ! Display prompt on console
1180   INPUT A$                            ! Read ENTER keypress on controller
1190   OUTPUT 717;”PG;”                    ! Clear analyzer display
1200 SUBEND
```

Data Transfer Using Markers

File Name EXAMP3A.BAS

Description There are two methods that can be used to read trace information from the analyzer:

- selectively, using the trace markers
- completely, using the trace-data array

If only specific information (such as a single point on the trace or the result of a marker search) is required, the marker output command can be used to read the information.

Trace-Data Formats and Transfers

Refer to Table 1-5, “HP 8702D/Option 011 Array-Data Formats,” on page 1-19. This table shows the number of bytes required to transfer a 201-point trace in the different formats. As you will see in the first example Form 4, ASCII data is the easiest to transfer, but the most time consuming due to the number of bytes in the trace. If you are using a PC-based controller, a more suitable format would be Form 5. To use any trace data format other than Form 4 (ASCII data) requires some care in transferring the data to the computer. Data types must be matched to read the bytes from the analyzer directly in to the variable array. The computer must be told to stop formatting the incoming data and treat it as a binary-data transfer. All of the other data formats also have a four byte header to deal with. The first two bytes are the ASCII characters `"#A"` that indicate that a fixed length block transfer follows, and the next two bytes form an integer containing the number of bytes in the block to follow. The header must be read in to separate the header from the rest of the block data to be mapped into an array. “Array-data formats” on page 1-18, discusses the different types of formats and their compositions.

Data may also be transferred from several different locations in the trace-processing chain. These examples will illustrate formatted-data transfers, but other locations in the trace-data processing chains may be accessed. Refer to Figure 5-1 on page 5-5.

In this section, an example of each of the data formats will be shown for comparison. A general rule of thumb is to use Form 1 (internal binary format) for traces that are not being utilized for data content. Learn strings, state transfers, and calibration data that are being transferred to a file and back are good examples. Refer to “Data Transfer Using Frequency-Array Information” on page 2-28.

Arrays which will be interpreted or processed within your program should be in Form 2, 3 or 5, whichever is appropriate for your computer. “Data Transfer Using Floating-Point Numbers” on page 2-25 shows how to transfer a trace in these formats.

In “Data Transfer Using ASCII Format” on page 2-22 and “Data Transfer Using Floating-Point Numbers” on page 2-25, the frequency counterpart of each data point in the array is also determined. Many applications generate a frequency and magnitude, or a phase array for the test results. Such data may be required for other data processing applications (such as comparing data from other measurements).

In “Data Transfer Using ASCII Format” on page 2-22, the frequency data is constructed from the frequency span information. Alternatively, it is possible to read the frequencies directly out of the instrument with the `OUTPLIML` command. `OUTPLIML` reports the limit-test results by transmitting the stimulus point tested, a number indicating the limit-test results, and then the upper and lower limits at that stimulus point (if available). The number indicating the limit results is a –1 for no test, 0 for fail, and 1 for pass. If there are no limits available, the analyzer transmits zeros. For this example, we delete the limit test information and keep the stimulus information.

In “Data Transfer Using Floating-Point Numbers” on page 2-25, the limit-test array is read into the controller and used to provide the values directly from the analyzer memory. Reading from the limit-test array is convenient, although it outputs the results in ASCII format (Form 4), which may be slow. If there is no other way to obtain the frequency data, this transfer time may be acceptable. Frequency information becomes more difficult to determine when not using the linear sweep mode. Log-frequency sweeps and list-frequency sweeps have quite different values for each data point. For these special cases, the additional time spent reading out the limit test results is an acceptable solution for obtaining the valid frequency information for each data point in the trace.

Example

Markers are the simplest form of trace-data transfer. You can position a marker in the following locations on the trace:

- a frequency location
- an actual data point location
- a trace data value

The marker data is always returned in Form 4, ASCII format. Each number is sent as a 24-character string; each character being a digit, sign, or decimal point. In the case of markers, three numbers are sent. The display format determines the values of the marker responses. Refer to Figure 5-1 on page 5-5 for additional information.

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

- The system is initialized.
- The selected frequency span is swept once.
- The marker is activated and placed on the maximum trace value.
- The three marker values (value 1, value 2, and stimulus) are output to the controller and displayed.
- The instrument is returned to local control and the program ends.

Running the Program

Execute the program. After performing an instrument preset, the analyzer switches into the log-magnitude mode and measures the S_{11} reflection values of the device under test.

The three values returned to the controller are:

1. reflection, in dB
2. a non-significant value
3. the stimulus frequency at the maximum point

A non-significant value means that the analyzer returned a value that is meaningless in this data format.

Table 5-2 on page 5-6, provides an easy reference for the types of data returned with the various data-format operational modes.

```
10  ! This program takes a sweep on the analyzer and turns on a marker.
20  ! The marker is positioned on the trace maximum and the marker data
30  ! is output in ASCII format.
40  !
50  ! EXAMP3A
60  !
70  ASSIGN @Nwa TO 716                    ! Assign an I/O path for the analyzer
80  !
90  CLEAR SCREEN
100 ! Initialize the analyzer
110 ABORT 7                               ! Generate an IFC (Interface Clear)
120 CLEAR @Nwa                            ! SDC (Selective Device Clear)
130 OUTPUT @Nwa;”OPC?;PRES;”              ! Preset the analyzer and wait
140 ENTER @Nwa;Reply                      ! Read in the 1 returned
150 !
160 OUTPUT @Nwa;”OPC?;SING”               ! Single sweep mode and wait
170 ENTER @Nwa;Reply                      ! Read 1 when sweep complete
180 !
190 OUTPUT @Nwa;”MARK1;”                  ! Turn on marker 1
200 OUTPUT @Nwa;”SEAMAX;”                 ! Find the maximum
210 !
220 OUTPUT @Nwa;”OUTPMARK;”               ! Request the current marker value
230 ENTER @Nwa;Value1,Value2,Stim         ! Read three marker values
240 !
250 ! Show the marker data received.
260 PRINT “ Value 1”,” Value 2”,”     Stimulus (Hz)”
270 PRINT Value1,Value2,Stim              ! Print the received values
280 PRINT
290 PRINT “ Compare the active marker block with the received values”
300 !
310 LOCAL @Nwa                            ! Release HP-IB control
320 END
```

Data Transfer Using ASCII Format

File Name EXAMP3B.BAS

Description This example shows you how to transfer a trace array from the analyzer using Form 4, an ASCII data transfer.

Table 5-2 on page 5-6 shows the relationship of the two values-per-point that are transferred to the analyzer. When Form 4 is used, each number is sent as a 24-character string (each character represented by a digit, sign, or decimal point). Since there are two numbers-per-point, plus a comma and line-feed, a 201-point transfer in Form 4 takes 10,050 bytes. This form is useful only when input-data formatting is difficult with the instrument controller. Refer to Table 1-5, “HP 8702D/Option 011 Array-Data Formats,” on page 1-19 for a comparison with the other formats.

Another example is included with the ASCII data transfer. A fairly common requirement is to create frequency-amplitude data pairs from the trace data. No frequency information is included with the trace-data transfer. Relating the data from a linear frequency sweep to frequency can be done by interrogating the analyzer start frequency, the frequency span, and the number of points. Given that information, the frequency of point *n* in a linear frequency sweep is defined by the equation:

$$F = \mathit{Start\ frequency} + (n-1) \times \mathit{Span}/(\mathit{Points}-1)$$

This example illustrates the technique of generating corresponding frequency data. This is a straight-forward solution for linear uniform sweeps. For other sweep types, frequency data is more difficult to construct and may be best read from the analyzer directly from the limit-test array. See “Data Transfer Using Frequency-Array Information” on page 2-28 for an example of this technique.

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

- The system is initialized.
- The trace-data array is allocated.
- The trace length is set to 11.
- The selected frequency span is swept once.
- The Form 4, ASCII format is set.
- The formatted trace is read from the analyzer.
- The frequency increments between the points are calculated.
- The marker is activated and placed at 30 kHz.
- The instrument is returned to local control and the program ends.

Running the Program

Run the program and watch the controller console. The analyzer will perform an instrument preset. The program will then print out the data values received from the analyzer. The marker is activated and placed at the left-hand edge of the analyzer display. Position the marker with the knob and compare the values read with the active marker with the results printed on the controller console. The data points should agree exactly. Keep in mind that no matter how many digits are displayed, the analyzer is specified to measure:

- magnitude to a resolution of 0.001 dB
- phase to a resolution of 0.01 degrees
- group delay to a resolution of 0.01 ps

Changing the display format will change the data sent with the `OUTPFORM` transfer. See Table 1-5, "HP 8702D/Option 011 Array-Data Formats," on page 1-19 for a list of the specific data that is provided with each format. The data from `OUTPFORM` reflects all the post processing such as:

- time domain
- gating
- electrical delay
- trace math
- smoothing

NOTE

Note that if time domain (option 110 only) is deactivated, operation is limited to 201 points in the lowpass mode.

```
10  ! This program shows an ASCII format trace data transfer using form 4.
20  ! The data is received as a string of ASCII characters, 24 characters
30  ! per data point and transferred into a real array in the controller. The
40  ! corresponding frequency data is calculated from the analyzer settings.
50  !
60  ! EXAMP3B
70  !
80  ASSIGN @Nwa TO 716                    ! Assign an I/O path to the analyzer
90  !
100 CLEAR SCREEN
110 ! Initialize
120 ABORT 7                               ! Generate an IFC (Interface Clear)
130 CLEAR @Nwa                            ! SDC (Selective Device Clear)
140 OUTPUT @Nwa;”OPC?;PRES;”              ! Preset the analyzer
150 ENTER @Nwa;Reply                      ! Read the 1 when complete
160 !
170 ! Trace values are two elements per point, display format dependent
180 DIM Dat(1:11,1:2)                     ! Trace data array
190 !
200 OUTPUT @Nwa;”POIN 11;”                ! Set trace length to 11 points
210 OUTPUT @Nwa;”OPC?;SING;”              ! Single sweep mode and wait
220 ENTER @Nwa;Reply                      ! Read reply
230 !
240 OUTPUT @Nwa;”FORM4;”                  ! Set form 4 ASCII format
250 OUTPUT @Nwa;”OUTPFORM;”               ! Send formatted trace to controller
260 ENTER @Nwa;Dat(*)                     ! Read in data array from analyzer
270 !
280 ! Now to calculate the frequency increments between points
290 OUTPUT @Nwa;”POIN?;”                  ! Read number of points in the trace
300 ENTER @Nwa;Num_points
310 OUTPUT @Nwa;”STAR?;”                  ! Read the start frequency
320 ENTER @Nwa;Startf
330 OUTPUT @Nwa;”SPAN?;”                  ! Read the span
340 ENTER @Nwa;Span
350 !
360 F_inc=Span/(Num_points-1)            ! Calculate fixed frequency increment
370 !
380 PRINT “Point”,”Freq (MHz)”,”  Value 1”,”  Value 2”
390 IMAGE 3D,7X,5D.3D,3X,3D.4D,3X,3D.4D   ! Formatting for controller display
400 !
410 FOR I=1 TO Num_points                 ! Loop through data points
420   Freq=Startf+(I-1)*F_inc             ! Calculate frequency of data point
430   PRINT USING 390;I,Freq/1.E+6,Dat(I,1),Dat(I,2)! Print analyzer data
440 NEXT I
450 !
460 OUTPUT @Nwa;”MARKDISC;”               ! Discrete marker mode
470 OUTPUT @Nwa;”MARK1 3E+4;”             ! Position marker at 30 KHz
480 !
490 OUTPUT @Nwa;”OPC?;WAIT;”              ! Wait for the analyzer to finish
500 ENTER @Nwa;Reply                      ! Read the 1 when complete
510 LOCAL 7                               ! Release HP-IB control
520 !
530 PRINT
540 PRINT “Position the marker with the knob and compare the values”
550 !
560 END
```

Data Transfer Using Floating-Point Numbers

File Name EXAMP3C.BAS

Description This example program illustrates data transfer using Form 3 in which data is transmitted in the floating-point formats. Form 2 is nearly identical except for the IEEE 32-bit format of 4 bytes-per-value. Form 5 reverses the order of the bytes to conform with the PC conventions for defining a real number.

The block-data formats have a four-byte header. The first two bytes are the ASCII characters *#A* that indicate that a fixed-length block transfer follows, and the next two bytes form an integer containing the number of bytes in the block to follow. The header must be read in so that data order is maintained.

This transfer is more than twice as fast than a Form 4 transfer. With the Form 4 transfer, 10,050 bytes are sent (201 points × 2 values-per-point × 24 bytes-per-value). Using Form 2 to transfer the data, only 1612 bytes are sent (201 points × 2 values-per-point × 4 bytes-per-value). Refer to Table 1-5, “HP 8702D/Option 011 Array-Data Formats,” on page 1-19 to compare the formats.

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

- The system is initialized.
- The integer variables are defined to contain the header information.
- The number of points in the trace is set to 11.
- The selected frequency span is swept once.
- Data-transfer format 3 is set.
- The headers are read from the trace.
- The array size is calculated and allocated.
- The trace data is read in and printed on the controller display.
- The marker is activated and placed at 30 kHz.
- The instrument is returned to local control and the program ends.

Running the Program

Run the program. The computer displays the number of elements and bytes associated with the transfer of the trace, as well as the first 10 data points. Position the marker and examine the data values. Compare the received values with the analyzer's marker values.

```
10  ! This program shows how to read in a data trace in IEEE 64 bit
20  ! format. The array header is used to determine the length of the
30  ! array and to allocate the array size.
40  !
50  ! EXAMP3C
60  !
70  CLEAR SCREEN
80  ! Initialize the analyzer
90  ASSIGN @Nwa TO 716                  ! Assign an I/O path for the analyzer
100 ASSIGN @Nwadat TO 716;FORMAT OFF    ! Binary data path definition
110 !
120 ABORT 7                             ! Generate an IFC (Interface Clear)
130 CLEAR @Nwa                          ! SDC (Selected Device Clear)
140 OUTPUT @Nwa;"OPC?;PRES;"            ! Preset the analyzer and wait
150 ENTER @Nwa;Reply                    ! Read the 1 when completed
160 !
170 INTEGER Dheader,Dlength             ! Integer variables for header info
180 Numpoints=11                        ! Number of points in the trace
190 OUTPUT @Nwa;"POIN";Numpoints;";"    ! Set number of points in trace
200 !
210 !  Set up data transfer
220 OUTPUT @Nwa;"OPC?;SING"             ! Single sweep and wait
230 ENTER @Nwa;Reply                    ! Read the 1 when completed
240 !
250 OUTPUT @Nwa;"FORM3;"                ! Select form 3 format
260 OUTPUT @Nwa;"OUTPFORM;"             ! Send formatted output trace
270 !
280 ENTER @Nwadat;Dheader,Dlength       ! Read headers from trace data
290 !
300 ALLOCATE Dat(1:Dlength/16,1:2)      ! Use length to determine array size
310 ENTER @Nwadat;Dat(*)                ! Read in trace data
320 !
330 PRINT "Size of array ";Dlength/16;" elements"
340 PRINT "Number of bytes ";Dlength
350 !
360 ! Print out the data array
370 PRINT "Element","Value 1","    Value 2"
380 IMAGE 3D,6X,3D.4D,6X,3D.4D
390 FOR I=1 TO Numpoints                ! Loop through the data points
400   PRINT USING 380;I,Dat(I,1),Dat(I,2)
410 NEXT I
420 !
430 OUTPUT @Nwa;"MARKDISC;"             ! Discrete marker mode
440 OUTPUT @Nwa;"MARK1 3E+4;"           ! Position marker at 30 KHz
450 !
460 OUTPUT @Nwa;"OPC?;WAIT;"            ! Wait for the analyzer to finish
470 ENTER @Nwa;Reply                    ! Read the 1 when complete
480 LOCAL @Nwa                          ! Release HP-IB control
490 !
500 PRINT
510 PRINT "Position the marker with the knob and compare the values."
520 !
530 END
```

Data Transfer Using Frequency-Array Information

File Name EXAMP3D.BAS

Description This example explains how to use the limit-test array to read the corresponding frequency values for the completed trace array into the controller. The analyzer is set to sweep from 10 MHz to 200 MHz in log-frequency mode with the number of points in the trace set to 11. This makes it very difficult to compute the frequency-point spacing in the trace. The points are equally spaced across the trace, but not equally spaced in relation to frequency (because the frequency span is displayed in a logarithmic scale, as opposed to a linear scale). The limit-test data array may be read from the analyzer to provide the frequency values for each data point. Four values are read for each data point on the analyzer. The test results and limit values are not used in this example, only the frequency values are used. This technique is the only method of obtaining the non-linear frequency data from the analyzer display. The test data and frequencies are printed on the controller display and the marker is enabled to allow the operator to examine the actual locations on the analyzer display.

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

- The system is initialized.
- The integer variables for the header information are defined.
- The number of points in the trace is set to 11.
- The frequency span (10 MHz to 200 MHz) is selected.
- The log-frequency sweep is selected.
- The data-transfer format 3 is set.
- The headers are read from the trace.
- The array size is calculated and allocated.
- The trace data is read in.
- The limit-test array is calculated and allocated.
- The limit-line test array is read in.
- The table header is printed.
- The program cycles through the trace values.
- The trace data and frequency are printed.
- The discrete-marker mode is activated.
- The marker is activated and placed at 10 MHz.
- The instrument is returned to local control and the program ends.

Running the Program

Run the program. Observe the controller display. The corresponding frequency values are shown with the trace-data values. Position the marker and observe the relationship between the frequency values and the point spacing on the trace. Compare the trace-data values on the analyzer with those received by the controller.

```
10  ! This program shows how to read in a trace and create the frequency
20  ! value associated with the trace data value. EXAMP3C is used to
30  ! read in the data from the analyzer. The start and stop
40  ! frequencies are set to provide two decades of log range. Log sweep
50  ! is set and the frequency data points are read from the limit test
60  ! array and displayed with the data points.
70  !
80  ! EXAMP3D
90  !
100 ASSIGN @Nwa TO 716                   ! Assign an I/O path for the analyzer
110 ASSIGN @Nwadat TO 716;FORMAT OFF     ! Binary path for data transfer
120 !
130 CLEAR SCREEN
140 ! Initialize the analyzer
150 ABORT 7                              ! Generate an IFC (Interface Clear)
160 CLEAR @Nwa                           ! SDC (Selective Device Clear)
170 OUTPUT @Nwa;"OPC?;PRES;"             ! Preset the analyzer and wait
180 ENTER @Nwa;Reply                     ! Read the 1 when completed
190 !
200 INTEGER Dheader,Dlength              ! Integer variables for header info
210 !
220 OUTPUT @Nwa;"POIN 11;"               ! Set trace length to 11 points
230 OUTPUT @Nwa;"STAR 10.E+6;"           ! Start frequency 10 MHz
240 OUTPUT @Nwa;"STOP 200.E+6;"          ! Stop frequency 200 MHz
250 OUTPUT @Nwa;"LOGFREQ;"               ! Set log frequency sweep
260 !
270 !  Set up data transfer
280 OUTPUT @Nwa;"OPC?;SING"              ! Single sweep and wait
290 ENTER @Nwa;Reply                     ! Read the 1 when completed
300 !
310 OUTPUT @Nwa;"FORM3;"                 ! Select form 3 trace format
320 OUTPUT @Nwa;"OUTPFORM;"              ! Output formatted trace
330 !
340 ENTER @Nwadat;Dheader,Dlength        ! Read headers from trace data
350 !
360 ALLOCATE Dat(1:Dlength/16,1:2)       ! Use length to determine array size
370 ENTER @Nwadat;Dat(*)                 ! Read in trace data
380 !
390 ! Create the corresponding frequency values for the array
400 !
410 ! Read the frequency values using the limit test array
420 ALLOCATE Freq(1:Dlength/16,1:4)      ! Limit line results array
430 ! Limit line values are frequency, test results, upper and lower limits
440 !
450 OUTPUT @Nwa;"OUTPLIML;"              ! Request limit line test results
460 ENTER @Nwa;Freq(*)                   ! Read 4 values per point
470 !
480 ! Display table of freq and data
490 !
500 PRINT " Freq (MHz)","Mag (dB)"       ! Print table header
510 FOR I=1 TO 11                        ! Cycle through the trace values
520   Freqm=Freq(I,1)/1.E+6              ! Convert frequency to MHz
530   PRINT USING "4D.6D,9X,3D.4D";Freqm,Dat(I,1)  ! Print trace data
540 NEXT I
550 !
560 ! Set up marker to examine frequency values
```

```
570 OUTPUT @Nwa;”MARKDISC;”             ! Discrete marker mode
580 OUTPUT @Nwa;”MARK1 10.E+6;”         ! Turn on marker and place at 10 MHz
590 !
600 OUTPUT @Nwa;”OPC?;WAIT;”            ! Wait for the analyzer to finish
610 ENTER @Nwa;Reply                    ! Read the 1 when complete
620 LOCAL @Nwa                          ! Release HP-IB control
630 PRINT                               ! Blank line
640 PRINT “Position marker and observe frequency point spacing”
650 !
660 END
```

Data Transfer Using Internal Binary Format

File Name EXAMP3E.BAS

Description Form 1 is used for rapid I/O transfer of analyzer data. It contains the least number of bytes-per-trace and does not require reformatting in the analyzer. This format is more difficult to convert into a numeric array in the controller. Analyzer-state information, such as learn strings and calibration arrays, may be easily transferred in this format because data conversion is not required. Recalling an instrument state that has been stored in a file and transferring instrument-state information to the analyzer are excellent applications of a Form 1 data transfer.

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

- The system is initialized.
- The integer variables for the header information are defined.
- The string variable for the header is defined.
- The selected frequency span is swept once.
- The internal-binary format is selected.
- The error-corrected data is output from the analyzer.
- The two data-header characters and the two length bytes are read in.
- The string buffer is allocated for data.
- The trace data is read into the string buffer.
- The analyzer is restored to continuous-sweep mode and queried for command completion.
- The instrument is returned to local control and the program ends.

Running the Program

Run the program. The analyzer is initialized. The header and the number of bytes in the block transfer are printed on the controller display. Once the transfer is complete, the number of bytes in the data string is printed. Compare the two numbers to be sure that the transfer was completed.

```
10  ! This program is an example of a form 1, internal format data
20  ! transfer. The data is stored in a string dimensioned to the
30  ! length of the data being transferred.
40  !
50  ! EXAMP3E
60  !
70  ASSIGN @Nwa TO 716                   ! Assign an I/O path for the analyzer
80  ASSIGN @Nwa_bin TO 716;FORMAT OFF    ! Binary path for data transfer
90  !
100 CLEAR SCREEN
110 ! Initialize the analyzer
120 ABORT 7                              ! Send IFC (Interface Clear)
130 CLEAR @Nwa                           ! SDC (Selective Device Clear)
140 OUTPUT @Nwa;”OPC?;PRES;”             ! Preset the analyzer and wait
150 ENTER @Nwa;Reply                     ! Read the 1 when completed
160 !
170 INTEGER Length                       ! Header length 2 bytes
180 DIM Header$[2]                       ! Header string 2 bytes
190 !
200 OUTPUT @Nwa;”OPC?;SING;”             ! Single sweep and wait
210 ENTER @Nwa;Reply                     ! Read the 1 when completed
220 !
230 OUTPUT @Nwa;”FORM1;”                 ! Select internal binary format
240 OUTPUT @Nwa;”OUTPDATA;”              ! Output error corrected data
250 !
260 ! Read in the data header two characters and two bytes for length
270 ! “#,2A”
280 !      # no early termination, terminate when ENTER is complete
290 !      2A read two chars
300 !
310 ENTER @Nwa_bin USING “#,2A”;Header$  ! Read header as 2 byte string
320 ENTER @Nwa_bin;Length                ! Read length as 2 byte integer
330 PRINT “Header “;Header$,”Array length”;Length
340 !
350 ALLOCATE Data$[Length]               ! String buffer for data bytes
360 ! “+,-K” format statement
370 ! + EOI as a terminator LF is suppressed and read as data
380 ! -K All characters are read and not interpreted LF is included
390 ENTER @Nwa_bin USING “+,-K”;Data$    ! Read trace into string array
400 !
410 PRINT “Number of bytes received “;LEN(Data$)
420 !
430 OUTPUT @Nwa;”CONT;”                  ! Restore continuous sweep
440 OUTPUT @Nwa;”OPC?;WAIT;”             ! Wait for the analyzer to finish
450 ENTER @Nwa;Reply                     ! Read the 1 when complete
460 !
470 LOCAL @Nwa                           ! Release HP-IB control
480 END
```

Using Error Queue

File Name EXAMP4A.BAS

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

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

- The system is initialized.
- The error-message string is allocated.
- The analyzer is released from remote control.
- The program begins an endless loop to read the error queue.
- The status byte is read with a serial poll.
- The program tests to see if an error is present in the queue.
- The error-queue bit is set.
- The program requests the contents of the error queue.
- The error number and string are read.
- The error messages are printed until there are no more errors in the queue.
- The instrument is returned to local control.
- The controller emits a beep to attract the attention of the operator and resumes searching for errors.

Running the Program

Run the program. The analyzer goes through the preset cycle. Nothing will happen at first. The program is waiting for an error condition to activate the error queue. To cause an error, press a blank softkey. The message `CAUTION: INVALID KEY` will appear on the analyzer’s display. The computer will beep and print two error messages. The first line will be the invalid key error message, and the second line will be the `NO ERRORS` message. To clear the error queue, you can either loop until the `NO ERRORS` message is received, or until the bit in the status register is cleared. In this case, we wait until the status bit in the status register is clear. Note that while the program is running, the analyzer remains in the local mode and the front-panel keys may be accessed.

The error queue will hold up to 20 errors until all the errors are read out or the instrument is preset. It is important to clear the error queue whenever errors are detected. Otherwise, old errors may be mistakenly associated with the current instrument state.

Press SYSTEM and then the unlabeled key several times quickly and watch the display. The number of errors observed should correspond to the number of times you pressed the key.

As another example, press CAL, then the *CALIBRATE MENU* key. Select the *RESPONSE* calibration. Press *DONE: RESPONSE* without performing any calibrations. Note the error message on the analyzer and on the controller display. Push the *THRU* key and then *DONE: RESPONSE*. We are not concerned with the validity of the calibration, just setting a simple calibration on the analyzer. Note that COR is displayed in the upper left-hand section of the graticule. Now, press *START* and ↑. This will generate an error because the start frequency has been changed, invalidating the calibration. This error is reported on the controller display as well. A complete list of error messages and their descriptions can be found in "Error Messages" in the *HP 8702D Reference* manual.

The program is in an infinite loop waiting for errors to occur. End the program by pressing RESET or BREAK on the controller keyboard.

NOTE

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

```
10  ! This program is an example of using the error queue to detect
20  ! errors generated by the analyzer. The status byte is read and
30  ! bit 3 is tested to determine if an error exists. The error queue
40  ! is printed out and emptied.
50  !
60  ! EXAMP4A
70  !
80  ASSIGN @Nwa TO 716                   ! Assign an I/O path for the analyzer
90  !
100 CLEAR SCREEN
110 ! Initialize the analyzer
120 ABORT 7                              ! Generate an IFC (Interface Clear)
130 CLEAR @Nwa                           ! SDC (Selective Device Clear)
140 OUTPUT @Nwa;”OPC?;PRES;”             ! Preset the analyzer and wait
150 ENTER @Nwa;Reply                     ! Read the 1 when complete
160 !
170 DIM Error$[50]                       ! String for analyzer error message
180 !
190 LOCAL @Nwa                           ! Release analyzer from remote control
200 !
210 LOOP                                 ! Endless loop to read error queue
220   REPEAT
230     Stat=SPOLL(@Nwa)                 ! Read status byte with serial poll
240   UNTIL BIT(Stat,3)                  ! Test for error queue present
250   !
260 ! Error queue bit is set
270   REPEAT                             ! Loop until error number is 0
280     OUTPUT @Nwa;”OUTPERRO;”          ! Request error queue contents
290     ENTER @Nwa;Err,Error$            ! Read error number and string
300     PRINT Err,Error$                 ! Print error messages
310   UNTIL Err=0                        ! No more errors in queue
320 !
330   LOCAL @Nwa                         ! Release analyzer from remote
340   BEEP 600,.2                        ! Beep to attract attention
350 END LOOP                             ! Repeat error search
360 !
370 END
```

Generating Interrupts

File Name EXAMP4B.BAS

Description Interrupts can be generated using the status-reporting mechanism. The status-byte bits can be enabled to generate a service request (SRQ) when set. In turn, the instrument controller can be set up to generate an interrupt on the SRQ and respond to the condition which caused the SRQ.

To generate an SRQ, a bit in the status byte is enabled using the `SREn;` command. A one (1) in a bit position enables that bit in the status byte. Hence, executing `SRE 8;` enables an SRQ on bit 3, the check-error queue, since the decimal value 8 equals 00001000 in binary representation. Whenever an error is put into the error queue and bit 3 is set, the SRQ line is asserted, illuminating the (S) indicator in the HP-IB status block on the front panel of the analyzer. The only way to clear the SRQ is to disable bit 3, re-enable bit 3, or read out all the errors from the queue.

A bit in the event-status register can be enabled so that it is summarized by bit 5 of the status byte. If any enabled bit in the event-status register is set, bit 5 of the status byte will also be set. For example, executing `ESE 66;` enables bits 1 and 6 of the event-status register, since in binary, the decimal number 66 equals 01000010. Hence, whenever active control is requested or a front-panel key is pressed, bit 5 of the status byte will be set. Similarly, executing `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, executing `ESE 32;SRE 32;` enables the syntax-error bit. When the syntax-error bit is set, the summary bit in the status byte will be set. This will, in turn, enable an SRQ on bit 5 of the status byte, the summary bit for the event-status register.

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

- The system is initialized.
- The status registers are cleared.
- The event-status register bit 5 is enabled.
- The status-register bit 5 is enabled.
- The interrupt pointer is enabled and points to a subroutine.
- A bad command is sent to the analyzer to generate errors.
- The controller reads a serial-poll byte from HP-IB in the event of an interrupt.
- The program tests for an SRQ.
- If the SRQ is not generated by the analyzer, the subroutine stops and displays, `SRQ FROM OTHER DEVICE`.
- If the SRQ was generated by the analyzer, the program reads the status byte and event-status register.
- If bit 5 in the event-status register is set, program prints, `SYNTAX ERROR FROM ANALYZER`.
- If bit 5 in the event-status register is NOT set, program prints, `SYNTAX ERROR BIT NOT SET.`
- The SRQ interrupt is re-enabled on the bus.
- At the finish, the interrupt is deactivated.
- The analyzer is released from remote control and the program ends.

Running the Program

Run the program. The computer will preset the analyzer, then pause for a second or two. After pausing, the program sends an invalid command string, `STIP 2GHZ;`, to cause a syntax error. This command is intended to be, `STOP 2 GHZ;`. The computer will display a series of messages from the SRQ-handler routine. The analyzer will display `CAUTION: SYNTAX ERROR` and the incorrect command, pointing to the first character it did not understand.

The SRQ can be cleared by reading the event-status register and clearing the latched bit, or by clearing the enable registers with the `CLES;` command. The syntax-error message on the analyzer display can only be cleared by the HP-IB Device Clear (DCL) message or Selected Device Clear (SDC) message. Device Clear is not commonly used because it clears every device on the bus. Selected Device Clear can be used to reset the input and output queue and the registers of a specific instrument on the bus. This will also clear all the interrupt definitions.

```
10  ! This program is an example of using an SRQ based interrupt to
20  ! detect an error condition in the analyzer. In this example, a
30  ! syntax error is generated with an invalid command. The status byte
40  ! is read in and tested. The error queue is read, printed out and
50  ! then cleared.
60  !
70  ! EXAMP4B
80  !
90  ASSIGN @Nwa TO 716                   ! Assign an I/O path for the analyzer
100 !
110 CLEAR SCREEN
120 ! Initialize the analyzer
130 ABORT 7                              ! Generate an IFC (Interface Clear)
140 CLEAR @Nwa                           ! SDC (Selective Device Clear)
150 OUTPUT @Nwa;”OPC?;PRES;”             ! Preset the analyzer and wait
160 ENTER @Nwa;Reply                     ! Read the one from the analyzer
170 !
180 DIM Error$[50]                       ! String for analyzer error message
190 ! Set up syntax error interrupt
200 OUTPUT @Nwa;”CLES;”                  ! Clear the status registers
210 !
220 ! Generate SRQ when bit 5 is set
230 OUTPUT @Nwa;”ESE 32;”                ! Event status register bit 5 enabled
240 !
250 ! Generate bit 5 in status register when syntax error occurs
260 OUTPUT @Nwa;”SRE 32;”                ! Status register bit 5 enabled
270 !
280 ! Setup the interrupt pointer to a subroutine
290 ON INTR 7 GOSUB Srq_det              ! When interrupt occurs go to Srq_det
300 Stat=SPOLL(@Nwa)                     ! Clear any pending SRQs
310 ENABLE INTR 7;2                      ! Set interrupt on HP-IB bit 2 (SRQ)
320 !
330 DISP “Waiting for bad syntax”
340 WAIT 2                               ! Pause for 2 seconds
350 !
360 OUTPUT @Nwa;”STIP 2GHZ;;”            ! Send bad STOP command syntax
370 !
380 WAIT 2                               ! Pause for 2 seconds
390 DISP ““                              ! Clear display line
400 GOTO Finish                          ! Exit program example
410 !
420 !*************************** Subroutines ******************************
430 !
440 Srq_det:                             ! SRQ handler
450 Stat=SPOLL(@Nwa)                     ! Read serial poll byte from HP-IB
460 PRINT “Stat from Serial Poll”;Stat
470 IF BIT(Stat,6) THEN                  ! Test for SRQ
480   PRINT “SRQ received from analyzer”
490 ELSE                                 ! No SRQ from analyzer
500   PRINT “SRQ from other device”
510   STOP                               ! Stop if not from analyzer
520 END IF
530 !
540 IF BIT(Stat,5) THEN                  ! Event status register bit set
550   PRINT “Event Status Register caused SRQ”
560 ELSE                                 ! Some other bit set
```

```
570   PRINT “Some other bit caused the SRQ”
580   STOP                                ! Stop if bit not set
590 END IF
600   !
610 REPEAT
620   OUTPUT @Nwa;”OUTPERRO;”             ! Read analyzer error queue
630   ENTER @Nwa;Err,Error$               ! Read error number and string
640   PRINT Err,Error$                    ! Print error message
650 UNTIL Err=0                           ! No more errors in queue
660 !
670 PRINT                                 ! White space
680 ENABLE INTR 7;2                       ! Re-enable SRQ interrupt on HP-IB
690 RETURN
700 !
710 !************************** End Subroutines ******************************
720 !
730 Finish:                               ! End of program and exit
740 DISP “Finished”
750 OFF INTR 7                           ! Turn off interrupt
760 LOCAL @Nwa                            ! Release HP-IB control
770 END
```

Power Meter Calibration

File Name EXAMP4C.BAS

Description For increased accuracy of the analyzer's PORT 1-output power, a power meter calibration is available. This measurement-accuracy enhancement technique is described in the "Error Messages" section of the *HP 8702D Reference* manual. The example described will perform the sample and sweep calibration under HP-IB remote control.

The power meter is usually connected to PORT 1 for the forward measurements. Its address must be set correctly and it must be connected to the HP-IB. The power meter address can be set by pressing: LOCAL, *SET ADDRESSES*, *ADDRESS P MTR/HPIB* and using the ↑ and ↓ keys or the numeric key pad to complete the process. The appropriate command must be selected for the model number of the power meter being used. Press *POWER MTR: [ ]* until the model being used is displayed between the brackets. The correction factors for the power sensor are entered into the analyzer. All of these steps are explained in the "Error Messages" section of the *HP 8702D Reference* manual.

The number of readings-per-point must also be selected before starting. The number of points directly affects the measurement time of the calibration sequence. The power meter must interact with the analyzer for each of the selected points and read the number of values specified for each trace point. Typically, two readings-per-point is considered appropriate. More than two readings-per-point could lead to unacceptable processing time.

To control a power meter calibration via HP-IB, the analyzer must be set to pass-control mode. The analyzer must position the local oscillator to a point in the sweep and read the power present at the power meter sensor. For this operation to take place, the system controller must set up the measurement and then pass control to the analyzer to read each data point in the sweep. After reading the data point from the power meter, the analyzer passes control back to the system controller. The analyzer then sets up to measure the next point and again requests control from the system controller. This process continues until the analyzer signals that the entire sweep has been measured point-by-point.

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

- The system is initialized.
- The number of points in the trace is set.
- The number of readings-per-point is set.
- The frequency span is set.
- The reference channel is measured.
- The power meter-calibration array is allocated.
- The power meter model is chosen.
- The status registers are cleared.
- The request-control summary bit is enabled.
- The pass-control mode is enabled.
- A calibration sweep is taken to begin the sequence.
- The status byte is read until control is requested.
- The computer passes control to the analyzer.
- The display is cleared and the analyzer is set to talker/listener mode.
- The HP-IB interface status is read until control is returned.
- The program loops until all the points have been measured.
- The power meter calibration is enabled.
- The calibration data is output to the controller in Form 4, ASCII format.
- The power meter-calibration factors are read into the controller.
- The analyzer is released from remote control and the program ends.

Running the Program

The analyzer is preset and the power meter-calibration routine begins. The analyzer displays the message, `WAITING FOR HP-IB CONTROL,` when it is requesting control. The system controller display prints, `Passing Control,` when control is passed to the analyzer. The controller displays, `Waiting for request,` while the analyzer has control and is reading the power meter.

The interaction of the messages and the movement of the cursor allow observation of the calibration process. Once the calibration is complete, the analyzer displays, `POWER METER CAL IS COMPLETE,` and the system controller displays, `Finished with Power meter Cal`.

The power meter-calibration mode (with one sweep of correction data) is enabled and the calibration is switched ON. At the completion of the program, talker/listener mode is restored, the event-status registers are cleared (to halt the status-byte interaction), the power meter correction factors are displayed, the sweep is placed in continuous-sweep mode, the analyzer is released from HP-IB control, and the program ends.

```
10  ! This routine does a power meter cal using pass control.
20  ! A measurement cycle takes place on each point of the trace. The
30  ! point is measured by the power meter and the measured value read
40  ! into the analyzer. The command TAKCS; arms this measurement mode.
50  ! The number of measurements is determined by the number of points in
60  ! the trace, the number of readings per point and an extra measurement
70  ! cycle to release the power meter.
80  ! Control is passed to the analyzer, the point is measured and
90  ! the data is transferred to the analyzer. Control is passed back to
100 ! the controller and the cycle begins again. Serial poll is used to
110 ! read the status byte of the analyzer and test the logic.
120 ! The HP-IB interface status register is monitored to determine when
130 ! control is returned to the interface from the analyzer.
140 !
150 ! EXAMP4C
160 !
170 ASSIGN @Nwa TO 716                    ! Assign an I/O path for the analyzer
180 !
190 CLEAR SCREEN
200 ! Initialize the analyzer
210 ABORT 7                               ! Generate an IFC (Interface Clear)
220 CLEAR @Nwa                            ! SDC (Selective Device Clear)
230 OUTPUT @Nwa;”OPC?;PRES;”              ! Preset the analyzer and wait
240 ENTER @Nwa;Reply                      ! Read the 1 when complete
250 !
260 INTEGER Stat
270 !
280 ! Set up the analyzer parameters
290 Numpoints=11                          ! Number of points in the trace
300 Numreads=2                            ! Number of readings per point
310 Startf=1.00E+8                        ! Start frequency
320 Stopf=5.0E+8                          ! Stop frequency
330 !
340 OUTPUT @Nwa;”POIN”;Numpoints;”;”      ! Set trace length to numpoints
350 OUTPUT @Nwa;”NUMR”;Numreads;”;”       ! Set number of readings per point
360 OUTPUT @Nwa;”STAR”;Startf            ! Set start frequency
370 OUTPUT @Nwa;”STOP”;Stopf              ! Set stop frequency
380 OUTPUT @Nwa;”MEASR;”                  ! Measure the reference channel
390 !
400 ALLOCATE Pmcal(1:Numpoints)           ! Create power meter cal array
410 !
420 ! Store the original trace for comparison
430 OUTPUT @Nwa;”DATI;”
440 OUTPUT @Nwa;”DISPDATM;”
450 OUTPUT @Nwa;”AUTO;”
460 !
470 ! Select the power meter being used for cal
480 ! OUTPUT @Nwa;”POWM ON;”                ! Select 436A power meter
490 OUTPUT @Nwa;”POWMOFF;DEBUON;”         ! Select 437B/438A power meter
500 !
510 ! Set analyzer HP-IB, status regs to interrupt on pass control
520 OUTPUT @Nwa;”CLES;”                   ! Clear status registers
530 OUTPUT @Nwa;”ESE2;”                   ! Enable request control summary bit
540 OUTPUT @Nwa;”SRE32;”                  ! SRQ on events status register
550 !
560 PRINT “Beginning Power Meter CAL”
```

```
570 OUTPUT @Nwa;"USEPASC;"               ! Enable pass control operation
580 OUTPUT @Nwa;"TAKCS;"                ! Take Cal Sweep
590 !
600 FOR I=1 TO Numpoints*Numreads+1      ! Points * Number of readings plus 1
610   ! Serial poll does not place analyzer in remote operation
620   ! and does not require the analyzer to process the command.
630   !
640   REPEAT                             ! Repeat until SRQ detected
650     Stat=SPOLL(@Nwa)                 ! Serial poll to read status byte
660     DISP "Stat ";Stat;" Waiting for request"
670   UNTIL BIT(Stat,6)                  ! SRQ detected for request control
680   OUTPUT @Nwa;"ESR?;"                ! Read status register to clear
690   ENTER @Nwa;Reply                  ! Read and discard register value
700   !
710   PRINT "Passing Control"            ! status read and passing control
720   PASS CONTROL @Nwa                  ! Pass control to analyzer
730   !
740   REPEAT
750   ! Read HP-IB interface state information register.
760     STATUS 7,6;Hpib                  ! Test HP-IB register for control
770   !
780   ! Reading the interface status register does not interact with the
790   ! analyzer. Bit 6 is set when control is returned.
800   !
810     DISP "Waiting for control"
820   UNTIL BIT(Hpib,6)                  ! Loop until control is returned
830 NEXT I
840 !
850 PRINT "Finished with Power meter Cal"
860 DISP ""                              ! Clear display message
870 !
880 OUTPUT @Nwa;"TALKLIST;"              ! Restore Talker/Listener operation
890 OUTPUT @Nwa;"CLES;"                  ! Clear and reset status byte operation
900 !
910 OUTPUT @Nwa;"PWMCONES;"              ! Power meter cal correct one sweep
920 OUTPUT @Nwa;"OPC?;WAIT;"             ! Wait for the analyzer to finish
930 ENTER @Nwa;Reply                     ! Read the 1 when complete
940 !
950 ! Read the power meter cal correction factors
960 OUTPUT @Nwa;"FORM4;"                 ! ASCII data format to read cal data
970 OUTPUT @Nwa;"OUTPPMCAL1;"            ! Request the power meter cal factors
980 ENTER @Nwa;Pmcal(*)                 ! Read the factors
990 !
1000! Display the power meter cal factors
1010 PRINT "Point","Factor"
1020 FOR I=1 TO Numpoints                ! Cycle throught the factors
1030 PRINT I,Pmcal(I)
1040 NEXT I
1050!
1060 LOCAL @Nwa                          ! Release HP-IB control
1070 END
```

Using the Learn String

File Name EXAMP5A.BAS

Description This section provides several different examples of performing analyzer system setups.

NOTE

The most efficient option for storing and recalling analyzer states is using the analyzer's internal registers to save the CAL data. Recalling these registers is the fastest solution to restoring analyzer setups. Refer to "Chapter 8, "Saving Data, States, and the Display" in the *HP 8702D User's Guide* for detailed information on the analyzer's internal storage registers. In the event that all the registers have been used, the internal disk drive is not used, or if internal memory limitations exist, then these external solutions become viable.

The purpose of this example is to demonstrate several programming options for storing and recalling entire instrument states over HP-IB. The examples describe two different processes for storing and recalling instrument states. The first example accomplishes the task using the learn string. The second example involves reading both the learn string and the calibration arrays out of the analyzer and storing them to disk or storing them in the system controller itself.

Using the learn string is a very rapid way of saving the instrument state, but using direct disk access has the advantage of automatically storing calibrations, cal kits, and data along with the instrument state.

A complete analyzer setup requires sending the learn string and a calibration array to set the analyzer parameters. The CAL array may also be placed in the analyzer, just as if a calibration was performed. By sending both sets of data, the analyzer may be quickly setup for a measurement.

Several different measurements may be required in the course of testing a device. An efficient way of performing multiple measurements is to send both the calibration array and the learn string, and then perform the measurements.

Example

The learn string is a very fast and easy way to read an instrument state. The learn string includes all front-panel settings, the limit table for each channel, and the list-frequency table. It can be read out of the analyzer with the command `OUTPLEAS`, and input to the analyzer with the command `INPULEAS`. This array is always transmitted in Form 1, the internal format for the analyzer. It cannot be longer than 3000 bytes. The example for a Form 1 transfer could also have been used. However, the "Using the Learn String" example is the simplest solution for reading the learn string from the analyzer.

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

- The system is initialized.
- The string storage is allocated.
- The learn string is requested.
- The string is read without any processing.
- The analyzer is released from remote control.
- The instrument state is changed by the operator.
- The learn string is sent back to the analyzer.
- The analyzer is released from remote control and the program ends.

Running the Program

Run the program. When the program stops, change the instrument state and press ENTER on the controller. The analyzer will be returned to its original state by sending the learn string to the analyzer.

```
10  ! This program shows how to retrieve a learn string from the analyzer
20  ! into a string array. The state of the analyzer is then changed and the
30  ! learn string reloaded to return the analyzer to the previous settings.
40  !
50  ! EXAMP5A
60  !
70  ASSIGN @Nwa TO 716                    ! Assign an I/O path for the analyzer
80  !
90  CLEAR SCREEN
100 ! Initialize the analyzer
110 ABORT 7                               ! Generate an IFC (Interface Clear)
120 CLEAR @Nwa                            ! SDC (Selected Device Clear)
130 !
140 DIM State$[3000]                      ! Define a string for contents
150 !
160 OUTPUT @Nwa;”OUTPLEAS;”               ! Output the learn string
170 ENTER @Nwa USING “+,-K”;State$        ! Read the string with no processing
180              ! + Terminate on EOI only
190              ! -K ignore LF as terminator treat as data
200              !
210 LOCAL @Nwa                            ! Release HP-IB control
220 !
230 INPUT “Change state and press ENTER”,A$
240 !
250 OUTPUT @Nwa;”INPULEAS;”;State$;       ! Send the learn string to analyzer
260 DISP “Analyzer state has been restored!”
270 !
280 OUTPUT @Nwa;”OPC?;WAIT;”              ! Wait for the analyzer to finish
290 ENTER @Nwa;Reply                      ! Read the 1 when complete
300 LOCAL @Nwa                            ! Release HP-IB control
310 END
```

Reading Calibration Data

File Name EXAMP5B.BAS

Description This example demonstrates:

- how to read measurement calibration data out of the analyzer
- how to read it back into the analyzer
- how to determine which calibration is active

The data used to perform measurement-error correction is stored inside the analyzer in one (or more) of twelve calibration-coefficient arrays. Each array is a specific error coefficient, and is stored and transmitted as an error-corrected data array. Each point is a real/imaginary pair, and the number of points in the array is the same as the number of points in the trace. The five array-data formats also apply to the transfer of calibration-coefficient arrays. "Printing, Plotting, or Saving Measurement Results" in the *HP 8702D User's Guide* contains information on the storage locations for calibration coefficients and different calibration types.

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

A computer can also store calibration coefficients in the analyzer. To do this, declare the type of calibration data about to be stored in the analyzer just as if you were about to perform that calibration. Then, instead of calling up different classes, transfer the calibration coefficients using the `INPUCALC`*`nn;`* commands. The variables *`nn`* are a data pair appended to the command representing a calibration number from *`01`* through *`12`*. When all the coefficients are stored in the analyzer, activate the calibration by issuing the mnemonic `SAVC;`, and trigger a sweep on the analyzer.

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

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

- The system is initialized.
- The calibration types and number of arrays are defined.
- The integer variables for reading the headers are defined.
- The calibration type and number of arrays are read by the controller.
- The output is formatted in Form 3.
- The number of points in the trace is read.
- The memory is allocated for the calibration arrays.
- Each calibration array is requested from the analyzer.
- The elements from each calibration array are read in.
- The next calibration array is requested until all the arrays have been read.
- The calibration type is sent to the analyzer.
- Each calibration array is sent.
- The calibration is activated.
- The analyzer is released from remote control and the program ends.

Running the Program

Before executing the program, perform a calibration.

The program is able to detect which type of calibration is active. With that information, it predicts how many arrays to read out. When all the arrays have been sent to the computer, the program prompts the operator. The operator then switches OFF the calibration or performs a completely different calibration on the analyzer and continues the program. The computer reloads the old calibration. The operator should not preset the analyzer because the instrument settings must be the same as those that were present when the calibration was taken.

NOTE

The retransmitted calibration is associated with the current instrument state; the instrument has no way of knowing the original state associated with the calibration data. For this reason, it is recommended that the learn string be used to store the instrument state whenever calibration data is stored. The next example demonstrates how to reload the analyzer state with both the learn string and the calibration arrays.

```
10  ! This program shows how to manipulate calibration data from the analyzer.
20  ! It demonstrates how to read calibration data from the analyzer, and
30  ! how to replace it. The type of calibration active is determined and
40  ! the program reads in the correct number of arrays. The number of points
50  ! in the trace, and in the cal array, is determined and used to dimension
60  ! storage arrays.
70  !
80  ! EXAMP5B
90  !
100 ASSIGN @Nwa TO 716                    ! Assign an I/O path for the analyzer
110 ASSIGN @Nwa_bin TO 716;FORMAT OFF     ! Assign binary path
120 !
130 CLEAR SCREEN
140 ! Initialize the analyzer
150 ABORT 7                               ! Generate an IFC (Interface Clear)
160 CLEAR @Nwa                            ! SDC (Selected Device Clear)
170 !
180 ! Data for determining CAL type and number of arrays
190 DATA “CALIRESP”,1,”CALIRAI”,2,”CALIS111”,3
200 DATA “CALIS221”,3,”CALIFUL2”,12
210 DATA “NOOP”,0
220 !
230 INTEGER Hdr,Lgth,I,J                  ! Integers for reading headers
240 !
250 READ Calt$,Numb                       ! Read CAL type from data statement
260 IF Numb=0 THEN GOTO 690               ! If no CAL type is present Exit
270 OUTPUT @Nwa;Calt$;”?;”                ! Query if CAL type is active
280 ENTER @Nwa;Active                     ! Read 1 if active
290 IF NOT Active THEN GOTO 250           ! Load another CAL type and re-try
300 !
310 PRINT Calt$,Numb                      ! Active CAL and number of arrays
320 !
330 OUTPUT @Nwa;”FORM3;”                  ! Form 3 IEEE 64 bit floating point
340 OUTPUT @Nwa;”POIN?;”                  ! Request trace length
350 ENTER @Nwa;Poin                       ! Read number of points
360 ALLOCATE Cal(1:Numb,1:Poin,1:2)       ! Arrays for CAL arrays
370 !         Number of arrays, number of points real and imag value per point
380 !
390 FOR I=1 TO Numb                           ! Read arrays
400   OUTPUT @Nwa USING “K,ZZ”;”OUTPCALC”,I   ! Format I to add 0 in command
410   ENTER @Nwa_bin;Hdr,Lgth                 ! Read header & length from array
420   FOR J=1 TO Poin                         ! Read elements for CAL array
430     ENTER @Nwa_bin;Cal(I,J,1),Cal(I,J,2)  ! Read real & imag pair elements
440   NEXT J                                  ! Next location in array
450 NEXT I                                    ! Next CAL array
460 !
470 ! All CAL arrays have been read
480 !
490 INPUT “PRESS RETURN TO RE-TRANSMIT CALIBRATION”,Dum$
500 !
510 OUTPUT @Nwa;”FORM3;”                          ! Use same format as read
520 OUTPUT @Nwa;Calt$;”;”                         ! Send CAL type to analyzer
530 !
540 FOR I=1 TO Numb                               ! Send each array in CAL
550   DISP “TRANSMITTING ARRAY: “,I               ! Show array number
560   OUTPUT @Nwa USING “K,ZZ”;”INPUCALC”,I       ! Send array number 0 format
```

```
570   OUTPUT @Nwa_bin;Hdr,Lgth                    ! Send header & array length
580   FOR J=1 TO Poin                             ! Send each array element
590     OUTPUT @Nwa_bin;Cal(I,J,1),Cal(I,J,2)     ! Real and Imag pair
600   NEXT J                                      ! Next element in array
610 NEXT I                                        ! Next array
620 !
630 OUTPUT @Nwa;”SAVC;”                           ! Activate CAL
640 !
650 OUTPUT @Nwa;”CONT;”                           ! Restore continuous sweep
660 OUTPUT @Nwa;”OPC?;WAIT;”                      ! Wait for analyzer to finish
670 ENTER @Nwa;Reply                              ! Read the 1 when complete
680 !
690 DISP “Finished with CAL transfer”
700 LOCAL @Nwa                                    ! Release HP-IB control
710 END
```

Using Instrument States

File Name EXAMP5C.BAS

Description

NOTE

The instrument state may also be stored in the analyzer's internal registers. This is the fastest and most efficient method for toggling between instrument states. This example is for use when the analyzer's internal memory is full, or when there are other internal-memory limitations.

This example demonstrates using both the learn string and the calibration arrays to completely re-program the analyzer state. If you were performing two entirely different measurements on a device and wanted to quickly change between instrument states and perform the measurements, this example program is a potential solution.

The example will request the learn string and a calibration array from the analyzer and store them in a disk file on the system controller. Once the storage is complete, the operator will be prompted to change the state of the analyzer and then reload the state that was previously stored in the disk file. Once the file is created on the disk, the state information can be retrieved from the controller and restored on the analyzer.

NOTE

The disk file can only be created once. Errors will occur if the operator repeatedly tries to recreate the file.

For this example, only a thru response calibration will be performed and transferred. This means only one calibration array will be read from the analyzer and written to the disk file with the instrument state. To work with more elaborate calibrations, additional arrays will need to be defined and transferred to the disk file. This is not difficult, but requires some additional programming steps which were omitted in the interest of presenting a simple example.

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

- The integers for reading the headers are defined.
- The system is initialized.
- An array is created to hold the learn string.
- The learn string is requested by the controller.
- The number of points in the trace is read.
- The controller allocates an array for the calibration data.
- The calibration data is read into the controller.
- The controller creates and assigns a data file for the calibration array and the learn string.
- The learn string and calibration array are stored in the disk file.
- The operator presses ENTER on the controller to read the calibration data back into the analyzer.
- The learn string is read from the disk file and output to the analyzer.
- The calibration array is read in from the disk file and stored in the analyzer.
- The analyzer is returned to continuous-sweep mode.
- The analyzer is released from remote control and the program ends.

Running the Program

Setup the analyzer and perform a through calibration.

Run the program. The program prompts the operator to change the state of the analyzer and then press ENTER to continue. At this point, the analyzer state is stored on the disk file in the controller. Pressing ENTER will begin the transfer from the disk file to internal arrays within the controller and then on to the analyzer. Once completed, the original state will be restored, the analyzer will be sweeping, the analyzer will be calibrated, and COR will be displayed on the analyzer's display.

```
10  ! This program reads an instrument state and stores it in a disk file.
20  ! The learn string and CAL array are both read into the controller and
30  ! then transferred to a disk file for storage. The file contents are
40  ! then restored to the analyzer. The analyzer is preset to the default
50  ! settings before the instrument state is transferred back.
60  !
70  ! EXAMP5C
80  !
90  ASSIGN @Nwa TO 716                   ! Assign an I/O path for the analyzer
100 ASSIGN @Nwa_bin TO 716;FORMAT OFF    ! Assign a binary path
110 !
120 INTEGER Head,Length                  ! Integer 2 byte format for headers
130 !
140 CLEAR SCREEN
150 ! Initialize the analyzer
160 ABORT 7                              ! Generate an IFC (Interface Clear)
170 CLEAR @Nwa                           ! SDC (Selected Device Clear)
180 !
190 ! Read in the learn string as a form 1 binary data trace
200 DIM Learn$[3000]                     ! Array to hold learn string
210 !
220 OUTPUT @Nwa;”OPC?;SING;”             ! Place analyzer in single sweep
230 ENTER @Nwa;Reply                     ! Read the 1 when complete
240 !
250 OUTPUT @Nwa;”OUTPLEAS;”              ! Request learn string
260 ENTER @Nwa USING “+,-K”;Learn$
270 !
280 ! Allocate an array for storing the CAL data
290 OUTPUT @Nwa;”POIN?;”                 ! Find number of points in trace
300 ENTER @Nwa;Num_points                ! Read number to allocate array
310 ALLOCATE Cal_array(1:Num_points,1:2)   ! Real and Imag for each point
320 !
330 ! Read Cal array
340 OUTPUT @Nwa;”FORM3;”                 ! Form 3 64 bit floating point data
350 OUTPUT @Nwa;”OUTPCALC01;”            ! Request the cal array
360 !
370 ! Read the #A and 2 byte length as integers
380 ENTER @Nwa_bin;Head,Length,Cal_array(*)   ! Read cal array data
390 !
400 ! Write instrument state data to disk file
410 ! CREATE BDAT “DATA_FILE:,1406”,1,Length+3000  ! Create data file once
420 ASSIGN @File TO “DATA_FILE:,1406          “    ! Assign I/O path to file
430 OUTPUT @File;Learn$                  ! Send learn string to disk file
440 OUTPUT @File;Head,Length,Cal_array(*)  ! Send CAL arrays to disk file
450 ASSIGN @File TO *                              ! Close file
460 !
470 INPUT “Cal data received. Press ENTER to send it back.”,A$
480 !
490 ! Read arrays from file
500 !
510 DIM Learn2$[3000]                    ! String for learn string storage
520 ASSIGN @File TO “DATA_FILE:,1406”    ! Open file for reading arrays
530 ENTER @File;Learn2$                  ! Read learn string from file
540 !
550 ENTER @File;Head,Length              ! Read CAL data headers from file
560 Size=Length/16                    ! Array is 2 numbers, 8 bytes per number
```

```
570 ALLOCATE Cal_array2(1:Size,1:2)      ! new cal array from file record
580 ENTER @File;Cal_array2(*)            ! Read cal array from disk file
590 !
600 ! Send Learn string back
610 OUTPUT @Nwa;”INPULEAS;”,Learn2$      ! Send learn string array
620 !
630 ! Send Cal array back
640 OUTPUT @Nwa;”CALIRESP;”              ! Send CAL type (Response)
650 OUTPUT @Nwa;”INPUCALC01;”            ! Output CAL array to analyzer
660 OUTPUT @Nwa_bin;Head,Length,Cal_array2(*)
670 OUTPUT @Nwa;”OPC?;SAVC;”             ! Save the CAL array
680 ENTER @Nwa;Reply                     ! Read the 1 when complete
690 !
700 OUTPUT @Nwa;”;CONT;”                 ! Start the analyzer sweeping
710 OUTPUT @Nwa;”OPC?;WAIT;”             ! Wait for the analyzer to finish
720 ENTER @Nwa;Reply
730 LOCAL @Nwa                           ! Release HP-IB control
740 END
```

Setting a List Frequency Sweep

File Name

EXAMP6A.BAS

Description

Using List-Frequency Mode
The analyzer normally takes data points spaced at regular intervals across the overall frequency range of the measurement. For example, for a 2 GHz frequency span using 201 points, data will be taken at intervals of 10 MHz. The list-frequency mode allows the operator to select the specific points, or frequency spacing between points, at which measurements are to be made. This mode of operation allows flexibility in setting up tests that insure device performance in an efficient manner. By only sampling specific points, measurement time is reduced. These programs emulate operation from the analyzer's front panel when using list sweeps.

The following two examples illustrate the use of the analyzer's list-frequency mode to perform arbitrary frequency testing. EXAMP6A allows the operator to construct a table of list-frequency segments which is then loaded into the analyzer's list-frequency table. There are a maximum of 30 segments available. Each segment stipulates a start and stop frequency, and the number of data points to be taken over that frequency range. EXAMP6B lets the operator select a specific segment to "zoom-in." A single instrument can be programmed to measure several different devices, each with its own frequency range, using a single calibration. When a specific device is connected, the operator selects the appropriate segment for that device. Note that list-frequency segments can be overlapped, but the total number of points in all the segments must not exceed 1632.

Example

The purpose of this example is to show how to create a list-frequency table and transmit it to the 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 takes the same steps as the front-panel key sequence, except that the analyzer automatically reorders each edited segment in order of increasing start frequency.

The list-frequency table is also carried as part of the learn string. While the table cannot be modified as part of the learn string, it can be stored and recalled with very little effort by storing and recalling the learn string. Refer to “Data-Processing Chain” on page 1-21 for additional information on using learn strings.

This example takes advantage of the computer’s ability to simplify:

- creating a list-frequency table
- editing a list-frequency table

The table is entered and completely edited before being transmitted to the analyzer. To simplify the programming task, options such as entering center frequency, frequency span, or step size are not included.

The list-frequency information may be acquired using the limit-test results array. The actual stimulus points are available as the first element in the array.

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

- The system is initialized.
- The existing list frequencies are edited and cleared.
- The number of segments to be defined is read in.
- An array for the list segments is defined.
- The parameters for each segment are requested.
- If the operator wants to edit, the segment parameters are re-entered.
- The new list is sent to the analyzer.
- The analyzer is released from remote control and the program ends.

Running the Program

CAUTION This example program will delete any existing limit lines before entering the new limits. If this is not desired, omit the line(s) that clear the existing limits (in this case, the executable `CLEL;`). This program begins by presetting the analyzer. The programmer will have to add the necessary command lines to set the analyzer to the specific operating conditions required for testing. The example program will show the limit lines defined, but the limits will always fail without additional analyzer setup.

The program displays the frequency-list table as it is entered. During editing, the displayed table is updated as each line is edited. The table is not re-ordered. At the completion of editing, the table is entered into the analyzer, and list-frequency mode is switched ON. During editing, pressing ENTER leaves an entry at the old value.

If the analyzer display is within the range of the segments entered, then the number of points-per-segment may be observed on the analyzer's display.

Activate a marker and select the discrete-marker mode to observe the point spacing. Use an exaggerated scale with just a few points to find the list-frequency spacing between points.

```
10  ! This program shows how to enter and edit a list frequency table.
20  ! Any existing table is deleted and a new table is defined and
30  ! edited. This list is then sent to the analyzer. Any number of
40  ! segments or points may be entered. Be sure not to enter more than
50  ! 1632 points or 30 segments.
60  !
70  !  EXAMP6A
80  !
90  ASSIGN @Nwa TO 716                ! Assign an I/O path for the analyzer
100 !
110 CLEAR SCREEN
120 ! Initialize the analyzer
130 ABORT 7                           ! Generate an IFC (Interface Clear)
140 CLEAR @Nwa                        ! SDC (Selective Device Clear)
150 OUTPUT @Nwa;”OPC?;PRES;”          ! Preset the analyzer and wait
160 ENTER @Nwa;Reply                  ! Read the 1 when complete
170 !
180 OUTPUT @Nwa;”EDITLIST;”           ! Begin editing the frequency list
190 OUTPUT @Nwa;”CLEL;”               ! Clear the existing list frequencies
200 !
210 INPUT “Number of segments?”,Numb   ! Read number of segments to define
220 ALLOCATE Table(1:Numb,1:3)        ! Define an array for the list segments
230 !
240 PRINT USING “10A,15A,15A,20A”;”SEGMENT”,”START(MHZ)”,”STOP(MHZ)”,”NUMBER OF
      POINTS”
250 !
260 FOR I=1 TO Numb        ! Cycle through the segments and read in the values
270   GOSUB Loadpoin
280 NEXT I
290 !
300 LOOP
310   INPUT “DO YOU WANT TO EDIT? Y OR N”,An$
320 EXIT IF An$=”N”
330   INPUT “ENTRY NUMBER?”,I          ! Get an entry number
340   GOSUB Loadpoin                  ! Go load point
350 END LOOP
360 !
370 OUTPUT @Nwa;”EDITLIST”            ! Send the new list to the analyzer
380 FOR I=1 TO Numb                           ! Send one segment at a time
390   OUTPUT @Nwa;”SADD;”                     ! Add a segment
400   OUTPUT @Nwa;”STAR”;Table(I,1);”MHZ;”     ! Start frequency
410   OUTPUT @Nwa;”STOP”;Table(I,2);”MHZ;”     ! Stop frequency
420   OUTPUT @Nwa;”POIN”,Table(I,3),”;”        ! Number of points
430   OUTPUT @Nwa;”SDON;”                     ! Segment done
440 NEXT I                            ! Next segment to send to the analyzer
450 !
460 OUTPUT @Nwa;”EDITDONE;”           ! Done with list
470 OUTPUT @Nwa;”LISFREQ;”            ! Set list frequency mode
480 !
490 OUTPUT @Nwa;”OPC?;WAIT;”          ! Wait for analyzer to finish
500 ENTER @Nwa;Reply                  ! Read the 1 when complete
510 LOCAL @Nwa                        ! Release HP-IB control
520 STOP                             ! End of main program
530 !
540 !   ***************************Subroutines ******************************
550 !
```

```
560 Loadpoin:                          ! Sub to read in each segment value
570 INPUT “START FREQUENCY? (MHZ)”,Table(I,1)   ! Read start frequency
580 INPUT “STOP FREQUENCY? (MHZ)”,Table(I,2)    ! Read stop frequency
590 INPUT “NUMBER OF POINTS?”,Table(I,3)        ! Read number of points in seg
600 IF Table(I,3)=1 THEN Table(I,2)=Table(I,1)  ! Single point same start stop
610 !
620 ! Print new segment into table on display
630 PRINT TABXY(0,I+1);I;TAB(10);Table(I,1);TAB(25);
640 PRINT Table(I,2);TAB(40),Table(I,3)
650 RETURN
660 END
```

Selecting a Single Segment

File Name EXAMP6B.BAS

Description This example program demonstrates how to define a single segment as the operating-frequency range of the analyzer from a table of segments stored in the controller. The program assumes that a list-frequency table has already been entered into the analyzer, either manually, or using the program in “Setting a List Frequency Sweep” on page 2-56.

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 operator can choose which segment is to be used by the analyzer. Note that only one segment can be chosen at a time.

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

- The system is initialized.
- The list-frequency segment is edited.
- The largest segment number possible is set.
- The highest segment number in use is requested.
- The number of actual segments in use is read in.
- A list-frequency table is defined and the segments are read in to the controller from the analyzer.
- The operator selects one of the segments of the sweep.
- The controller "zooms-in" and sweeps the defined segment.
- The operator ends the program by entering segment number (0).
- The analyzer returns to sweeping all the segments in the table.
- The activation loop is ended and the program ends.

Running the Program

The program will read the parameters for each list-frequency segment from the analyzer, and build a table containing all the segments. The parameters of each segment will be printed on the computer screen.

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

```
10  ! This program shows how to select a single segment from a list
20  ! frequency sweep and activate it as the sweep. The list frequency
30  ! table is read from the analyzer and displayed on the computer
40  ! screen. The operator is prompted to select a segment and the
50  ! program then activates it. All the segments are activated upon
60  ! completion.
70  !
80  ! EXAMP6B
90  !
100 ASSIGN @Nwa TO 716                   !  Assign an I/O path for the analyzer
110 !
120 CLEAR SCREEN
130 ! Initialize the analyzer
140 ABORT 7                              ! Generate an IFC (Interface Clear)
150 CLEAR @Nwa                           ! SDC (Selected Device Clear)
160 !
170 ! Print header for table of existing segments
180 PRINT USING "10A,15A,15A,20A";"SEGMENT","START(MHZ)","STOP(MHZ)","NUMBER OF
       POINTS"
190 OUTPUT @Nwa;"EDITLIST;"              ! Edit list frequency segment
200 OUTPUT @Nwa;"SEDI30;"                ! Set largest segment number
210 OUTPUT @Nwa;"SEDI?;"                 ! Request number of highest segment
220 ENTER @Nwa;Numsegs                   ! Read number of actual segments
230 !
240 ! Setup table and read segments from analyzer
250 ALLOCATE Table(1:Numsegs,1:3)        ! Allocate table of segments
260 FOR I=1 TO Numsegs                   ! Cycle through segments
270   GOSUB Readlist                     ! Read in segment definitions
280 NEXT I                               ! Next segment
290 !
300 ! Loop and read segment to be activated
310 LOOP                                 ! Request operator to enter segment
320   INPUT "SELECT SEGMENT NUMBER: (0 TO EXIT)",Segment
330 EXIT IF Segment=0                    ! Exit point
340   OUTPUT @Nwa;"EDITDONE;";"SSEG";Segment;";" ! Set active segment to sweep
350 END LOOP                             ! End activation loop
360 !
370 OUTPUT @Nwa;"ASEG;"                  ! Set all segment sweep
380 DISP "PROGRAM ENDED"
390 !
400 OUTPUT @Nwa;"OPC?;WAIT;"             ! Wait for analyzer to finish
410 ENTER @Nwa;Reply                     ! Read the 1 when complete
420 LOCAL @Nwa                           ! Release HP-IB control
430 STOP                                 ! End of main program
440 !
450 ! ************************** Subroutines *******************************
460 !
470 Readlist:                            ! Read segment list from analyzer
480 OUTPUT @Nwa;"EDITLIST;"              ! Edit segment list
490 OUTPUT @Nwa;"SEDI",I,";"             ! Select segment to edit
500 OUTPUT @Nwa;"STAR;"                 ! Send start freq to display value
510 OUTPUT @Nwa;"OUTPACTI;"              ! Output active function value
520 ENTER @Nwa;Table(I,1)               ! Read start frequency
530 OUTPUT @Nwa;"STOP;"                  ! Send stop freq to display value
540 OUTPUT @Nwa;"OUTPACTI;"              ! Output active function value
550 ENTER @Nwa;Table(I,2)                ! Read stop frequency
```

```
560 OUTPUT @Nwa;”POIN;”                 ! Send number of points to display
570 OUTPUT @Nwa;”OUTPACTI;”             ! Output active function value
580 ENTER @Nwa;Table(I,3)               ! Read number of points
590 !
600 IF I=18 THEN                        ! Pause if more than 17 segments
610 INPUT “PRESS RETURN FOR MORE”,A$    ! Read Return to continue
620 END IF
630 ! Print new header for segment data
640 IMAGE 4D,6X,4D.6D,3X,4D.6D,3X,4D    ! Format image to disp segment data
650 PRINT USING 640;I;Table(I,1)/1.E+6;Table(I,2)/1.E+6;Table(I,3)
660 RETURN
670 !
680 END
```

Setting up Limit Lines

File Name EXAMP6C.BAS

Description The purpose of this example is to show how to create a limit-test table and transmit it to the analyzer.

The command sequence for entering a limit-test table imitates the key sequence followed when entering a table from the analyzer's front panel. There is a command for every key-press. Editing a limit is also the same as the key sequence, but remember that the analyzer automatically reorders the table in order of increasing start frequency.

The limit-test table is also carried as part of the learn string. While it cannot be modified as part of the learn string, the learn string can be stored and recalled with very little effort. Refer to "Data-Processing Chain" on page 1-21 for details on using learn strings.

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

This example automates the front-panel operation of entering a limit-test table.

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

- The system is initialized.
- The limit lines are edited and cleared.
- The number of limits is requested.
- The limit table is created.
- The string array of limit types is created.
- The operator is prompted to enter the new limit values.
- The new limit table is sent back to the analyzer.
- The limit line is activated.
- The limit test is activated.
- The analyzer is returned to local control and the program ends.

Running the Program

CAUTION This example program will delete any existing limit lines before entering the new limits. If this is not desired, omit the line(s) that clear the existing limits (in this case, the executable `CLEL;` contained in LINE 190). This program begins by presetting the analyzer. The programmer will have to add the necessary command lines to set the analyzer to the operating conditions required for testing. The example program will show the limit lines defined, but the limits will always fail without additional analyzer setup.

The program displays the limit table as it is entered. During editing, the displayed table is updated as each line is edited. The table is not reordered. At the completion of editing, the table is entered into the analyzer, and limit-testing mode switched ON. The analyzer will rearrange the table in ascending order starting with the lowest start frequency entry.

```
10  ! This program shows how to create a limit table and send it to the
20  ! analyzer. The operator enters the desired limits when prompted for
30  ! the stimulus value, upper and lower value and type of limit
40  ! desired. Once the table is created, the limits are sent to the
50  ! analyzer and activated.
60  !
70  ! EXAMP6C
80  !
90  ASSIGN @Nwa TO 716                   ! Assign an I/O path for the analyzer
100 !
110 CLEAR SCREEN
120 ! Initialize the analyzer
130 ABORT 7                              ! Generate an IFC (Interface clear)
140 CLEAR @Nwa                           ! SDC (Selected Device Clear)
150 OUTPUT @Nwa;”OPC?;PRES;”             ! Preset the analyzer and wait
160 ENTER @Nwa;Reply                     ! Read the 1 when completed
170 !
180 OUTPUT @Nwa;”EDITLIML;”              ! Edit limit lines
190 OUTPUT @Nwa;”CLEL;”                  ! Clear any existing limits
200 INPUT “NUMBER OF LIMITS?”,Numb       ! Request the number of limits
210 ALLOCATE Table(1:Numb,1:3)           ! Create a table
220 ALLOCATE Limtype$(Numb)[2]           ! Create string array of limit types
230 !
240 ! Print out the header for the table
250 PRINT USING “10A,20A,15A,20A”;”SEG”,”STIMULUS (MHz)”,”UPPER (dB)”,”LOWER
      (dB)”,”TYPE”
260 !
270 ! Prompt the operator to enter the limit values
280 FOR I=1 TO Numb                      ! Cycle through the limits
290   GOSUB Loadlimit                    ! Go read limit values
300 NEXT I                               ! Next limit value
310 !
320 ! Allow the operator to edit the array entered
330 LOOP                                 ! Cycle to edit limit lines
340   INPUT “DO YOU WANT TO EDIT? Y OR N”,An$
350 EXIT IF An$=”N”                      ! Exit loop on N and send to analyzer
360   INPUT “ENTRY NUMBER?”,I            ! Read limit number to edit
370   GOSUB Loadlimit                    ! Go read limit values
380 END LOOP                            ! Next edit entry
390 !
400 ! Send the limit line array segments to the analyzer
410 OUTPUT @Nwa;”EDITLIML;”              ! Edit the limit
420 FOR I=1 TO Numb                      ! Each segment of the limit
430   OUTPUT @Nwa;”SADD;”                  ! Add segment
440   OUTPUT @Nwa;”LIMS”;Table(I,1);”MHZ”  ! Send segment stimulus value
450   OUTPUT @Nwa;”LIMU”;Table(I,2);”DB”   ! Upper limit value
460   OUTPUT @Nwa;”LIML”;Table(I,3);”DB”   ! Lower limit value
470   IF Limtype$(I)=”FL” THEN OUTPUT @Nwa;”LIMTFL;” ! Flat limit
480   IF Limtype$(I)=”SL” THEN OUTPUT @Nwa;”LIMTSL;” ! Sloping limit
490   IF Limtype$(I)=”SP” THEN OUTPUT @Nwa;”LIMTSP;” ! Point limit
500   OUTPUT @Nwa;”SDON;”                ! Segment done
510 NEXT I                               ! Next segment
520 !
530 OUTPUT @Nwa;”EDITDONE;”              ! Edit complete
540 OUTPUT @Nwa;”LIMILINEON;”            ! Turn limit line on
550 OUTPUT @Nwa;”LIMITESTON;”            ! Turn limit test on
```

```
560 !
570 OUTPUT @Nwa;"OPC?;WAIT;"              ! Wait for the analyzer to finish
580 ENTER @Nwa;Reply                      ! Read the 1 when complete
590 !
600 LOCAL @Nwa                            ! Release HP-IB control
610 STOP                                  ! End of main program
620 !
630 !**************************** Subroutines ********************************
640 !
650 Loadlimit:                            ! Sub to interact to load data
660 INPUT "STIMULUS VALUE? (MHz)",Table(I,1)  ! and print table created
670 INPUT "UPPER LIMIT VALUE? (DB)",Table(I,2)
680 INPUT "LOWER LIMIT VALUE? (DB)",Table(I,3)
690 INPUT "LIMIT TYPE? (FL=FLAT, SL=SLOPED, SP=SINGLE POINT)",Limtype$(I)
700 !
710 ! Format and display table values
720 PRINT TABXY(0,I+1);I;TAB(10);Table(I,1);TAB(30);Table(I,2);TAB(45),
     Table(I,3),TAB(67);Limtype$(I)
730 RETURN                                 ! Next limit value
740 !
750 END
```

Performing Pass/Fail Tests

File Name EXAMP6D.BAS

Description 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 “Using the Learn String” on page 2-45.

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. Refer to Table 5-2, “Units as a Function of Display Format,” on page 5-6 for additional information. Their purpose is to allow the computer to determine whether the test/search executed was successful. They are used in the following sequence:

1. Clear event status register B.
2. Trigger the limit test or marker search.
3. Check the appropriate fail bit.

When using limit testing, the best way to trigger the limit test is to trigger a single sweep. By the time the single sweep command `SING;` finishes, limit testing will have occurred.

NOTE

If the device is tuned during the sweep, it may be tuned into and then out of limit, causing a limit test to qualify as "passed" when the device is not in fact within the specified limits.

When using marker searches (max, min, target, and widths), outputting marker or bandwidth values automatically triggers any related searches. Therefore, all that is required is to check the fail bit after reading the data.

In this example, several consecutive sweeps must qualify as "passing" in order to insure that the limit-test pass was not extraneous due to the device settling or operator tuning during the sweep. Upon running the program, the number of "passed" sweeps for qualification is entered. For very slow sweeps, a small number of sweeps, such as two, are appropriate. For relatively fast sweeps, where the device requires time to settle after tuning, as many sweeps as six or more sweeps may be more 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, as shown in “Setting a List Frequency Sweep” on page 2-56. The manual (front-panel entry) sequence is closely followed.

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

- The system is initialized.
- The pass counter is initialized on entry.
- The analyzer takes a sweep.
- The event status register B byte is output and the channel 1 limit is tested.
- If the device fails the first sweep, the operator is prompted to insure it is tuned correctly and the device is measured again.
- Once the device passes, the operator is prompted not to touch the device as testing continues.
- If the device passes the required number of sweeps, the operator is prompted that the device has passed and to connect the next device for testing.
- The program initializes the pass counter and begins to measure the new device.

Running the Program

NOTE

This program assumes a response calibration (thru calibration) or a full 2-port measurement calibration has been performed prior to running the program.

Set up a limit-test table on channel 1 for a specific device either manually, or using the program in “Using the Learn String” on page 2-45.

Run the program, and enter the number of passed sweeps desired for qualification. After entering the qualification number, connect the device. When a sweep passes, the computer beeps. When enough consecutive sweeps qualify the device as "passing," the computer emits a dual-tone beep to attract the attention of the operator, and then prompts for a new device.

To test the program's pass/fail accuracy, try causing the device under test to fail by loosening the cables connecting the device under test to the analyzer and running the program again.

```
10  ! This program demonstrates Pass/Fail tests using limit lines. The
20  ! program uses the latch-on-fail limit bits in event status register
30  ! B to determine if the device performance passes the specified test
40  ! limit lines.  It then requires that the device passes for multiple
50  ! consecutive sweeps in order to ensure that the device is static in
60  ! the response and not varying. The operator specifies how many sweeps
70  ! are required to pass the test.
80  !
90  ! EXAMP6D
100 !
110 ASSIGN @Nwa TO 716                     ! Assign an I/O path for the analyzer
120 !
130 CLEAR SCREEN
140 ! Initialize the analyzer  No preset to retain settings for testing
150 ABORT 7                                ! Generate an IFC (Interface Clear)
160 CLEAR @Nwa                             ! SDC (Selected Device Clear)
170 !
180 INPUT “Number of consecutive passed sweeps for qualification?”,Qual
190 Pass=0                                 ! Initialize pass counter on entry
200 !
210 Tune: DISP “TUNE DEVICE AS NECESSARY”  ! Device is not passing warning
220 !
230 Measure:OUTPUT @Nwa;”OPC?;SING;”      ! Single sweep and wait
240 ENTER @Nwa;Reply                       ! Read the 1 when completed
250 !
260 OUTPUT @Nwa;”ESB?;”                    ! Event status register B byte
270 ENTER @Nwa;Estat                       ! Reading byte clears the register
280 !
290 IF BIT(Estat,4) THEN                   ! Bit 4 is failed limit on channel 1
300      IF Pass>0 THEN BEEP 1200,.05     ! passed before? Now not passing beep
310      Pass=0                            ! Reset pass to 0
320      GOTO Tune                         ! Adjust and measure again
330 END IF
340   !
350 BEEP 2500,.01                          ! Limit test passed passing beep
360 Pass=Pass+1                            ! Increment number of passes
370 DISP “LEAVE DEVICE ALONE”              ! Warn not to adjust as it passed
380 !
390 IF Pass<Qual THEN GOTO Measure         ! If not enough passes to qualify
400 !
410 ! Device passed
420 DISP “DEVICE PASSED!”                  ! Number of passes enough to qualify
430 FOR I=1 TO 10                          ! Announce the device passed and
440      BEEP 1000,.05                     ! prompt operator to connect new
450      BEEP 2000,.01                     ! device to test.
460 NEXT I
470 !
480 INPUT “PRESS RETURN FOR NEXT DEVICE”,Dum$
490 Pass=0                                 ! Initialize pass counter
500 GOTO Measure                           ! Begin measurement
510 !
520 END
```

Operation Using Talker/Listener Mode

File Name EXAMP7A1.BAS

Description The analyzer has three operating modes with respect to HP-IB. These modes can be changed by accessing softkeys in the LOCAL menu. System-controller mode is used when no computer is present. This mode allows the analyzer to control the system. The other two modes allow a remote system controller to coordinate certain actions: in talker/listener mode the remote system controller can control the analyzer, as well as coordinate plotting and printing; and in pass-control mode the remote system controller can pass active control to the analyzer so that the analyzer can plot, print, control a power meter, or load/ store to disk. The amount of peripheral interaction is the main difference between talker/listener and pass-control mode.

Example The commands `OUTPPLOT` and `OUTPPRIN` allow talker/listener mode plotting and printing via a one way data path from the analyzer to the plotter or printer. The computer sets up the path by addressing the analyzer to talk, the plotter to listen, and then releases control of the analyzer in order to transfer the data. The analyzer will then make the plot or print. When it is finished, it asserts the End or Identify (EOI) control line on HP-IB. The controller detects the presence of EOI and re-asserts control of the HP-IB. This example program makes a plot using the talker/listener mode.

NOTE

One of the attributes of the `OUTPPLOT` command is that the plot can include the current softkey menu. The plotting of the softkeys is enabled with the `PSOFTON;` command and disabled with the `PSOFTOFF;` command.

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

- The system is initialized.
- The selected frequency span is swept once.
- The plot command is sent to the analyzer.
- The analyzer is set to talker mode and the plotter is set to listener mode.
- The plot is spooled to the plotter.
- The analyzer is set to listener mode when the controller detects an EOI from the analyzer.
- The controller puts the analyzer back in continuous-sweep mode.
- The analyzer is returned to local control and the program ends.

Running the Program

If a problem arises with the plotter, such as no pen or paper, the analyzer cannot detect the situation because it only has a one-way path of communication. Therefore, the analyzer attempts to continue plotting until you intervene and abort the plot by pressing the LOCAL key. This key aborts the plot, causes the warning message, `CAUTION: PLOT ABORTED,` asserts EOI, and frees the computer.

Because of possible peripheral malfunctions, it is generally advisable to use pass-control mode, which allows two way communication between the peripherals and the analyzer.

```
10  ! This example shows a plot operation under the control of the
20  ! analyzer. The analyzer is commanded to output plot data, the
30  ! plotter is addressed to listen, and the analyzer to talk. The
40  ! controller watches for EOI at the end of the plot sequence and
50  ! then regains control of the HP-IB operations.
60  !
70  ! EXAMP7A1
80  !
90  ASSIGN @Nwa TO 716                     ! Assign an I/O path for the analyzer
100 !
110 CLEAR SCREEN
120 ! Initialize analyzer without preset to preserve data
130 ABORT 7                                ! Generate an IFC (Interface Clear)
140 CLEAR @Nwa                             ! SDC (Selected Device Clear)
150 !
160 OUTPUT @Nwa;”OPC?;SING;”               ! Stop sweep and prepare for plot
170 ENTER @Nwa;Reply                       ! Read in “1” when completed
180 !
190 OUTPUT @Nwa;”OUTPPLOT;”               ! Send plot command
200 SEND 7;UNL LISTEN 5 TALK 16 DATA      ! Unlisten address devices and plot
210 DISP “Plotting and waiting for EOI”
220 WAIT .5                                ! Pause 500 mS to start process
230 !
240 REPEAT                                 ! Loop until EOI detected bit is set
250   STATUS 7,7;Stat                      ! Read HP-IB interface register 7
260 UNTIL BIT(Stat,11)                     ! Test bit 11 EOI on HP-IB
270 !
280 End_plot:DISP “End of plot”
290 !
300 OUTPUT @Nwa;”CONT;”                    ! Restore continuous sweep
310 OUTPUT @Nwa;”OPC?;WAIT;”               ! Wait for analyzer to finish
320 ENTER @Nwa;Reply                       ! Read the 1 when complete
330 LOCAL @Nwa                             ! Release remote control
340 END
```

Controlling Peripherals

File Name EXAMP7A2.BAS

Description If the analyzer is in pass-control mode and it receives a command telling it to plot, print, control a power meter, or store/load to disk, it sets bit 1 in the event-status register to indicate that it requires control of the bus. If the computer then uses the HP-IB pass-control command to pass control to the analyzer, the analyzer will take control of the bus and access the peripheral. When the analyzer no longer requires control, it will pass control back to the computer. For a discussion on the pass-control mode, refer to “Pass-control mode” on page 1-10.

In this example, the pass-control mode is used to allow the analyzer to dump a screen display to a printer.

Pass-control mode allows the analyzer to control the printer while sending the screen display to be printed. Once the printer-dump operation is complete, the analyzer passes control back to the controller and the controller continues programming the analyzer. The analyzer requests control from the instrument controller and the controller allows the analyzer to take control of the HP-IB and dump the plot. The instrument controller must not interact with the HP-IB while this remote analyzer control is taking place.

NOTE

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

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

When the analyzer becomes the active system controller, it is free to address devices to talk and listen as required. The only functions denied the analyzer are the ability to assert the interface clear line (IFC), and the remote line (REN). These are reserved for the master system controller. As the active system controller, the analyzer can send and receive messages from printers, plotters, and disk drives.

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

- The system is initialized.
- The status registers are cleared.
- Bit 1 of ESR request control is set.
- The ESR interrupt for SRQ is enabled.
- The pass-control mode is enabled.
- The data is dumped to the printer.
- The program loops until the SRQ bit is set.
- The status byte is read with a serial poll.
- The program tests for bit 6, SRQ.
- If SRQ is detected, the program tests for pass control (bit 5 of the status byte).
- If the analyzer requests control, the system controller gives the analyzer control of the bus.
- The program loops and waits for the analyzer to complete the print dump.
- The controller reads the HP-IB interface status register.
- If bit 6 is active in the controller, control has returned from the analyzer to the controller.
- The status-byte assignments are cleared.
- The analyzer returns to continuous-sweep mode.
- The analyzer is returned to local control and the program ends.

Running the Program

The analyzer will briefly flash the message `WAITING FOR CONTROL`, before actually receiving control and generating the printer output. The computer will display the `Printing from analyzer and waiting for control` message.

When the printer output is complete, the analyzer passes control back to the address stored as the controller address under the LOCAL, *SET ADDRESSES* menu. The computer will detect the return of active control and exit the wait loop. The controller will display the message `Control returned from analyzer` and then release the analyzer from remote control.

NOTE

Because the program waits for the analyzer's request for control, it can be used to respond to front-panel requests as well. Remove the `PRINALL;` command from the program and run the program. Nothing will happen until the operator requests a print, plot, or disk access from the front panel of the analyzer. For example, press LOCAL, COPY, *PRINT MONOCHROME*.

```
10  ! This example shows a pass-control operation to print the display
20  ! under the analyzer HP-IB control. The controller reads the status
30  ! of the analyzer looking for SRQ to indicate that the analyzer is
40  ! requesting control. Once control is passed to the analyzer, the
50  ! controller monitors the status of its interface registers to detect
60  ! when the interface is again the active controller. The analyzer will
70  ! pass control back to the controller when finished.
80  !
90  ! EXAMP7A2
100 !
110 ASSIGN @Nwa TO 716                   ! Assign an I/O path for the analyzer
120 !
130 CLEAR SCREEN
140 ! Initialize the analyzer without preset to preserve data
150 ABORT 7                              ! Generate an IFC (Interface Clear)
160 CLEAR @Nwa                           ! SDC (Selected Device Clear)
170 !
180 OUTPUT @Nwa;”OPC?;SING;”             ! Single sweep and stop for print
190 ENTER @Nwa;Reply                     ! Read in “1” when complete
200 !
210 OUTPUT @Nwa;”CLES;”                  ! Clear status registers
220 OUTPUT @Nwa;”ESE2;”                  ! Enable bit 1 of ESR request control
230 OUTPUT @Nwa;”SRE32;”                 ! Enable ESR interrupt for SRQ
240 !
250 OUTPUT @Nwa;”USEPASC;”               ! Enable pass control mode
260 OUTPUT @Nwa;”PRINALL;”               ! Begin printer dump
270 !
280 REPEAT                               ! Loop until SRQ bit is set
290   Stat=SPOLL(@Nwa)                   ! Read status byte with serial poll
300 UNTIL BIT(Stat,6)                    ! Test for bit 6, SRQ
310 !
320 Pass_control:                        ! SRQ detected. Test for pass control
330 IF BIT(Stat,5) THEN                  ! Requested pass control
340   PASS CONTROL @Nwa                  ! Send take control message
350 ELSE                                 ! Not bit 5, some other event
360   DISP “SRQ but not request pass control”
370   STOP                               ! Halt program
380 END IF
390 !
400 DISP “Printing from analyzer and waiting for control”
410 !
420 REPEAT                               ! Loop and wait for completion
430   STATUS 7,6;Hpib                    ! Read HP-IB interface register
440 UNTIL BIT(Hpib,6)                    ! Bit 6 is active controller
450 !
460 DISP “Control returned from analyzer”
470 OUTPUT @Nwa;”TALKLIST;”              ! Set talker/listener mode again
480 OUTPUT @Nwa;”CLES;”                  ! Clear status byte assignments
490 !
500 OUTPUT @Nwa;”CONT;”                  ! Start analyzer sweeping again
510 OUTPUT @Nwa;”OPC?;WAIT;”             ! Wait for analyzer to finish
520 ENTER @Nwa;Reply                     ! Read the 1 when complete
530 !
540 LOCAL @Nwa                           ! Release HP-IB control
550 END
```

Printing Via the Serial Port

File Name EXAMP7A3.BAS

Description This program will select the serial port and program the analyzer to copy its display to a printer. There are a number of commands associated with the serial and parallel ports which allow the programmer to configure the output modes, for example the baud rate and the handshake type used by the port and the printer. In this example, the serial port is configured by the program. The interface may also be configured from the analyzer's front-panel keys by pressing LOCAL, *SET ADDRESSES*, *PRINTER PORT*. This menu allows manual selection of the serial-interface parameters.

Since the HP-IB port is not being used for the copy operation, programming of the analyzer and measurement operations may continue once the copy operation has been initiated. An internal spooler in the analyzer's memory provides buffering of the printer operation. In the example which follows, the status byte of the analyzer is checked to determine when the print operation is complete.

- The analyzer is initialized.
- A single sweep is taken and the analyzer is placed in hold mode.
- The status registers are cleared.
- The copy-complete bit is set and enabled.
- The printer operation and communication modes are set.
- The print command is sent.
- The analyzer is released from remote control and placed in continuous-sweep mode.
- The analyzer is polled until the status bit representing copy complete is detected.
- The analyzer is released from remote control and the program ends.

Running the Program

Run the program. The analyzer is initialized, set to single-sweep mode, and a sweep is taken. The program sets the system up to print the analyzer's display to an HP ThinkJet printer connected to the interface. At this time, the analyzer can continue making measurements as the ThinkJet prints the display. When the analyzer display has finished printing, the controller displays the message: `DONE`, the analyzer is released from HP-IB control, and the program ends.

```
10  ! This program shows how to set up and print the display through the
20  ! serial printer port.
30  !
40  ! EXAMP7A3
50  !
60  ASSIGN @Nwa TO 716                    ! Assign an I/O path for the analyzer
70  !
80  CLEAR SCREEN
90  ! Initialize the analyzer without preset to preserve the data
100 ABORT 7                               ! Generate an IFC (Interface Clear)
110 CLEAR @Nwa                            ! SDC (Selected Device Clear)
120 !
130 OUTPUT @Nwa;”OPC?;SING;”              ! Single sweep and stop for print
140 ENTER @Nwa;Reply                      ! Read the 1 when complete
150 !
160 OUTPUT @Nwa;”CLES;”                   ! Clear status registers
170 OUTPUT @Nwa;”ESNB128;”                ! Enable copy complete
180 OUTPUT @Nwa;”SRE4;”                   ! Enable Event Status Register B
190 OUTPUT @Nwa;”PRNTRAUTF OFF;”          ! Set printer auto feed off
200 OUTPUT @Nwa;”PRNTYPTJ;”               ! Select ThinkJet printer
210 OUTPUT @Nwa;”PRNPRTSERI;”             ! Select serial port for output
220 OUTPUT @Nwa;”PRNTRBAUD 9600;”         ! Set baud rate to 9600 bps
230 OUTPUT @Nwa;”PRNHNDSHK XON;”          ! Use Xon-Xoff handshake
240 !
250 OUTPUT @Nwa;”PRINALL;”                ! Print screen
260 !
270 DISP “PRINTING”
280 !
290 !  Set up next measurement over HP-IB
300 OUTPUT @Nwa;”CONT;”                   ! Restore continuous sweep
310 !
320 ! Measurements can continue but wait for print to finish
330 REPEAT                                ! Test for bit 2 (4) ESRB
340 Stat=SPOLL(@Nwa)
350 UNTIL BIT(Stat,2)                     ! Wait for printer to complete
360 !
370 DISP “DONE”
380 LOCAL @Nwa                            ! Release HP-IB control
390 END
```

Plotting Data

File Name EXAMP7B1.BAS

Description Another report-generation technique is to transfer the plotter string to a disk file, and retrieve and plot the disk file at another time. Test time is increased when a plot occurs during the process. It may be more convenient to plot the data at another site or time. One solution to this problem is to capture the plot data using the controller and store it to a disk file. This disk file may then be read from the controller and the contents transferred to a plotter. This next example shows a method of accomplishing this task.

The analyzer is initialized without presetting the analyzer. The data that is in place on the analyzer is not disturbed by the program operation. A large string is dimensioned to hold the plotter commands as they are received from the analyzer. The length of this string depends upon the complexity of the analyzer's display. The analyzer is placed in the single-sweep mode and `OPC?;SING;` is used to make sure that operation is complete before plotting. The plotting begins with the `OUTPPLOT;` command.

The string transfer is ended by the controller detecting the EOI line which the analyzer pulls at the end of the transfer. The string transfer terminates and the plot data is now stored in a string in the analyzer.

These strings contain ASCII characters which represent the plotter commands in HP-GL (Hewlett-Packard Graphics Language). A disk file is created and the string is written into the file containing the display-plot commands.

Once the strings are transferred to the disk file, the file pointer is rewound and the data read out into a string for plotting. The string is sent to the plotter which uses the commands to generate a plot.

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

- The system is initialized.
- The string for plotter commands is defined.
- The frequency span is swept once.
- The plotter output is requested and read into the plot string.
- A plot file is created in the controller.
- The plot string is stored into the disk file.
- The plot string is read from the disk file and sent to the plotter.
- The analyzer returns to continuous-sweep mode.
- The analyzer is returned to local control and the program ends.

Running the Program

The program begins by initializing the analyzer and placing it into single-sweep mode. The plotter commands are captured into strings in the controller. The controller display prompts, `Plotter output complete. Press RETURN to store on disk.` Pressing ENTER causes the data to be stored to disk. Once this task is complete, the program prompts once more, `Plot to file is complete. Press Return to plot.` After pressing ENTER again, the string output is sent to the plotter and the plot begins. Once the plot is complete, the program prompts, `Plot is complete. End of program.`, and the analyzer begins sweeping and returns to local control.

```
10  ! This program shows how to read the plotter output from the analyzer
20  ! and store it in a disk file as an ASCII file. The disk file is then
30  ! read back into the controller and the plot commands sent to a
40  ! plotter to generate the plot of the analyzer display. This allows
50  ! plotting at a different time than data collection.
60  !
70  ! EXAMP7B1
80  !
90  ASSIGN @Nwa TO 716                   ! Assign an I/O path for the analyzer
100 ASSIGN @Plt TO 705                   ! Assign an I/O path for the plotter
110 !
120 CLEAR SCREEN
130 ! Initialize the analyzer without preset to preserve data
140 ABORT 7                              ! Generate an IFC (Interface Clear)
150 CLEAR @Nwa                           ! SDC (Selected Device Clear)
160 !
170 DIM Plot$[32000]                     ! Define string for plotter commands
180 !
190 OUTPUT @Nwa;”OPC?;SING;”             ! Stop sweep for plot and wait
200 ENTER @Nwa;Reply                     ! Read the 1 when complete
210 OUTPUT @Nwa;”OUTPPLOT;”              ! Request plotter output
220 !
230 ENTER @Nwa;Plot$                     ! Plotter output of analyzer display
240 !
250 INPUT “Plotter output complete. Press RETURN to store on disk.”,Reply$
260 !
270 ! Disk file operations
280 ! Create data file on disk  32000/256 = 125 records
290 !CREATE ASCII “PLOTFILE:,1400”,125   ! Use only once to generate file
300 ASSIGN @File TO “PLOTFILE:,1400”     ! Assign file I/O path
310 OUTPUT @File;Plot$                   ! Write plot string to file
320 !
330 INPUT “Plot to file is complete. Press Return to plot.”,A$
340 !
350 ! Read plotter commands from file and send to plotter
360 RESET @File                          ! Reset file pointer to beginning
370 ENTER @File;Plot$                    ! Read plot string from file
380 OUTPUT @Plt;Plot$                    ! Send plot string to plotter
390 !
400 !
410 DISP “Plot is complete. End of program.”
420 OUTPUT @Nwa;”CONT;”                  ! Restore continuous sweep
430 OUTPUT @Nwa;”OPC?;WAIT;”             ! Wait for analyzer to finish
440 ENTER @Nwa;Reply                     ! Read the 1 when complete
450 LOCAL @Nwa                           ! Release HP-IB control
460 END
```

Reading Plot Files from Disk

File Name EXAMP7B2.BAS

Description This example program reads and plots files which have been stored on a LIF formatted disk by the HP 8702D Option 011. The plots may be sent to either a plotter with autofeed capability, such as the HP 7550B, or an HP-GL/2 compatible printer, such as a LaserJet 4 Series (monochrome) printer or a DeskJet 1200C (color) printer.

NOTE

Sending plots to disk is discussed in the chapter titled "Printing, Plotting, and Saving Measurement Results" in the *HP 8702D User's Guide*.

This section provides detailed information on file naming conventions and instructions on printing multiple plots-per-page.

This program requires HP BASIC 6.0 or greater which provides the use of wild cards with the catalog command.

The peripheral HP-IB addresses and the hard copy device selection are hard coded and may need to be changed for different systems configurations. Refer to lines 1130 through 1240 in the example program.

This program example provides the form feeds to separate the plots. If the HP 8702D Option 011 has been configured to store multiple plots-per-page, this program will generate those plots. A file naming convention has been devised to allow the program to perform several printer-setup functions. These include: initializing the printer for HP-GL/2 at the beginning of a page, configuring the printer to plot multiple plots to the same page, if desired, and then sending a page eject (form feed) to the hardcopy device at the completion of the printing process.

The plot file name is made up of four parts. The first three are generated automatically by the HP 8702D Option 011 whenever a plot is requested:

1. the prefix "PLOT"
2. a two-digit sequence number in the range of 00 to 31
3. a two-letter output-format code to indicate the plot quadrant position:

- LU (Left Upper)
- LL (Left Lower)
- RU (Right Upper)
- RL (Right Lower)
- FP (Full Page)

For example, the first full page plot to a disk would be named "PLOT00FP". The second plot, to be located in the lower-right hand corner of the page would be named "PLOT01RL," and so on.

The fourth part is an optional character. It is used to indicate that the file is part of a multiple-file plot on the same graticule. See "Printing Multiple Measurements Per Page" in the *HP 8702D User's Guide*.

For detailed information on plotting to disk and outputting the plot files to a printer/plotter see "Printing, Plotting, and Saving Measurements Results" in the *HP 8702D User's Guide.*

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

- Hardcopy device control strings are created.
- The hardcopy output device is defined.
- The disk storing the file names is cataloged.
- Files that match the name specifier in the file name array (Flnm$) are counted.
- Flnm$ is dimensioned for the number of files that match the name specifier.
- The file name array is sorted.
- A form feed is sent to the hardcopy device.
- Each of the files in the file name array is processed and sent to the output device.
- The current file name root string is defined.
- If the hardcopy device is a printer, then an HP-GL initialization string is output.
- Each file which matches the file name root string to the hardcopy device is output.
- After a form feed is sent to the hardcopy device, the printing/plotting process begins.

Running the Program

This program allows you to plot or print as many as four HP 8702D Option 011 display dumps (stored on a LIF formatted disk) on one side of a single sheet of paper. Refer to the instructions detailed in the chapter "Printing, Plotting, and Saving Measurement Results," in the *HP 8702D User's Guide.*

```
10  ! This example program reads and plots files which have been stored on
20  ! a LIF formatted disk by the analyzer. The plots are sent to either a
30  ! plotter with auto-feed capability, such as the HP 7550B, or an HP-GL/2
40  ! compatible printer, such as the LaserJet 4 series (monochrome) or the
50  ! DeskJet 1200C (color).
60  !
70  ! Sending plots to disk is discussed in the Printing, Plotting and Saving
80  ! Measurement Results section of the analyzer User's Guide. The file
90  ! naming conventions are discussed in this section and will provide more
100 ! details.
110 !
120 ! This program example will provide form feeds to separate the plots. If
130 ! multiple plots per page have been stored, this program will generate
140 ! those plots.
150 ! A file naming convention has been devised to allow the program
160 ! to initialize the printer for HP-GL/2 at the beginning of a
170 ! page, to plot multiple plots to the same page, if desired, and
180 ! when all plots to the same page have been completed then send a
190 ! page eject (form feed) to the hardcopy device.
200 !
210 ! The plot file name is made up of four parts, the first three of which
220 ! are generated automatically by the analyzer whenever a plot is requested:
230 ! The prefix, “PLOT”, a two digit sequence number, in the range of 00 to
240 ! 31, a two letter output format code to indicate the plot quadrant
250 ! position or full page, LU (Left Upper), LL (Left Lower), RU (Right
260 ! Upper), RL (Right Lower) or FP (Full Page). For example, the first
270 ! full page plot to a disk would be named “PLOT00FP”.
280 !
290 ! a. Build hardcopy device control strings.
300 ! b. Define output hardcopy device.
310 ! c. Catalog disk storing the file names which match file name specifier
320 !    in the file name array, Flnm$, and setting the number of files that
330 !    match.
340 ! d. Dimension Flnm$ for the number of files matched.
350 ! e. Sort the file name array.
360 ! f. Form feed the hardcopy device.
370 ! g. Process each of the files in the file name array.
380 !    1. Define the current file name root string.
390 !    2. If hardcopy device is a printer then output HP-GL initialization
400 !       string.
410 !    3. Output each file which matches the file name root string to the
420 !       hardcopy device.
430 !    4. Output a form feed to the hardcopy device.
440 !
450 ! This program requires HP BASIC 6.0 or greater which provides the use of
460 ! wild cards with the catalog command.
470 !
480 ! The peripheral HP-IB addresses and the hard copy device selection are
490 ! hard coded and may need to be changed for different systems
```

```
500 ! configurations. See lines 1130 to 1240.
510 !
520 ! EXAMP7B2
530 !
540 WILDCARDS UX;ESCAPE “\”             ! Enable HP-UX style wild cards
550 OPTION BASE 1                       ! Set numeric arrays to start at 1
560 DIM Flnm$(1:200)[14]                ! Plot filename array
570 DIM Hpglinit$[80]                   ! Printer HPGL initialization string
580 DIM Srch$[60]                       ! Search string for plot filenames
590 DIM Esc_chr$[1]                     ! Escape character ASCII 27
600 INTEGER Plt_arry1(1:32767)          ! Plotter command array
610 INTEGER Plt_arry2(1:2,1:32767)      ! Additional plot arrays if required
620 INTEGER Plttr                       ! Plotter for output
630 INTEGER Prntr                       ! Printer for output
640 INTEGER Outputdvc                   ! Output device selected
650 INTEGER Root_mtch                   ! Root plot file flag
660 INTEGER Flnm_idx                    ! Pointer to filename array
670 INTEGER Nbr_files                   ! Number of files which are plot files
680 INTEGER Prfx_lngth                  ! Length of prefix defined in filename
690 INTEGER Root_lngth                  ! Length of root name in file name
700 INTEGER Arry1_sz                    ! Number of data works in plot file
710 INTEGER Arry2_sz                    ! Number of arrays in plot file
720 INTEGER Plttr_addr                  ! Plotter address
730 INTEGER Prntr_addr                  ! Printer address
740 REAL Rcrd_lngth                     ! Record length in plot file
750 REAL Nmbr_rcrds                     ! Number of records in plot file
760 REAL Nmbr_wrds                      ! Number of data words in plot file
770  !
780 Esc_chr$=CHR$(27)           ! Escape character (1B hex) ASCII 27 (Decimal)
790  !
800  !   ***   Build control string for printers   ***
810  !
820  ! Build hardcopy device control string containing setup commands for
830  ! printer output.
840  ! Reset, conditional page eject
850 Hpglinit$=Esc_chr$&”E”
860  ! Page size A  8.5 x 11
870 Hpglinit$=Hpglinit$&Esc_chr$&”&12A”
880  ! Landscape orientation
890 Hpglinit$=Hpglinit$&Esc_chr$&”&l1O”
900  ! No left margin
910 Hpglinit$=Hpglinit$&Esc_chr$&”&a0L”
920  ! No right margin
930 Hpglinit$=Hpglinit$&Esc_chr$&”&a400M”
940  ! No top margin
950 Hpglinit$=Hpglinit$&Esc_chr$&”&l0E”
960  ! Picture frame size 10.66 inches x 7.847 inches
970  ! (720 decipoints per inch)
980 Hpglinit$=Hpglinit$&Esc_chr$&”*c7680x5650Y”
990  ! Move cursor to anchor point
1000 Hpglinit$=Hpglinit$&Esc_chr$&”*p50x50Y”
1010 ! Set picture frame anchor point
1020 Hpglinit$=Hpglinit$&Esc_chr$&”*c0T”
1030 ! Set CMY palette
1040 Hpglinit$=Hpglinit$&Esc_chr$&”*r-3U”
1050 ! Enter HPGL mode with the cursor (pen) at the PCL current save position
```

```
1060 !
1070 ! Exit HPGL mode to accept printer command
1080 Hpgl_exit$=Esc_chr$&"%0A"
1090 !
1100 ! Conditional form feed (page eject)
1110 Form_feed$=Esc_chr$&"E"
1120 !
1130 !   ***   Initialize varibles and assign output device   ***
1140 !
1150 ! Define device selection flags to determine plotter or printer
1160 ! Select device with 1 and set Outputdvc to define it
1170 Plttr=1                        ! Select plotter for output
1180 Prntr=0                        ! Select printer for output
1190 Outputdvc=Plttr                ! define output device as plotter
1200 !
1210 False=0                        ! Define flags for logic tests
1220 True=1
1230 !
1240 Prfx$="PLOT"                   ! define plot file name prefix string
1250 !
1260 !   ***   Initialize HP-IB device addresses   ***
1270 !
1280 Prntr_addr=701                ! Printer HP-IB address
1290 Plttr_addr=705                ! Plotter HP-IB address
1300 !
1310 ! Set address of flexible disk drive containing plot files
1320 Msi$=":,1400"
1330 !
1340 ! Define I/O paths for plot output with no formatting on data
1350 IF Outputdvc=Plttr THEN        ! select output device for plotting
1360  ASSIGN @Prntpltdvc TO Plttr_addr;FORMAT OFF    ! Select plotter
1370 ELSE
1380  ASSIGN @Prntpltdvc TO Prntr_slctr;FORMAT OFF   ! Select printer
1390 END IF
1400 !
1410 !   ***  Search disk for plot files  ***
1420 !
1430 ! Define the plot file name specifier:  Prefix (Prfx$), two
1440 ! sequence digits([0-9][0-9]), two output format specifier
1450 ! characters ([FLR][LPU]) and an optional character (s)
1460 Srch$=Prfx$&"[0-9][0-9][FLR][LPU]*"
1470 !
1480 ! Catalog the files that match the plot file name specifier by
1490 ! putting the names in the string array Flnm$ and the number of files
1500 ! in the integer Nbr_files. Suppress the catalog header text.
1510 CAT Srch$&Msi$ TO Flnm$(*);COUNT Nbr_files,NO HEADER,NAMES
1520 !
1530 ! If no files are found then print message and stop
1540 IF Nbr_files=0 THEN
1550  PRINT "No files found;  program terminated"
1560  STOP
1570 END IF
1580 !
1590 !   ***   Sort the plot file names found   ***
1600 !
1610 ! Re-dimension the file name array to the actual number of files that
```

```
1620 ! were found.
1630 REDIM Flnm$(1:Nbr_files)
1640 !
1650 ! Sort the file names into alphabetical order
1660 MAT SORT Flnm$(*)
1670 !
1680 GOSUB Frm_fd                     ! Send a form feed to hardcopy device
1690 PRINT
1700 !
1710 !   ***   Cycle through the filenames and plot the data   ***
1720 !
1730 Flnm_idx=1                       ! Initialize the Flnm$ array index
1740 Prfx_lngth=LEN(Prfx$)            ! Find the length of the Prfx$ string
1750 !
1760 ! Process each of the files in the Flnm$ array
1770 WHILE Flnm_idx<=Nbr_files
1780   ! Define Root$ = file name prefix string plus the two sequence digits
1790   Root$=Flnm$(Flnm_idx)[1;Prfx_lngth+2]
1800   ! Find length of the Root$
1810   Root_lngth=LEN(Root$)
1820   ! If the two output specifier characters are "FP" (full page) then
1830   ! include them as part of the Root$
1840   IF Flnm$(Flnm_idx)[Root_lngth+1;2]="FP" THEN
1850     Root$=Root$&"FP"
1860     Root_lngth=Root_lngth+2
1870   END IF
1880   !
1890   Root_mtch=True             ! initialize file matches Root$ flag
1900   ! If the ouput device is a printer send HP-GL initiliazation string
1910   IF Outputdvc=Prntr THEN
1920     OUTPUT @Prntpltdvc USING "#,K";Hpglinit$
1930   END IF
1940   !
1950   !   ***   Plot files on the same page   ***
1960   !
1970   ! While the root portion of the file names match Root$, output the
1980   ! plot files to the same page.
1990   WHILE Root_mtch=True
2000     ! Print the name of the plot file
2010     PRINT Flnm$(Flnm_idx),
2020     ! Output the plot file to the hardcopy device
2030     GOSUB Otpt_fl
2040     ! Increment Flnm$ array index
2050     Flnm_idx=Flnm_idx+1
2060     ! If all plot files have been output or the plot file name does not
2070     ! match Root$ then set Root_Mtch=False to end plotting of the same
2080     ! page.
2090     IF Flnm_idx>Nbr_files THEN
2100       Root_mtch=False
2110     ELSE
2120       IF Root$<>Flnm$(Flnm_idx)[1;Root_lngth] THEN Root_mtch=False
2130     END IF
2140   END WHILE              ! Loop to plot to the same page
2150   !
2160   PRINT
2170   ! Output form feed to output device to eject page
```

```
2180   GOSUB Frm_fd
2190 END WHILE                ! Loop to plot next page
2200  !
2210 STOP
2220 !
2230 !***************************Subroutines ******************************
2240 !
2250 Otpt_fl:! Read the file into an array(s) and then output the array(s)
2260 ! to the hardcopy device.
2270 !
2280 ! Open and read the plot file size
2290 ASSIGN @Ldisc TO Flnm$(Flnm_idx)&Msi$
2300 !
2310 ! Get number of records in file from I/O path status registers
2320 STATUS @Ldisc,3;Nmbr_rcrds
2330 !
2340 ! Get record length
2350 STATUS @Ldisc,4;Rcrd_lngth
2360 !
2370 ! Compute the number of words of data in the file
2380 Nmbr_wrds=Nmbr_rcrds*Rcrd_lngth/2
2390 !
2400 ! Determine the number of arrays necessary to hold the plot file data.
2410 ! The maximum size of an RMB array is 32767. If the number of words of
2420 ! data is greater than 32767 then multiple arrays must be used to hold
2430 ! all of the data.
2440 !
2450 ! Compute the dimensions of Plt_arry1 and Plt_arry2 that are required
2460 ! to hold the plot data.
2470 Arry1_sz=Nmbr_wrds MOD 32767
2480 Arry2_sz=INT(Nmbr_wrds/32767)
2490 !
2500 ! Re-dimension Plt_arry1 to the correct size
2510 REDIM Plt_arry1(1:Arry1_sz)
2520 !
2530 ! If the number of words of data is less than 32767 then use one array
2540 IF Arry2_sz=0 THEN
2550   ENTER @Ldisc;Plt_arry1(*)            ! Read the plot data from the file
2560 ELSE
2570   ! Use 2 arrays to read data
2580   ENTER @Ldisc;Plt_arry2(*),Plt_arry1(*)  ! Read the plot data from file
2590 END IF
2600 !
2610 ASSIGN @Ldisc TO *                        ! Close plot file
2620 !
2630 ! Output the data to the hardcopy device
2640 IF Arry2_sz=0 THEN                        ! Only one array <32767 words
2650   OUTPUT @Prntpltdvc;Plt_arry1(*)
2660 ELSE                                     ! Data > 32767 so send 2 arrays
2670   OUTPUT @Prntpltdvc;Plt_arry2(*),Plt_arry1(*)
2680 END IF
2690 RETURN
2700  !
2710  !************************************************************************
2720  !
2730 Frm_fd:! Send a form feed (page eject) command to the hardcopy device
```

```
2740 IF Outputdvc=Plttr THEN                    ! Plotter output
2750   OUTPUT @Prntpltdvc USING “#,K”;”PG;”
2760 ELSE                                       ! For printer output. The
2770   ! printer first must exit HP-GL mode before sending form feed command
2780   OUTPUT @Prntpltdvc USING “#,K”;Hpgl_exit$&Form_feed$
2790 END IF
2800 RETURN
2810  !
2820 END
```

Reading ASCII Instrument Files

File Name EXAMP7C.BAS

Description Another approach to accessing the analyzer's test results is to store the data onto a disk file from the analyzer. This operation generates an ASCII file of the analyzer data in a CITIFILE format. A typical file is generated by the next example and is shown below:

```
CITIFILE A.01.00
#NA VERSION HP8702D.06.09
NAME DATA
VAR FREQ MAG 11
DATA S[1,1] RI
SEG_LIST_BEGIN
SEG 100000000 200000000 11
SEG_LIST_END
BEGIN
8.30566E-1,-1.36749E-1
8.27392E-1,-1.43676E-1
8.26080E-1,-1.52069E-1
8.25653E-1,-1.60003E-1
8.26385E-1,-1.68029E-1
8.26507E-1,-1.77154E-1
8.26263E-1,-1.87316E-1
8.26721E-1,-1.97265E-1
8.2724E-1,-2.07611E-1
8.28552E-1,-2.19940E-1
8.29620E-1,-2.31109E-1
END
```

This data file is stored by the analyzer under remote control or manually from the front panel. Refer to "Saving Data, States, and the Display" in the *HP 8702D User's Guide* for more details on manual operation. This program performs the same steps that are required to manually store a file from the front panel.

This program stores a file in the same manner as an operator would store a file onto the analyzer's internal disk drive from the front panel.

This example explains the process of storing the data from the analyzer to a file on the internal disk drive. There is also a program to read the data from the file into a data array for further processing or reformatting to another file type. The internal drive will store in the same format that is present on the disk. A new disk may be formatted in either LIF or DOS. For the example, the assumption has been made that the format transformation has already taken place, and there is a file that can be read record by record, from which data can be retrieved.

The goal of this example is to recover an array of stimulus frequency along with the trace-data values. CITIFILES contain the real and imaginary values of each data point. Some further transformation will be required to obtain magnitude values, for example.

This file contains more information than will be used in this example. The file is accessed and the records read from the file and printed on the controller display to observe the actual file contents. The file pointer is reset and the records are then read and interpreted for their data contents.

The first six records are skipped for this example. The seventh record contains the stimulus-frequency values and the number of points in the trace. These values are read from the record. The frequency increment, or point spacing, is calculated and used later in the frequency-data calculations for a point. Two more records are skipped and the next is the first record representing data values. The data values are read using a loop until the values for the number of points have been recovered from the file. The data values are tabulated and printed out on the controller display.

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

- An I/O path is assigned to the analyzer.
- The system is initialized.
- A string is dimensioned to hold a file record.
- The analyzer operating state is set.
- The internal drive is selected for storage (only ASCII data is stored).
- A file name is entered and the data stored into it.
- The operator is prompted to move the disk to the controller disk drive.
- The disk file is read and the contents displayed.
- The file pointer is rewound.
- The file contents are converted to trace data.
- The frequency and complex-data pair is displayed for each point.
- The analyzer is restored to continuous-sweep mode.
- The analyzer is returned to local control and the program ends.

Running the Program

Remove the disk from the analyzer's disk file and place it into the controller disk drive.

The contents of this disk file are shown at the beginning of this section. The file lines are read and then printed on the controller display. The file pointer is reset and the lines are interpreted for their data contents.

The first 6 lines are skipped for this example. The seventh line contains the stimulus frequency values and the number of points in the trace. These values are read from the line. The frequency increment, or point spacing, is calculated and used later to calculate the frequency data for a point. Two more lines are skipped. The next line is the first of the data values. The data values are read in a loop until the number of points have been recovered from the file. The data values are tabulated and printed out on the controller display.

```
10  ! This program shows how to store an ASCII data file in CITIFILE format
20  ! and retrieve the data with the controller. The disk is written in the
30  ! analyzer system and then moved to the controller disk and the data
40  ! accessed.
50  !
60  ! EXAMP7C
70  !
80  ASSIGN @Nwa TO 716                   ! Assign an I/O path for the analyzer
90  !
100 CLEAR SCREEN
110 ABORT 7                              ! Generate an IFC (Interface Clear)
120 CLEAR @Nwa                           ! SDC (Selected Device Clear)
130 OUTPUT @Nwa;"OPC?;PRES;"             ! Preset the analyzer and wait
140 ENTER @Nwa;Reply                     ! Read the 1 when complete
150 !
160 DIM Record$[80]                      ! String to read the disk records
170 !
180 ! Set up analyzer
190 OUTPUT @Nwa;"STAR100MHZ;"            ! Start frequency 100 MHz
200 OUTPUT @Nwa;"STOP 200MHZ"            ! Stop frequency 200 MHz
210 OUTPUT @Nwa;"POIN11;"                ! Trace length 11 points
220 OUTPUT @Nwa;"OPC?;SING;"             ! Single sweep and wait
230 ENTER @Nwa;Reply                     ! Read in the 1 when complete
240 !
250 ! Program disk storage operation
260 !
270 OUTPUT @Nwa;"INTD;"                  ! Select internal disk file
280 OUTPUT @Nwa;"EXTMFORMON;"            ! Store formated data
290 OUTPUT @Nwa;"EXTMDATOON;"            ! Store data file only
300 INPUT "Enter data file name (5 chars)",File_name$ ! Get file name
310 File_name$=UPC$(File_name$)          ! File names are uppercase
320 OUTPUT @Nwa;"TITF1""";File_name$;""";" ! Title for save reg 1
330 OUTPUT @Nwa;"SAVUASCI;"              ! Save as ASCII file
340 !
350 OUTPUT @Nwa;"STOR1;"                 ! Store data to disk file
360 OUTPUT @Nwa;"OPC?;WAIT;"             ! Wait until store is complete
370 ENTER @Nwa;Reply
380 !
390 ! File storage is complete
400 !
410 INPUT "Place disk in controller disk drive, then press Return",A$
420 !
430 ! Read data file information
440 !
450 ASSIGN @File TO File_name$&"D1:,1400"  ! Open an I/O path for file
460 Record_cnt=1                           ! Counter to count records
470 !
480 PRINT CHR$(12);                        ! Formfeed to clear display
490 PRINT "Contents of data file"          ! Show contents of the data file
500 Readfile:   !
510 ON END @File GOTO End_file             ! Test for end of file and exit
520 ENTER @File;Record$                    ! Read ASCII record
530 PRINT Record_cnt,Record$               ! print record on display
540 Record_cnt=Record_cnt+1                ! Increment record counter
550 GOTO Readfile                         ! Read next record
560 !
```

```
570 End_file: !                             ! Reached the end of file
580 PRINT “End of File. “;Record_cnt-1;” Records found”
590 INPUT “Press Return to continue”,A$
600 PRINT CHR$(12);                         ! Formfeed to clear display
610 !
620 ! Read file data into arrays
630 !
640 RESET @File                             ! Rewind file pointer to beginning
650 FOR I=1 TO 6
660   ENTER @File;Record$                   ! Skip first six records
670 NEXT I
680 ENTER @File;Record$                     ! Read frequency data record
690 Record$=Record$[POS(Record$,” “)+1]  ! skip SEG to first space + 1
700 Startf=VAL(Record$)                     ! Read start frequency
710 Record$=Record$[POS(Record$,” “)+1]  ! Skip to next space + 1
720 Stopf=VAL(Record$)                      ! Read stop frequency
730 Record$=Record$[POS(Record$,” “)+1]  ! Skip to next space +1
740 Num_points=VAL(Record$)                 ! Read the number of points
750 PRINT “ Number of points in file “;Num_points
760 PRINT                                   ! White space
770 !
780 Freq_inc=(Stopf-Startf)/(Num_points-1) ! Compute frequency increment
790 !
800 ALLOCATE Array(Num_points,2)            ! Allocate array from Num_points
810 ENTER @File;Record$                     ! Skip SEG_LIST_END record
820 ENTER @File;Record$                     ! Skip BEGIN record
830 !
840 ! Read in the data array
850 PRINT “Freq (MHz)  Data 1    Data 2”  ! Table header for data array
860 FOR I=1 TO Num_points                   ! Read in array entries
870   ENTER @File;Record$                   ! Read in the record of 2 entries
880   !
890   Array(I,1)=VAL(Record$)               ! Read first data value
900   Data$=Record$[POS(Record$,”,”)+1]  ! Skip to comma and next value
910   Array(I,2)=VAL(Data$)                 ! Read second data value
920   !
930   Freq=Startf+(Freq_inc*(I-1))          ! Compute stimulus value for array
940   Freq=Freq/1.E+6                       ! Convert frequency to MHz
950   !
960   PRINT Freq,Array(I,1),Array(I,2)      ! Print data array values
970 NEXT I                                  ! Read next array data points
980 !
990  OUTPUT @Nwa;”CONT;”                    ! Restore continuous sweep
1000 OUTPUT @Nwa;”OPC?;WAIT;”               ! Wait for analyzer to finish
1010 ENTER @Nwa;Reply                       ! Read the 1 when complete
1020 LOCAL @Nwa                             ! Release HP-IB control
1030 END
```

3

Language Reference

Language Reference

This chapter is the reference for all HP 8702D programming commands. Commands are listed alphabetically.

Table 3-1. Notation Conventions and Definitions

Convention	Description
< >	Angle brackets indicate values entered by the programmer.
|	"Or" indicates a choice of one element from a list.
[ ]	Square brackets indicate that the enclosed items are optional.
{}	When several items are enclosed by braces, one, and only one of these elements must be selected.
<number>	A numerical operand.
<integer>	An ASCII string representing an integer. This is defined by the IEEE 488.2 <NR1> format.
<string>	A character-string operand which must be enclosed by quotes.

AB

Measures and displays A/B on the active channel.

Syntax `AB`

ADDR

Sets the HP-IB address for the specified device.

Syntax `ADDR{CONT|DISC|PERI|PLOT|POWM|PRIN}[?|` <*number*>]

<*number*> can range from 0 to 30.

Option	Description
`CONT`	Controller HP-IB address. The address where control is returned after a pass control.
`DISC`	Disk HP-IB address.
`PERI`	Peripheral HP-IB address.
`PLOT`	Plotter HP-IB address.
`POWM`	Power meter HP-IB address.
`PRIN`	Printer HP-IB address.

Query Response <*number*>

ALTAB

Places the analyzer in the alternate inputs measurement mode, where inputs A and B are measured on alternate sweeps.

Syntax `ALTAB`

See Also `CHOPAB`

ANAB

Enables the analog bus for service use.

Syntax `ANAB{?|ON|OFF}`

Query Response {`1|0`}

Example `ANABON`

ANAI

Measures and displays the data at the auxiliary input (ANALOG IN).

Syntax `ANAI[?|` <*number*>`]`

<*number*> can range from 1 to 31.

This parameter can only be applied when ANAB is set to on. Do not use the <*number*> parameter when ANAB is OFF.

Query Response <*number*>

AR

Measures and displays A/R on the active channel.

Syntax `AR`

ASEG

Uses all segments for list frequency sweep.

Syntax `ASEG`

ASSS

Asserts the sequence status bit.

Syntax `ASSS`

ATTP

Selects the amount of attenuation at PORT 1 or 2.

Syntax `ATTP{1|2}`[`?|` <*number*>]

<*number*> is 0, 10, 20,..., 70 dB.

Query Response <*number*>

AUTO

Autoscales the active channel.

Syntax	`AUTO`

AVERFACT

Sets the averaging factor on the active channel.

Syntax	`AVERFACT[?| <number>]` *<number>* can range from 0 to 999.
Query Response	*<number>*

AVERO

Turns averaging on and off on the active channel.

Syntax	`AVERO{?|ON|OFF}`
Example	`AVEROON`
Query Response	`{1|0}`

AVERREST

Restarts the averaging on the active channel.

Syntax	`AVERREST`

BACI

Sets the background intensity of the display.

Syntax

`BACI`[`?`| <*number*>]

<*number*> can range from 0 to 100.

Query Response

<*number*>

BANDPASS

Selects the time domain bandpass mode.

Syntax

`BANDPASS`

BEEP

Controls the warning beeper.

Syntax `BEEP{DONE|FAIL|WARN}{?|ON|OFF}`

Option	Description
`DONE`	Controls the warning beeper for completion of functions such as save, done with calibration standard, and data trace saved.
`FAIL`	Controls the warning beeper for a limit test failure.
`WARN`	Controls the warning beeper for the generation of a warning message.

Query Response `{1|0}`

Example `BEEPDONEON`

BR

Measures and displays B/R on the active channel.

Syntax `BR`

C

Sets the open capacitance values of an open circuit while it is being defined as a calibration standard.

Syntax `C{0|1|2|3} [?|` *<number>*`]`

`Where` *<number>* value can be:

Option	Description	Number Value
`0`	±10k	10–15 F
`1`	± 10k	10–27 F/Hz
`2`	±10k	10–36 F/Hz^2
`3`	±10k	10–45 F/Hz^3

Query Response *<number>*

CAL1

Accepted for compatibility with the HP 8510A, where its function is to begin a calibration sequence.

Syntax `CAL1`

CALFCALF

Sets the power meter calibration factor corrections for the particular sensor used. Sensor B is only valid for the HP 438A, which has two input channels.

Syntax

`CALFCALF[?|` <*number*>]

<*number*> can range from 0 to 200%.

Query Response

<*number*>

CALFFREQ

Selects the frequency for the calibration factor correction.

Syntax

`CALFFREQ[?|` <*number*>]

Where <*number*> can range from:

Condition	Range
Frequency sweeps	30 kHz to 3 GHz (to 6 GHz for option 006)
Power sweeps	–15 to 20 dBm in range 0, 25 dB maximum in other ranges (–100 to +100 with power meter cal on)
CW time	0 to 24 hours
Frequency sweep (transform on)	±1/frequency step
CW time sweep (transform on)	±1/time step

Query Response

<*number*>

CALFSEN

Edits the sensor A or B calibration factor table.

Syntax `CALFSEN{A|B}`

CALI

Begins a calibration sequence.

Syntax `CALI{EORM|FUL2|OERM|ONE2|RAI|RESP|S111|S221|TRL2}`

Option	Description
`EORM`	Begins a response and match measurement calibration for E/O devices.
`FUL2`	Begins a calibration sequence for full 2-port measurement calibration.
`OERM`	Begins a response and match measurement calibration for O/E devices.
`ONE2`	Begins a calibration sequence for one-path 2-port.
`RAI`	Begins a calibration sequence for response and isolation.
`RESP`	Begins a calibration sequence for response.
`S111`	Begins a calibration sequence for S11 1-port.
`S221`	Begins a calibration sequence for S22 1-port.
`TRL2`	Begins a calibration sequence for thru, reflect, line or Line, reflect, match (TRL*/LRM*) 2-port.

Example `CALIS111`

CALK

Selects a calibration kit.

Syntax `CALK{35MD|35MM|7MM|N50|N75|USED}`

Option	Description
`35MD`	Selects a calibration kit for 3.5 mm (HP 85033D).
`35MM`	Selects a calibration kit for 3.5 mm (HP 85033C).
`7MM`	Selects a calibration kit for 7 mm.
`N50`	Selects a calibration kit for type-N 50 ohm.
`N75`	Selects a calibration kit for type-N 75 ohm.
`USED`	Selects a user-defined calibration kit.

Example `CALKUSED`

CALN

Turns calibration type to off.

Syntax `CALN`

CBRI

Adjusts the color brightness of the selected display feature.

Syntax `CBRI[?| <`*number*`>]`

<*number*> can range from 0 to 100.

Query Response <*number*>

CENT

Sets the center stimulus value. If a list frequency segment is being edited, sets the center of the list segment.

Syntax `CENT[?|` <*number*>]

<*number*> can range from:

Condition	Range
Frequency sweeps	30 kHz to 3 GHz (to 6 GHz for option 006)
Power sweeps	–15 to 20 dBm in range 0, 25 dB maximum in other ranges (–100 to +100 with power meter cal on)
CW time	0 to 24 hours
Frequency sweep (transform on)	±1/frequency step
CW time sweep (transform on)	±1/time step

Query Response <*number*>

CHAN

Makes channel 1 or 2 the active channel. OPC compatible.

Syntax `CHAN{1|2}`

Example `CHAN2`

CHOPAB

Places the analyzer in the chop measurement mode. As opposed to `ALTAB`.

Syntax `CHOPAB`

CLAD

Class done, modify cal kit, specify class.

Syntax `CLAD`

CLASS

Calls reflection standard classes during a calibration sequence.

Syntax `CLASS{11A|11B|11C|22A|22B|22C}`

Description If only one standard is in the class, it is measured. If there is more than one, the standard being used must be selected with `STAN{A|B|C|D|E|F|G}`. If there is only one standard in the class, these commands are OPC compatible.

Option	Description
`11A`	S11 1-port, opens.
`11B`	S11 1-port, shorts.
`11C`	S11 1-port, loads.
`22A`	S22 1-port, opens.
`22B`	S22 1-port, shorts.
`22C`	S22 1-port, loads.

Example `CLASS11A`

CLEA

Clears the indicated save/recall registers.

Syntax `CLEA{1|2|3|4|5}`

Example `CLEA1`

CLEABIT

Clears the specified bit on the GPIO.

Syntax `CLEABIT[?| <number>]`

<number> can range from 0 to 7.

Query Response *<number>*

CLEAL

Clears the limit line list. Should be preceded by `EDITLIML`.

Syntax `CLEAL`

CLEARALL

Clears all the save/recall registers. OPC compatible.

Syntax `CLEARALL`

CLEAREG

Clears save/recall registers 01 through 31. `CLEAREG01` through `CLEAREG05` are the same as `CLEA1` through `CLEA5`.

Syntax `CLEAREG[<integer>]`

<integer> can range from 01 to 31.

Example `CLEAREG01`

CLEASEQ

Clears the sequence from the internal registers.

Syntax `CLEASEQ{1|2|3|4|5|6}`

Example `CLEASEQ1`

CLEL

Clears the desired list.

Syntax `CLEL`

Description This could be a frequency list, power loss list, cal sensor list, or limit test list. Should be preceded by `EDITLIML`, `EDITLIST`, `POWLLIST`, `CALFSENA`, or `CALFSENB`.

CLES

Clears the status register, the event-status registers, and the enable registers.

Syntax `CLES`

CLS

Clears the status register.

Syntax `CLS`

COAX

Selects coaxial offsets instead of waveguide while defining a standard during a cal kit modification.

Syntax `COAX`

COEF

Enters the indicated coefficient for the cal standard. This command follows `STDTRECE` or `STDTSOUR` to specify whether it is a source or receiver coefficient.

Syntax `COEF{A|B|C|D|E|F|G|H|I}`

Example `COEFA`

COLO

Selects a display function for color modification.

Syntax `COLO{CH1D|CH1M|CH2D|CH2M|GRAT|TEXT|WARN}`

Option	Description
`CH1D`	Selects the display feature for color modification for channel 1 data and limit lines.
`CH1M`	Selects the display feature for color modification for channel 1 memory.
`CH2D`	Selects the display feature for color modification for channel 2 data and limit lines.
`CH2M`	Selects the display feature for color modification for channel 2 memory.
`GRAT`	Selects the display feature for color modification for graticule.
`TEXT`	Selects the display feature for color modification for text.
`WARN`	Selects the display feature for color modification for warning.

Example `COLOWARN`

COLOR

Adjusts the color saturation for the selected display feature.

Syntax `COLOR`[`?|` <*number*>]

<*number*> can range from 0 to 100.

Query Response <*number*>

CONS

Continues the paused sequence.

Syntax `CONS`

CONT

Sets continuous sweep trigger mode.

Syntax `CONT`

CONV

Converts the S-parameter data.

Syntax `CONV{1DS|OFF|YREF|YTRA|ZREF|ZTRA}`

Option	Description
`1DS`	Converts the S-parameter data to inverted.
`OFF`	Converts the S-parameter data to OFF.
`YREF`	Converts the S-parameter data to Y:reflection.
`YTRA`	Converts the S-parameter data to Y:transmission.
`ZREF`	Converts the S-parameter data to Z:reflection.
`ZTRA`	Converts the S-parameter data to Z:transmission.

Example `CONVZTRA`

COPYFRFT

Copies the file titles into the register titles.

Syntax `COPYFRFT`

COPYFRRT

Copies save/recall register titles to the disk register titles.

Syntax `COPYFRRT`

CORI

Turns interpolative error correction on and off.

Syntax `CORI{?|ON|OFF}`

Query Response {`1|0`}

CORR

Turns error correction on and off.

Syntax `CORR{?|ON|OFF}`

Query Response {`1|0`}

COUC

Couples and uncouples the stimulus between the channels.

Syntax `COUC{?|ON|OFF}`

Query Response `{1|0}`

COUP

Couples the power when coupled channels is turned OFF, `COUCOFF`.

Syntax `COUP{?|ON|OFF}`

Query Response `{1|0}`

CSWI

Selects test set continuous switching (ON) or test set hold (OFF) when there is a 2-port calibration active.

Syntax `CSWI{?|ON|OFF}`

Description Continuous switching is allowed only when the power ranges on both attenuator ports are set the same. When continuous switching is ON, the analyzer measures all four S-parameters each time before displaying the data for a full 2-port cal measurement. In test set hold mode, the analyzer measures all four S-parameters once and then measures the desired parameter continuously. This is known as a quasi 2-port cal measurement and it is less accurate than a full 2-port calibrated measurement.

Query Response `{1|0}`

CWFREQ

Sets the CW frequency for power sweep and CW frequency modes.

Syntax

`CWFREQ` [`?|` <*number*>]

<*number*> may be values:

Condition	Range
Frequency sweeps	30 kHz to 3 GHz (to 6 GHz for option 006)
Power sweeps	–15 to 20 dBm in range 0, 25 dB maximum in other ranges (–100 to +100 with power meter cal on)
CW time	0 to 24 hours
Frequency sweep (transform on)	±1/frequency step
CW time sweep (transform on)	±1/time step

Description

While the list frequency table segment is being edited, it sets the center frequency of the current segment.

Query Response

<*number*>

CWTIME

Selects the CW time sweep type.

Syntax

`CWTIME`

D1DIVD2

Divides the data in channel 1 by the data in channel 2 and displays the result on channel 2.

Syntax `D1DIVD2{?|ON|OFF}`

Query Response `{1|0}`

DATI

Stores trace in channel memory. OPC compatible.

Syntax `DATI`

DCONV

Selects down converter for mixer measurements.

Syntax `DCONV`

DEBU

Turns the HP-IB debug mode on and off.

Syntax `DEBU{?|ON|OFF}`

Description When ON, option 011 scrolls incoming HP-IB commands across the display.

Query Response `{1|0}`

DECRLOOC

Decrements the sequencing loop counter by 1. `NEWSEQ` must precede to ensure that a sequence is currently being created or modified.

Syntax `DECRLOOC`

DEFA

Sets the standards to default definitions.

Syntax `DEFA{OPTS|RCOE|SCOE}`

Option	Description
`OPTS`	Optical standards.
`RCOE`	Receiver coefficients.
`SCOE`	Source coefficients.

Example `DEFAOPTS`

DEFC

Sets the default colors for all display features.

Syntax `DEFC`

DEFLPRINT

Sets the printer to the following default setup conditions:

Table 3-2. Default Printer Settings

Parameter	Setting
Print	Monochrome
Auto-feed	On
Print Colors:	
Ch 1 Data	Magenta
Ch 1 Memory	Green
Ch 2 Data	Blue
Ch 2 Memory	Red
Graticule	Cyan
Warning	Black
Text	Black

Syntax

```
DEFLPRINT
```

DEFLTCPIO

Sets up the following default state for copy. There is no equivalent front-panel key.

Table 3-3. Plotter Settings

Parameter	Setting
Plotter Type	PLOTTER
Plotter Port	SERIAL
Baud Rate	9600
Handshake	Xon-Xoff
HP-IB Address	5

Table 3-4. Printer Settings

Parameter	Setting
Printer Type	DESKJET
Printer Port	PARALLEL
Baud Rate	19200
Handshake	Xon-Xoff
HP-IB Address	1
Parallel Port	COPY

Syntax

```
DEFLTCPIO
```

DEFS

Begins standard definition during cal kit modification.

Syntax `DEFS[?|` <*number*>]

<*number*> is the standard number.

Query Response <*number*>

DELA

Displays the data formatted as group delay.

Syntax `DELA`

DELO

Turns the delta marker mode OFF.

Syntax `DELO`

DELR

Makes the indicated marker the delta reference.

Syntax `DELR{1|2|3|4}`

Example `DELR1`

DELRFIXM

Makes the fixed marker the delta reference.

Syntax `DELRFIXM`

DEMO

Sets the transform demodulation.

Syntax `DEMO{AMPL|OFF|PHAS}`

Option	Description
`AMPL`	Sets the transform demodulation to amplitude demodulation. Only has an effect with a CW time transform.
`OFF`	Turns the transform demodulation function OFF.
`PHAS`	Sets the transform demodulation to phase demodulation.

Example `DEMOAMPL`

DFLT

Sets the plotter to the default setup conditions. (Refer to “Default Plotter Settings” on page 3-30.)

Syntax `DFLT`

Table 3-5. Default Plotter Settings

Item	Setting
Plot Data	On
Plot Mem	On
Plot Grat	On
Plot Text	On
Plot Mkr	On
Auto-feed	On
Scale Plot	Full
Plot Speed	Fast
Channel 1	
Pen number for data	2
Pen number for memory	5
Pen number for graticule	1
Pen number for text	7
Pen number for marker	7
Channel 2	
Pen number for data	3
Pen number for memory	6
Pen number for graticule	1
Pen number for text	7
Pen number for marker	7
Line Type for Data	7
Line Type for Memory	7

DIRS

Sets the number of files in the directory at disk initialization.

Syntax `DIRS[?|` <*number*>]

<*number*> can range from 8 to 8192.

Query Response <*number*>

DISCUNIT

Specifies which disk in a multiple-disk drive is to be used for disk registers.

Syntax `DISCUNIT[?|` <*number*>]

<*number*> can range from 0 to 30.

Query Response <*number*>

DISCVOLU

Specifies which volume of a multiple-volume disk drive (for example, a Winchester) is to be used for disk registers.

Syntax `DISCVOLU[?|` <*number*>]

<*number*> can range from 0 to 30.

Query Response <*number*>

DISM

Displays the response and stimulus values for all markers that are turned on.

Syntax `DISM{?|ON|OFF}`

Query Response `{1|0}`

Example `DISMON`

DISP

Displays data and/or memory on the active channel.

Syntax `DISP{DATA|DATM|DDM|DMM|MEMO}`

Option	Description
`DATA`	Displays data only on the active channel.
`DATM`	Displays the data and memory on the active channel.
`DDM`	Displays data divided by memory (linear division, log subtraction) on the active channel.
`DMM`	Displays data minus memory (linear subtraction) on the active channel.
`MEMO`	Displays memory only on the active channel.

Example `DISPMEMO`

DIVI

Displays data divided by memory (linear division, log subtraction) on the active channel.

Syntax	`DIVI`
See Also	`DISPDDM`

DONE

Done with a class of standards, during a calibration. Only needed when multiple standards are measured to complete the class. OPC compatible.

Syntax	`DONE`

DONM

Done modifying a test sequence.

Syntax	`DONM`

DOSEQ

Begins execution of the selected sequence.

Syntax	`DOSEQ{1|2|3|4|5|6}`
Example	`DOSEQ1`

DOWN

Decrements the value in the active entry area (down key).

Syntax `DOWN`

DUAC

Dual channel display.

Syntax `DUAC{?|ON|OFF}`

Query Response `{1|0}`

Example `DUACON`

DUPLSEQ[X]SEQ[Y]

Duplicates sequence X to sequence Y.

Syntax `DUPLSEQ[<X>]SEQ[<Y>]`

<X> and <Y> can range from 1 to 6.

Example `DUPLSEQ1SEQ6`

EDITDONE

Done editing list frequency or limit table. OPC compatible.

Syntax `EDITDONE`

EDITLIML

Begin editing limit table.

Syntax `EDITLIML`

EDITLIST

Begin editing list frequency table.

Syntax `EDITLIST`

ELED

Sets the electrical delay offset.

Syntax `ELED`[`?|` <*number*>]

<*number*> is ±1.0 s

Query Response <*number*>

EMIB

Sends out a beep during a sequence.

Syntax `EMIB`

Description `NEWSEQ` must precede to ensure that a sequence is currently being created or modified.

ENTO

Turns the active entry area OFF.

Syntax `ENTO`

ESB?

Outputs event-status register B.

Syntax `ESB?`

ESE

Enables the selected event-status register bits to be summarized by bit 5 in the status byte.

Syntax ESE[? | <*number*>]

<*number*> is 0 to 255.

Description An event-status register bit is enabled when the corresponding bit in the operand D is set.

Query Response <*number*>

ESNB

Enables the selected event-status register B bits to be summarized by bit 2 of the status byte. Much like `ESE`.

Syntax `ESNB`[`?`| <*number*>]

<*number*> can range from 0 to 255.

Query Response <*number*>

ESR?

Outputs the value of the event-status register.

Syntax `ESR?`

EXTD

Selects the external disk as the active storage device. Used with `STORSEQ`, `TITF`, `PURG`, `STOR`, `LOAD`, and `LOADSEQ`.

Syntax `EXTD`

EXTM

Stores data on disk.

.Syntax `EXTM{DATA|DATO|FORM|GRAP|RAW}{?|ON|OFF}`

Option	Description
`DATA`	Stores register error corrected data on disk.
`DATO`	Stores register data array only on disk.
`FORM`	Stores register formatted trace data on disk.
`GRAP`	Stores user graphics on disk.
`RAW`	Stores raw data arrays on disk.

Query Response `{1|0}`

Example `EXTMRAWON`

EXTT

Sets the external trigger mode.

Syntax `EXTT{HIGH|LOW|OFF|ON|POIN}`

Option	Description
`HIGH`	Sets the external trigger polarity high.
`LOW`	Sets the external trigger polarity low.
`OFF`	Deactivates the external trigger mode. OPC compatible.
`ON`	Activates the external trigger on sweep mode. OPC compatible.
`POIN`	Sets the external trigger to auto trigger on point. OPC compatible.

Example `EXTTPOIN`

FIXE

Specifies a fixed load, as opposed to a sliding load, when defining a standard during a cal kit modification.

Syntax `FIXE`

FOCU

Adjusts display focus.

Syntax `FOCU[?| <number>]`

<number> can range from 0 to 100%.

Query Response *<number>*

FORM

Sets the data format for array transfers in and out of the instrument.

Syntax `FORM{1|2|3|4|5}`

Option	Description
1	Option 011 internal format. Preceded by 4 byte header.
2	32 bit floating point format. Preceded by 4 byte header.
3	64 bit floating point format. Preceded by 4 byte header.
4	ASCII format. No header.
5	32 bit floating point PC format. Bytes reversed.

Example `FORM1`

FORMAT

Selects the disk format.

Syntax `FORMAT{DOS|LIF}`

Option	Description
`DOS`	Selects DOS as the disk format.
`LIF`	Selects LIF as the disk format.

Example `FORMATDOS`

FREO

Frequency blank. Turns off frequency notation.

Syntax `FREO`

FREQOFFS

Activates the frequency offset instrument mode. OPC compatible.

Syntax `FREQOFFS{?|ON|OFF}`

Query Response `{1|0}`

Example `FREQOFFSON`

FRER

HP-IB free run. Acts the same as `CONT`.

Syntax `FRER`

FULP

Selects full page plotting, as opposed to plotting in one of the four quadrants.

Syntax `FULP`

FWD

Selects a forward calibration class during a 2-port calibration sequence.

Syntax `FWD{I|M|T}`

Option	Description
`I`	Selects a forward calibration class to isolation, during a 2-port calibration sequence. OPC compatible if there is only one standard in the class.
`M`	Selects a forward calibration class to match, during a 2-port calibration sequence. OPC compatible if there is only one standard in the class.
`T`	Selects a forward calibration class to transmission, during a 2-port calibration sequence. OPC compatible if there is only one standard in the class.

Example `FWDT`

GATE

Sets the time domain gate time.

Syntax `GATE{CENT|SPAN|STAR|STOP}[?|` <*number*>]

Option	Description
`CENT`	Sets the time domain gate center time.
`SPAN`	Sets the time domain gate span time.
`STAR`	Sets the time domain gate start time.
`STOP`	Sets the time domain gate stop time.

<*number*> values can be within the following ranges:

Condition	Range
Frequency sweeps	30 kHz to 3 GHz (to 6 GHz for option 006)
Power sweeps	–15 to 20 dBm in range 0, 25 dB maximum in other ranges (–100 to +100 with power meter cal on)
CW time	0 to 24 hours
Frequency sweep (transform on)	±1/frequency step
CW time sweep (transform on)	±1/time step

Query Response <*number*>

GATEO

Turns the time domain gate on or off.

Syntax `GATEO{?|ON|OFF}`

Query Response `{1|0}`

Example `GATEOON`

GATS

Sets the gate shape.

Syntax `GATS{MAXI|MINI|NORM|WIDE}`

Option	Description
`MAXI`	Sets the gate shape to maximum.
`MINI`	Sets the gate shape to minimum.
`NORM`	Sets the gate shape to normal.
`WIDE`	Sets the gate shape to wide.

Example `GATSMAXI`

GOSUB

Invokes a sequence as a subroutine. Used with `SEQ`.

Syntax `GOSUB`

GUIS

Initiates the guided setup feature.

Syntax `GUIS`

HARM

Sets the harmonic measurement mode, option 002 (OPC compatible).

Syntax `HARM{OFF|SEC|THIR}`

Option	Description
`OFF`	Turns OFF harmonic measurement mode.
`SEC`	Measures the second harmonic.
`THIR`	Measures the third harmonic.

Example `HARMOFF`

HOLD

Puts the sweep trigger into hold.

Syntax `HOLD`

IDN?

Outputs the identification string: "HEWLETT PACKARD,8702D,0,X.XX", where X.XX is the firmware revision of the instrument.

Syntax `IDN?`

IFBI

Tests the specified input GPIO bit (`PARAIN`).

Syntax `IFBI{HIGH|LOW}`

Option	Description
`HIGH`	Invokes the sequence specified by `SEQ`.
`LOW`	Invokes the sequence specified by `SEQ`.

Example `IFBIHIGH`

IFBW

Sets the IF bandwidth.

Syntax `IFBW[?|` <*number*>]

<*number*> values are 10, 30, 100, 300, 1000, or 3000 Hz.

Query Response <*number*>

IFLC

Branches an executing sequence to a new sequence if condition is satisfied.

Syntax `IFLC{EQZESEQ|NEZESEQ} {1|2|3|4|5|6}`

Option	Description
`EQZESEQ`	If loop counter equals zero, then do sequence. `NEWSEQ` must precede to ensure that a sequence is currently being created or modified.
`NEZESEQ`	If loop counter does not equal zero, then do sequence. `NEWSEQ` must precede to ensure that a sequence is currently being created or modified.

Example `IFLCEQZESEQ 1`

IFLT

Branches an executing sequence to a new sequence if condition is satisfied.

Syntax `IFLT{FAILSEQ|PASSSEQ} {1|2|3|4|5|6}`

Option	Description
`FAILSEQ`	Limit test fails. `NEWSEQ` must precede to ensure that a sequence is currently being created or modified.
`PASSSEQ`	Limit test passes. `NEWSEQ` must precede to ensure that a sequence is currently being created or modified.

Example `IFLTFAILSEQ 1`

IMAG

Selects the imaginary display format.

Syntax `IMAG`

INCRLOOC

Increments the sequencing loop counter by 1. `NEWSEQ` must precede to ensure that a sequence is currently being created or modified.

Syntax `INCRLOOC`

INDEREFR

Sets the index of refraction value of transmission median.

Syntax `INDEREFR`[`?`| <*number*>]

<*number*> can range from .001 to 500.

Query Response <*number*>

INI

Initializes the internal or external disk. All information on the disk will be destroyed.

Syntax

```
INI{D|E}
```

Option	Description
`D`	Initializes the internal disk. All information on the disk will be destroyed.
`E`	Initializes the external disk. All information on the disk will be destroyed. Requires pass control when using the HP-IB port.

Example

```
INID
```

INPU

Inputs a block of binary data in format readout with the corresponding `OUTP` command.

Syntax

```
INPU{CALR|CALS|REC2|SOU2}
```

Option	Description
`CALR`	Disk 1 receiver cal data.
`CALS`	Disk 1 source cal data.
`REC2`	Disk 2 receiver cal data.
`SOU2`	Disk 2 source cal data.

Example

```
INPUCALR
```

INPUCALC

Inputs individual calibration coefficient arrays.

Syntax

`INPUCALC`{<*integer*>} [<*array*>]

<*integer*> can range from array 01 to 12. Array can be form 1–form 5. Refer to "Array-data formats" on page 1-18 for information on the arrary formats.

Description

Before sending the array, issue a `CALIXXXX` command, where `XXX` specifies the calibration type of the data. Then input the cal arrays. Lastly store the data with `SAVC`. The instrument goes into hold, displaying uncorrected data: `SING` completes the process by displaying error corrected data. Refer to "Calibration Arrays" on page 3-85 for the contents of the different arrays.

INPUCALK

Inputs a cal kit read out with `OUTPCALK` . After the transfer, the data should be saved into the user cal kit area with `SAVEUSEK`.

Syntax

`INPUCALK` [<*array*>]

Array can be form 1 only. Refer to "Array-data formats" on page 1-18 for information on the arrary formats.

INPUDATA

Inputs an error corrected data array, using current format. The instrument stops sweeping, and then formats and displays the data.

Syntax

`INPUDATA`[<*array*>] Array can be form 1–form 5. Refer to "Array-data formats" on page 1-18 for information on the arrary formats.

INPUFORM

Inputs a formatted data array, using current format. The instrument stops sweeping and displays the data.

Syntax `INPUFORM`[<*array*>]

Array can be form 1–form 5. Refer to “Array-data formats” on page 1-18 for information on the arrary formats.

INPULEAS

Inputs a learn string read out by `OUTPLEAS`.

Syntax `INPULEAS`[<*string*>]

INPUPMCAL

Inputs power meter calibration arrays into the instrument. Values should be entered as 100× desired source power in dB.

Syntax `INPUPMCAL{1|2}`

INPURAW

Inputs a raw data array using the current format. See `OUTPRAW` for the meaning of the arrays. The instrument stops sweeping, error corrects the data, then formats and displays the data.

Syntax `INPURAW{1|2|3|4}`

INSM

Selects the instrument mode.

Syntax `INSM{EXSA|EXSM|NETA|TUNR}`

Option	Description
`EXSA`	Selects the external source, auto instrument mode (OPC compatible).
`EXSM`	Selects the external source, manual instrument mode (OPC compatible).
`NETA`	Selects the standard analyzer instrument mode (OPC compatible).
`TUNR`	Selects the tuned receiver instrument mode (OPC compatible).

Example `INSMEXSA`

INTD

Selects the internal disk as the active storage device.

Syntax `INTD`

See Also `STORSEQ`, `TITF`, `PURG`, `STOR`, `LOAD`, and `LOADSEQ`.

INTE

Sets the display intensity.

Syntax `INTE[?| <`*number*`>]`

<*number*> can range from 0 to 100%.

Query Response <*number*>

INTM

Selects the internal memory as the active storage device.

Syntax `INTM`

ISO

Isolation subsequence step in a 2-port calibration.

Syntax `ISO{D|L|OP}`

Option	Description
`D`	Done with isolation subsequence in a 2-port calibration. OPC compatible.
`L`	Begins the isolation subsequence step in a 2-port calibration.
`OP`	Begins the isolation subsequence for one path, two port calibration.

Example `ISOD`

KEY

Sends a keycode, equivalent to actually pressing the key. It does not matter if the front-panel is in remote mode. Refer to “Key codes” on page 5-7 for information on the key codes.

Syntax `KEY[?|` <*number*>]

<*number*> can range from 1 to 63.

Query Response <*number*>

KITD

Calibration kit done: the last step in modifying a cal kit.

Syntax `KITD`

KOR?

Outputs a two byte key code/knob count. If the number is positive, it is a key code. If it is negative, then set bit 15 equal to bit 14. The resulting integer is the knob count, either positive or negative, depending on the direction of turn (negative for clockwise.) There are approximately 120 counts per knob turn.

Syntax `KOR?`

LAB

Enters a label during a cal kit modification.

Syntax

`LAB<option> [<string>]`

<string> value can be up to 10 characters.

<option> value can be:

Option	Description
`EFWDM`	Enters label for forward match.
`EFWDT`	Enters label for forward transmission.
`ERESI`	Enters label for response, response and isolation.
`ERESP`	Enters label for response.
`EREVM`	Enters label for reverse match.
`REVT`	Enters label for reverse transmission.
`ES11A`	Enters label for S11A (opens).
`ES11B`	Enters label for S11B (shorts).
`ES11C`	Enters label for S11C (loads).
`ES22A`	Enters label for S22A (opens).
`ES22B`	Enters label for S22B (shorts).
`ES22C`	Enters label for S22C (loads).
`ETLFM`	Enters label for TRL, Line, Forward, Match.
`ETLFT`	Enters label for TRL, Line, Forward, Trans.
`ETLRM`	Enters label for TRL, Line, Reverse, Match.
`ETLRT`	Enters label for TRL, Line, Reverse, Trans.
`ETRFM`	Enters label for TRL, Reflect, Forward, Match.
`ETRRM`	Enters label for TRL, Reflect, Reverse, Match.
`ETTFM`	Enters label for TRL, Thru, Forward, Match.
`ETTFT`	Enters label for TRL, Thru, Forward, Trans.
`ETTRM`	Enters label for TRL, Thru, Reverse, Match.
`ETTRT`	Enters label for TRL, Thru, Reverse, Trans.
`K`	Enters a cal kit label.
`S`	Enters a standard's label during standard definition.

LEF

Selects a plot in the left upper or lower quadrant.

Syntax `LEF{L|U}`

Option	Description
`L`	Selects a plot in the left lower quadrant.
`U`	Selects a plot in the left upper quadrant.

Example `LEFL`

LIM

Sets the limit parameters.

Syntax

`LIM`*<option>*`[?|` *<number>*]

<option> values can be:

Option	Description
`IAMPO`	Enters the limit line amplitude offset.
`D`	Sets the limit delta value while editing a limit line segment.
`L`	Sets the lower limit value.
`M`	Sets the middle limit value.
`S`	Sets the limit stimulus break point.
`U`	Sets the upper limit value.

<number> values can range from:

For log magnitude:	±500 dB
For phase:	±500 degrees
For Smith chart and polar:	±500 units
For linear magnitude:	±500 units
For SWR:	±500 units

The scale is always positive, and has minimum values of .001 dB, 10e–12 degrees, 10e–15 seconds, and 10 picounits.

Query Response *<number>*

LIMILINE

Turns the display of the limit lines on and off.

Syntax	`LIMILINE{?|ON|OFF}`
Query Response	{`1|0`}
Example	`LIMILINEON`

LIMIMAOF

Marker to limit offset. Centers the limit lines about the current marker position using the limit amplitude offset function.

Syntax	`LIMIMAOF`

LIMISTIO

Enters the stimulus offset of the limit lines.

Syntax

`LIMISTIO[?| <number>]`

<*number*> may be values:

Condition	Range
Frequency sweeps	30 kHz to 3 GHz (to 6 GHz for option 006)
Power sweeps	–15 to 20 dBm in range 0, 25 dB maximum in other ranges (–100 to +100 with power meter cal on)
CW time	0 to 24 hours
Frequency sweep (transform on)	±1/frequency step
CW time sweep (transform on)	±1/time step

Query Response <*number*>

LIMITEST

Turns limit testing on and off.

Syntax

`LIMITEST{?|ON|OFF}`

Query Response {`1|0`}

Example

`LIMITESTON`

LIMT

Sets the limit segment value.

Syntax `LIMT{FL|SL|SP}`

Option	Description
`FL`	Makes the segment a flat line.
`SL`	Makes the segment a sloping line.
`SP`	Makes the segment a single point.

Example `LIMTFL`

LIN

Sets the linear value.

Syntax `LIN{FREQ|M}`

Option	Description
`FREQ`	Selects a linear frequency sweep.
`M`	Selects the linear magnitude display format.

Example `LINFREQ`

LINT

Enters the plotting line type.

Syntax `LINT{DATA|MEMO}[?|` *<number>*]

Option	Description
`DATA`	Enters the line type for plotting data.
`MEMO`	Enters the line type for plotting memory.

<number> can range from 0 to 7.

Query Response *<number>*

LISFREQ

Selects the list frequency sweep mode.

Syntax `LISFREQ`

LISV

Activates the list values function.

Syntax `LISV`

Description The next page of values can be called with `NEXP`. The current page can be plotted or printed, in raster graphics mode, with `PLOT` or `PRINALL`. The entire list can be printed, in ASCII text mode, with `PRINTALL`.

LOAD

Loads the file from disk with the name indicated by the previous `TITFn` command. The actual file recalled depends on the file title in the file position specified. Requires pass control mode when using the HP-IB port.

Syntax `LOAD{1|2|3|4|5}`

Option	Description
`1`	Loads the file from disk using the file name provided by the preceding `TITF1` command.
`2`	Loads the file from disk using the file name provided by the preceding `TITF2` command.
`3`	Loads the file from disk using the file name provided by the preceding `TITF3` command.
`4`	Loads the file from disk using the file name provided by the preceding `TITF4` command.
`5`	Loads the file from disk using the file name provided by the preceding `TITF5` command.

Example `LOAD1`

LOADRE

Loads cal data from disk with filename indicated by the previous `TITFn` or `READRECTn` commands.

Syntax `LOADRE{C1|C2|C3|C4|C5|21|22|23|24|25}`

Option	Description
`C1-C5`	Loads cal data from disk into Receiver1.
`21-25`	Loads cal data from disk into Receiver2.

Example `LOADREC1 or LOADRE21`

LOADSEQ

Loads the indicated sequence from disk. Requires pass control mode when using the HP-IB port.

Syntax `LOADSEQ{1|2|3|4|5|6}`

Option	Description
`1`	Loads sequence 1 from disk.
`2`	Loads sequence 2 from disk.
`3`	Loads sequence 3 from disk.
`4`	Loads sequence 4 from disk.
`5`	Loads sequence 5 from disk.
`6`	Loads sequence 6 from disk.

Example `LOADSEQ1`

LOADSO

Loads cal data from disk with filename indicated by the previous `TITFn` or `READSOUTn` commands.

Syntax `LOADSO{U1|U2|U3|U4|U5|21|22|23|24|25}`

Option	Description
`U1-U5`	Loads the cal data from disk into Source1.
`21-25`	Loads the cal data from disk into Source2.

Example `LOADSOU1` or `LOADSO21`

LOG

Selects the log value.

Syntax `LOG{FREQ|M}`

Option	Description
`FREQ`	Selects a log frequency sweep.
`M`	Selects the log magnitude display format.

Example `LOGFREQ`

LOOC

Sets the value of the sequencing loop counter. `NEWSEQ` must precede to ensure that a sequence is currently being created or modified.

Syntax

`LOOC`[`?`| <*number*>]

<*number*> value can range from 0 to 32,760.

Query Response

<*number*>

LOW

Sets the low pass transform (option 110).

Syntax

`LOW`{`PIMPU`|`PSTEP`}

Option	Description
`PIMPU`	Turns off the low pass impulse transform (option 110).
`PSTEP`	Turns off the low pass step transform (option 110).

Example

`LOWPIMPU`

LRN

Description Same as `OUTPLEAS`. Outputs the learn string, which contains the entire front panel state, the limit table, and the list frequency table. It is always in form 1.

Syntax `LRN{?|[<`*string*`>]}`

Description Same as `INPULEAS`. Inputs a learn string read out by `OUTPLEAS`.

Syntax `LRN`[<*string*>]

MANTRIG

Sets the external trigger to manual trigger on point. OPC compatible.

Syntax `MANTRIG`

MARK

Makes the indicated marker active and sets its stimulus.

Syntax

`MARK{1|2|3|4|5}[?| <number>]`

<number> values may be:

Condition	Range
Frequency sweeps	30 kHz to 3 GHz (to 6 GHz for option 006)
Power sweeps	–15 to 20 dBm in range 0, 25 dB maximum in other ranges (–100 to +100 with power meter cal on)
CW time	0 to 24 hours
Frequency sweep (transform on)	±1/frequency step
CW time sweep (transform on)	±1/time step

Query Response *<number>*

MARKBUCK

Places the marker on a specific sweep point (bucket).

Syntax

`MARKBUCK[?| <number>]`

<number> is the bucket number, ranging from 0 to number of points less 1 (0 to 200, on 201 point sweep).

Query Response *<number>*

MARKCENT

Enters the marker stimulus as the center stimulus.

Syntax `MARKCENT`

MARKCONT

Places the markers continuously on the trace, not on discrete sample points.

Syntax `MARKCONT`

MARKCOUP

Couples the markers between the channels, as opposed to `MARKUNCO`.

Syntax `MARKCOUP`

MARKCW

Sets the CW frequency to the marker frequency.

Syntax `MARKCW`

MARKDELA

Sets electrical length so group delay is zero at the marker stimulus.

Syntax `MARKDELA`

MARKDISC

Places the markers in discrete placement mode.

Syntax `MARKDISC`

MARKFAUV

Sets the auxiliary value of the fixed marker position. Works in coordination with `MARKFVAL` and `MARKFSTI`.

Syntax

`MARKFAUV[?|` <*number*>]

<*number*> can range from:

For log magnitude:	±500 dB
For phase:	±500 degrees
For Smith chart and polar:	±500 units
For linear magnitude:	±500 units
For SWR:	±500 units

The scale is always positive, and has minimum values of .001 dB, 10e–12 degrees, 10e–15 seconds, and 10 picounits.

Query Response

<*number*>

MARKFSTI

Sets the stimulus position of the fixed marker.

Syntax

`MARKFSTI[?| <number>]`

<number> values may be:

Condition	Range
Frequency sweeps	30 kHz to 3 GHz (to 6 GHz for option 006)
Power sweeps	–15 to 20 dBm in range 0, 25 dB maximum in other ranges (–100 to +100 with power meter cal on)
CW time	0 to 24 hours
Frequency sweep (transform on)	±1/frequency step
CW time sweep (transform on)	±1/time step

Query Response *<number>*

MARKFVAL

Sets the value of the fixed marker position. Refer to Table 5-2, “Units as a Function of Display Format,” on page 5-6 for the meaning of value and auxiliary value as a function of display format.

Syntax

`MARKFVAL[?| <number>]`

<number> can range from:

For log magnitude:	±500 dB
For phase:	±500 degrees
For Smith chart and polar:	±500 units
For linear magnitude:	±500 units
For SWR:	±500 units

The scale is always positive, and has minimum values of .001 dB, 10e–12 degrees, 10e–15 seconds, and 10 picounits.

Query Response

<number>

MARKMAXI

Places the active marker for trace maximum.

Syntax

`MARKMAXI`

MARKMIDD

During a limit segment edit, makes the marker amplitude the limit segment middle value.

Syntax

`MARKMIDD`

MARKMINI

Places the active marker for trace minimum.

Syntax `MARKMINI`

MARKOFF

Turns all markers and marker functions OFF.

Syntax `MARKOFF`

MARKREF

Enters the marker amplitude as the reference value.

Syntax `MARKREF`

MARKSPAN

Enters the span between the active marker and the delta reference as the sweep span.

Syntax `MARKSPAN`

MARKSTAR

Enters the marker stimulus as the start stimulus.

Syntax `MARKSTAR`

MARKSTIM

During a limit segment edit, enters the marker stimulus as the limit stimulus break point.

Syntax `MARKSTIM`

MARKSTOP

Enters the marker stimulus as the stop stimulus.

Syntax `MARKSTOP`

MARKUNCO

Uncouples the markers between channels, as opposed to `MARKCOUP`.

Syntax `MARKUNCO`

MARKZERO

Places the fixed marker at the active marker position and makes it the delta reference.

Syntax `MARKZERO`

MAXF

Sets the maximum valid frequency of a standard being defined during a cal kit modification.

Syntax `MAXF[?|` <*number*>]

<*number*> can range from 0 to 1000 GHz.

Query Response <*number*>

MEAS

Measures and display the selected input/parameter on the active channel.

Syntax `MEAS{A|B|EO1|EO2|O1|O2|OE1|OE2|OO1|OO2|R}`

Option	Description
`A`	Measures and displays input A on the active channel.
`B`	Measures and displays input B on the active channel.
`EO1`	Measures and displays E/O transmission parameter as Port 1 → Port 2 or B/R.
`EO2`	Measures and displays E/O transmission parameter as A/R (Option 011 only).
`O1`	Measures and displays optical reflection parameter.
`O2`	Measures and displays optical reflection parameter (Option 011 only).
`OE1`	Measures and displays O/E transmission parameter as Port 1 → Port 2 or B/R.
`OE2`	Measures and displays O/E transmission parameter as A/R (Option 011 only).
`OO1`	Measures and displays O/O transmission parameter.
`OO2`	Measures and displays O/O transmission parameter (Option 001 only).
`R`	Measures and displays input R on the active channel.

Description Parameters followed by 1 are measured with Port 1 as the RF output and Port 2 for RF input for standard instruments; Option 011 (delete internal test set) measures as B/R. Parameters followed by a 2 are Option 011 only and are measured as A/R.

Example `MEASA`

MEASTAT

Turns trace statistics on and off.

Syntax	`MEASTAT{?|ON|OFF}`
Query Response	{`1|0`}
Example	`MEASTATON`

MENU

Activates the selected menu.

Syntax

`MENU{?|ON|OFF}{<`*option*`>}`

<*option*> values may be:

Option	Description
`{ON|OFF}`	Blanks the softkey menu.
`AVG`	Activates the AVG menu.
`CAL`	Activates the CAL menu.
`COPY`	Activates the COPY menu.
`DISP`	Activates the DISPLAY menu.
`FORM`	Activates the FORMAT menu.
`MARK`	Activates the MARKER menu.
`MEAS`	Activates the MEAS menu.
`MRKF`	Activates the MARKER FCTN menu.
`RECA`	Activates the SAVE RECALL menu.
`SCAL`	Activates the SCALE menu.
`SEQU`	Activates the SEQ menu.
`STIM`	Activates the STIMULUS menu.
`SYST`	Activates the SYSTEM menu.

Query Response

`{1|0}`

Example

```
MENUAVG
```

MINF

Sets the minimum valid frequency of a standard being defined during a cal kit modification.

Syntax `MINF`[`?|` <*number*>]

<*number*> can range from 0 to 1000 GHz.

Query Response <*number*>

MINU

Displays data minus memory, the same as `DISPDMM`.

Syntax `MINU`

MODI1

Begins the modify cal kit sequence.

Syntax `MODI1`

NEWSEQ

Begin modifying a sequence.

Syntax `NEWSEQ`{`1|2|3|4|5|6`}

Example `NEWSEQ1`

NEXP

Displays the next page of the operating parameters list.

Syntax `NEXP`

NOOP

No operation. OPC compatible.

Syntax `NOOP`

NUMG

Activates the number of groups of sweeps. A group is whatever is needed to update the current parameter once. This function restarts averaging if ON. OPC compatible.

Syntax `NUMG[?|` <*number*>]

<*number*> can range from 1 to 999.

Query Response <*number*>

NUMR

Sets the number of power meter readings per point used during a power meter calibration.

Syntax `NUMR[?|` <*number*>]

<*number*> can range from 1 to 100.

Query Response <*number*>

OFS

Specifies the offset during a cal kit modification.

Syntax `OFS{D|L|Z}[?|` <*number*>]

Option	Description
`D`	Specifies the delay offset during a cal kit modification.
`L`	Specifies the loss offset during a cal kit modification.
`Z`	Specifies the impedance offset during a cal kit modification.

<*number*> values may be:

Option	<number> value
`D`	±1 s
`L`	0 to 1000 TΩ/s
`Z`	0.1 to 500Ω

Query Response <*number*>

OFSO

Sets the optical standard parameter.

Syntax `OFSO{INDR|LENG|LOSS|RPOW}`

Option	Description
`INDR`	Sets the index of refraction parameter of the optical standard.
`LENG`	Sets the offset length parameter of the optical standard.
`LOSS`	Sets the offset loss parameter of the optical standard.
`RPOW`	Sets the reflected optical power parameter of the optical standard.

Example `OFSOINDR`

OMII

Omits the isolation step of a calibration sequence.

Syntax `OMII`

OPC

Operation complete. Reports the completion of the next command received by setting bit 0 in the event-status register, or by replying to an interrogation if `OPC?` is issued.

Syntax `OPC`

OPEP

Presents a list of key operating parameters. `NEXP` scrolls to the next page of parameters. Requesting a plot or print copies the current page. The current page can be plotted or printed, in raster graphics mode, with `PLOT` or `PRINALL`. The entire list can be printed, in ASCII text mode, with `PRINTALL`.

Syntax `OPEP`

OUTP

Outputs disk cal data as block of binary data.

Syntax `OUTP{CALR|CALS|REC2|SOU2}`

Option	Description
`CALR`	Receiver 1 disk data.
`CALS`	Source 1 disk data.
`REC2`	Receiver 2 disk data.
`SOU2`	Source 2 disk data.

Example `OUTPCALR`

OUTP

Outputs to the HP-IB ports.

Syntax `OUTP{PLOT|PRIN}`

Option	Description
`PLOT`	Outputs the plot string to the HP-IB ports. Can be directed to a plotter, or read into the computer. `PSOFT ON` and `OFF` controls whether the softkeys are included in the plot.
`PRIN`	Outputs to the HP-IB port a raster dump of the display, intended for a graphics printer. `PSOFT ON` and `OFF` controls whether the softkeys are included in the printout.

Example `OUTPPLOT`

OUTPACTI

Outputs the value of the active function, or the last active function if the active entry area is OFF.

Syntax `OUTPACTI`

OUTPAPER

Outputs the smoothing aperture in stimulus units, rather than as a percentage.

Syntax `OUTPAPER`

OUTPCALC

Outputs the error correction arrays for the active calibration on the active channel. Each array comes out in the current output format. They contain real/imaginary pairs, the same number of pairs as points in the sweep.

Syntax

`OUTPCALC`[<*string*>]

<*string*> can range from array `01` to `12`.

Table 3-6. Calibration Arrays

Array	Response	Response and Isolation	1-port	2-port[a]	TRL*/LRM*
1	ER or ET	E_X (ED)[b]	E_D	E_{DF}	E_{DF}
2		E_T (ER)	E_S	E_{SF}	E_{SF}
3			E_R	E_{RF}	E_{RF}
4				E_{XF}	E_{XF}
5				E_{LF}	E_{LF}
6				E_{TF}	E_{TF}
7				E_{DR}	E_{DR}
8				E_{SR}	E_{SR}
9				E_{RR}	E_{RR}
10				E_{XR}	E_{XR}
11				E_{LR}	E_{LR}
12				E_{TR}	E_{TR}

a. One path, 2-port cal duplicates arrays 1 to 6 in arrays 7 to 12.
b. Response and isolation corrects for crosstalk and transmission tracking in transmission measurements, and for directivity and reflection tracking in reflection measurements.

Meaning of first subscript:

D	=	directivity
S	=	source match
R	=	reflection tracking
X	=	crosstalk
L	=	load match
T	=	transmission tracking

Meaning of second subscript:

F	=	forward
R	=	reverse

OUTPCALK

Outputs the active calibration kit, as a less than 1000 byte string, in Form 1.

Syntax `OUTPCALK`

OUTPDATA

Outputs the error corrected data from the active channel in the current format. Refer to Figure 5-1, “Data processing chain,” on page 5.

Syntax `OUTPDATA`

OUTPERRO

Outputs the oldest error message in the error queue. Sends first the error number, and then the error message itself as a string no longer than 50 characters.

Syntax `OUTPERRO`

OUTPFORM

Outputs the formatted display data array from the active channel in the current format. Refer to Table 5-2, “Units as a Function of Display Format,” on page 5-6 for the contents of the array positions as a function of display format.

Syntax `OUTPFORM`

OUTPICALC

Outputs, over HP-IB, the interpolated calibration coefficient arrays for the active calibration on the active channel.

Syntax `OUTPICALC`[<*string*>]

<*string*> can range from array `01` to `12`.

OUTPIDEN

Outputs the identification string for the analyzer: "HEWLETT PACKARD,8702D,0,X.XX" where X.XX is the firmware revision.

Syntax `OUTPIDEN`

OUTPIPMCAL

Outputs, over HP-IB, the interpolated power meter calibration array for channel 1 or channel 2.

Syntax `OUTPIPM{CAL1|CAL2}`

Example `OUTPIPMCAL1`

OUTPKEY

Outputs the key code of the last key pressed. An invalid key is reported with a 63, a knob turn with a –1. Refer to Figure 5-2, “Key codes,” on page 7 for the front-panel key codes.

Syntax `OUTPKEY`

OUTPLEAS

Outputs the learn string, which contains the entire front panel state, the limit table, and the list frequency table. It is always in Form 1.

Syntax `OUTPLEAS`

OUTPLIM

Outputs the limit test results.

Syntax `OUTPLIM{F|L|M}`

Option	Description
`F`	Outputs the limit test results for each failed point.
`L`	Outputs the limit test results for each point in the sweep. This is a Form 4 transfer.
`M`	Outputs the limit test results at the marker.

Description The results consist of four fields. First is the stimulus value for the point. Second is an integer indicating test status. Third is the upper limit at that point. Fourth is the lower limit at that point. If there are no limits at that point, the third and fourth fields are zero. The test status is –1 for no test, 0 for fail, and 1 for pass.

Example `OUTPLIMF`

OUTPMARK

Outputs the marker values. The first two numbers are the marker response values, and the last is the stimulus value. Refer to Table 5-2, "Units as a Function of Display Format," on page 5-6 for the meaning of the response values as a function of display format.

Syntax `OUTPMARK`

OUTPMEMO

Outputs the memory trace from the active channel. The data is in real/imaginary pairs, and can be treated the same as data read with the `OUTPDATA` command.

Syntax `OUTPMEMO`

OUTPMSTA

Outputs the marker statistics: mean, standard deviation, and peak-to-peak variation, in that order. If statistics is not ON, it is turned ON to generate current values and turned OFF again.

Syntax `OUTPMSTA`

OUTPMWID

Outputs the marker bandwidth search results: bandwidth, center, and Q, in that order. If widths is not ON, it is turned ON to generate current values and turned OFF again.

Syntax `OUTPMWID`

OUTPMWIL

Outputs the marker bandwidths search results: bandwidth, center, loss, and Q in that order. If widths is not ON, it is turned ON to generate current values and turned OFF again.

Syntax `OUTPMWIL`

OUTPPMCAL

Outputs the power meter calibration array for channel 1 or channel 2. Values are sent as 100 times the source power in dB. A default array is used if a power meter calibration sweep, `TAKCS`, has not been taken.

Syntax `OUTPPM{CAL1|CAL2}`

Example `OUTPPMCAL1`

OUTPPRNALL

Prints all list values or operating and marker parameters in text mode to HP-IB. Activate desired function with `LISV` for values or `OPEP` for operating parameters.

Syntax `OUTPPRNALL`

OUTPRAW

Outputs the raw measurement data. Refer to Figure 5-1, “Data processing chain,” on page 5 for the meaning of the data. Normally, array 1 holds the current parameter. If a 2-port calibration is active, the arrays hold S11, S21, S12, and S22, respectively.

Syntax `OUTPRAW{1|2|3|4}`

Example `OUTPRAW1`

OUTPRFFR

Outputs the external source RF frequency. The instrument must be in external source mode, either `INSMECSA` or `INSEXSM`.

Syntax `OUTPRFFR`

OUTPSEQ

Outputs a sequence listing over HP-IB.

Syntax `OUTPSEQ{1|2|3|4|5|6}`

Example `OUTPSEQ1`

OUTPSTAT

Outputs the status byte.

Syntax `OUTPSTAT`

OUTPTITL

Outputs the display title.

Syntax `OUTPTITL`

PARAIN

Specified the input GPIO bit to be used by `IFBIHIGH` and `IFBILOW` tests.

Syntax

`PARAIN[?|` <*number*>]

<*number*> can range from 0 to 4.

Query Response

<*number*>

PARAL

Selects use of the parallel port: for general purpose I/O or for the copy function.

Syntax

`PARAL{?|GPIO|CPY}`

Values may be:

GPIO (General Purpose I/O)

CPY (Copy use)

Query Response

`{1|0}`

PARAOUT

Programs all GPIO output bits (0 to 255) at once.

Syntax

`PARAOUT[?|` <*number*>]

<*number*> can range from 0 to 255.

Query Response

<*number*>

PAUS

Inserts a pause into a sequence. `NEWSEQ` must precede to ensure that a sequence is currently being created or modified.

Syntax `PAUS`

PCB

Same as `ADDRCONT`. Indicates where control will be passed in pass control mode.

Syntax `PCB`[`?`**|** <*number*>]

<*number*> can range from 0 to 30.

Query Response <*number*>

PCOL

Selects the color for the indicated display feature.

Syntax `PCOL{<`*option*`>}[<`*color*`>]`

Option	Description
`DATA1`	Selects the color for channel 1 data.
`DATA2`	Selects the color for channel 2 data.
`MEMO1`	Selects the color for channel 1 memory.
`MEMO2`	Selects the color for channel 2 memory.
`GRAT`	Selects the color for the graticule.
`TEXT`	Selects the color for the display text.
`WARN`	Selects the color for the warning text.

<*color*> can be white, cyan, magneta, blue, yellow, green, red, or black.

PDATA

Selects whether trace data is plotted.

Syntax `PDATA{?|ON|OFF}`

Query Response `{1|0}`

Example `PDATAON`

PENN

Selects the pen for plotting the indicated display feature.

Syntax `PENN{DATA|GRAT|MARK|MEMO|TEXT}[?| <`*number*`>]`

Option	Description
`DATA`	Selects the pen for plotting the data trace.
`GRAT`	Selects the pen for plotting the graticule.
`MARK`	Selects the pen for plotting the markers and marker text.
`MEMO`	Selects the pen for plotting the memory trace.
`TEXT`	Selects the pen for plotting text and user graphics.

<*number*> can range from 0 to 10.

Query Response <*number*>

PGRAT

Selects whether the graticule is plotted.

Syntax `PGRAT{?|ON|OFF}`

Query Response `{1|0}`

Example `PGRATON`

PHAO

Sets the phase offset.

Syntax `PHAO[?|` <*number*>]

<*number*> range is ±360 degrees.

Query Response <*number*>

PHAS

Selects the phase display format.

Syntax `PHAS`

PLOS

Selects the pen speed for plotting.

Syntax `PLOS{SLOW|FAST}`

Example `PLOSFAST`

PLOT

Initiates a plot. Requires pass control mode when using the HP-IB port.

Syntax `PLOT`

PLTHNDSHK

Selects the plotter handshake mode as either Xon-Xoff or DTR-DSR, for serial (RS-232) bus.

Syntax `PLTHNDSHK{?|DTR|XON}`

Option	Description
`DTR`	Selects the `DTR-DSR` plotter handshake mode.
`XON`	Selects the `XON-XOFF` plotter handshake mode.

Query Response `{1|0}`

PLTPRT

Sets the plotter port.

Syntax `PLTPRT{HPIB|PARA|SERI}`

Option	Description
`HPIB`	Sets the plotter port to HP-IB.
`PARA`	Sets the plotter port to parallel.
`SERI`	Sets the plotter port to serial (RS-232).

Example `PLTPRTHPIB`

PLTPRTDISK

Plots measurement results to a disk for later retrieval.

Syntax `PLTPRTDISK`

PLTTRAUTF

Turns on and off the plotter auto feed.

Syntax	`PLTTRAUTF{?|ON|OFF }`
Query Response	{`1|0`}
Example	`PLTTRAUTFON`

PLTTRBAUD

Sets the plotter baud rate, for serial (RS-232) bus.

Syntax	`PLTTRBAUD`[`?|` <*number*>] <*number*> can be 1200, 2400, 4800, 9600, or 19200.
Query Response	<*number*>

PLTTRFORF

Sends a form feed to the plotter.

Syntax	`PLTTRFORF`

PLTTYP

Selects the plotter type.

Syntax `PLTTYP{HPGL|PLTR}`

Option	Description
`HPGL`	Selects HPGL-compatible printer as the plotter type.
`PLTR`	Selects plotter as the plotter type.

Example `PLTTYPHPGL`

PMEM

Selects whether memory is plotted.

Syntax `PMEM{?|ON|OFF}`

Query Response {`1|0`}

Example `PMEMON`

PMKR

Selects whether markers are plotted.

Syntax `PMKR{?|ON|OFF}`

Query Response {`1|0`}

Example `PMKRON`

PMTRTTIT

Reads power meter/HP-IB value into title string. `NEWSEQ` must precede to ensure that a sequence is currently being created or modified.

Syntax `PMTRTTIT`

POIN

Sets the number of points in the sweep. If a list frequency segment is being edited, sets the number of points in the segment.

Syntax `POIN[?|` <*number*>]

<*number*> can range from 1 to 1632.

Query Response <*number*>

POLA

Selects the polar display format.

Syntax `POLA`

POLM

Selects marker readout format for polar display.

Syntax `POLM{LIN|LOG|MRI}`

Option	Description
`LIN`	Selects linear markers for polar display.
`LOG`	Selects log markers for polar display.
`MRI`	Selects real/imaginary markers for polar display.

Example `POLMLIN`

PORE

Turn port extensions on and off.

Syntax `PORE{?|ON|OFF}`

Query Response {`1|0`}

Example `POREON`

PORT

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

Syntax `PORT{1|2|A|B}[?|` *<number>*]

Option	Description
`1`	Port 1
`2`	Port 2
`A`	Input A
`B`	Input B

<number> value is ±10 s

Query Response *<number>*

PORTP

Selects either coupled or uncoupled for the port powers for a given channel.

Syntax `PORTP{?|CPLD|UNCPLD}`

Query Response `{1|0}`

POWE

Sets the output power level.

Syntax

`POWE[?|` *<number>*]

<number> can range from –85 to +20 dBm.

Query Response

<number>

POWLFREQ

Selects the frequency for which a power loss correction is entered. This must be followed by `POWLLOSS`, which sets the value.

Syntax

`POWLFREQ[?|` *<number>*]

<number> values may be:

Condition	Range
Frequency sweeps	30 kHz to 3 GHz (to 6 GHz for option 006)
Power sweeps	–15 to 20 dBm in range 0, 25 dB maximum in other ranges (–100 to +100 with power meter cal on)
CW time	0 to 24 hours
Frequency sweep (transform on)	±1/frequency step
CW time sweep (transform on)	±1/time step

Query Response

<number>

POWLLIST

Begins editing a power loss list for a power meter calibration.

Syntax	`POWLLIST`

POWLLOSS

Sets the loss value for a particular frequency, `POWLFREQ`, in the power loss list.

Syntax	`POWLLOSS[?|` <*number*>] <*number*> can range from –9900 to 9900 dB.
Query Response	<*number*>

POWM

Selects whether the HP 436A (ON) or the HP 437B/438A (OFF) is to be used as the power meter in service procedures.

Syntax	`POWM{?|ON|OFF}`
Query Response	{`1|0`}
Example	`POWMON`

POWS

Selects power sweep, from the sweep type menu.

Syntax `POWS`

POWT

Turns power trip OFF which clears a power trip after an overload condition is detected at one of the input ports.

Syntax `POWT{?|ON|OFF}`

Query Response `{1|0}`

Example `POWTON`

PRAN

Selects the power range when in manual power range. Used with `PWRR` and `PMAN`.

Syntax `PRAN{0|1|2|3|4|5|6|7}`

Example `PRAN1`

PRES

Presets the analyzer to the factory preset state.

Syntax `PRES`

PRIC

Selects color print.

Syntax `PRIC`

PRIN

Begins printing the format selected. Requires pass control mode when using the HP-IB port.

Syntax `PRIN{ALL|TALL}`

Option	Description
`ALL`	Copies the display, in raster graphics mode, to a printer.
`TALL`	Prints all list values or operating and marker parameters in ASCII text mode.

Example `PRINALL`

PRINSEQ

Begins printing the sequence selected. Requires pass control mode when using the HP-IB port.

Syntax `PRINSEQ{1|2|3|4|5|6}`

Example `PRINSEQ1`

PRIS

Selects standard (monochrome) print.

Syntax `PRIS`

PRNHNDSHK

Selects the printer handshake mode as either Xon-Xoff or DTR-DSR, for serial (RS-232) bus.

Syntax `PRNHNDSHK{?|DTR|XON}`

Option	Description
`DTR`	DTR-DSR
`XON`	XON-XOFF

Query Response `{1|0}`

PRNPRT

Sets the printer port.

Syntax `PRNPRT{HPIB|PARA|SERI}`

Option	Description
`HPIB`	Sets the printer port to HP-IB.
`PARA`	Sets the printer port to parallel.
`SERI`	Sets the printer port to serial (RS-232) bus.

Example `PRNPRTHPIB`

PRNTRAUTF

Turns on and off the printer auto feed.

Syntax `PRNTRAUTF{?|ON|OFF}`

Query Response {`1|0`}

Example `PRNTRAUTFON`

PRNTRBAUD

Sets the printer baud rate.

Syntax `PRNTRBAUD[?|` <*number*>]

<*number*> can be 1200, 2400, 4800, 9600, or 19200.

Query Response <*number*>

PRNTRFORF

Sends a form feed to the printer.

Syntax `PRNTRFORF`

PRNTYP

Selects the printer type.

Syntax `PRNTYP{DJ|EP|LJ|PJ|TJ}`

Option	Description
`DJ`	Selects the DeskJet printer as the printer type.
`EP`	Selects the Epson ESC/P2 printer control language-compatible printer as the printer type.
`LJ`	Selects the LaserJet printer as the printer type.
`PJ`	Selects the PaintJet printer as the printer type.
`TJ`	Selects the ThinkJet printer as the printer type.

Example `PRNTYPDJ`

PSOFT

Controls whether softkeys are included in the hardcopy print or plot when using one of the following commands: `PLOT`, `PRINALL`, `OUTPPLOT` or `OUTP-PRIN`.

Syntax	`PSOFT{?|ON|OFF }`
Query Response	{`1|0`}
Example	`PSOFTON`

PTEXT

Selects whether text is plotted.

Syntax	`PTEXT{?|ON|OFF }`
Query Response	{`1|0`}
Example	`PTEXTON`

PTOS

Pauses the sequence to be followed by selection one of the six sequences (SEQ).

Syntax	`PTOS`

PURG

Purges the indicated file from disk. Requires pass control mode when using the HP-IB port.

Syntax `PURG{1|2|3|4|5}`

Example `PURG1`

PWMC

Select the type of power meter calibration desired.

Syntax `PWMC{EACS|OFF|ONES}[?|` *<number>*`]`

Option	Description
`EACS`	Selects the type of power meter calibration desired to each sweep.
`OFF`	Selects the type of power meter calibration desired to OFF.
`ONES`	Selects the type of power meter calibration desired to one sweep.

<number> can be –100 to 100 dB.

Query Response *<number>*

PWRLOSS

Selects whether or not to use the power loss table for a power meter calibration.

Syntax `PWRLOSS{?|ON|OFF}`

Query Response `{1|0}`

Example `PWRLOSSON`

PWRMCAL

Displays the power meter cal menu and sets the drive port cal power.

Syntax `PWRMCAL[?|` *<number>*]

<number> can be –100 to 100 dB.

Query Response *<number>*

PWRR

Selects the source power range auto or manual mode.

Syntax `PWRR{?|PAUTO|PMAN}`

Query Response `{1|0}`

Example `PWRR1`

RAI

Calls the class for the response and isolation calibration. OPC compatible if only one standard in class.

Syntax `RAI{ISOL|RESP}`

Option	Description
`ISOL`	Calls the isolation class for the response and isolation calibration.
`RESP`	Calls the response class for the response and isolation calibration.

Example `RAIISOL`

RAID

Completes the response and isolation cal sequence. OPC compatible.

Syntax `RAID`

RAMD

Indicates that the response and match cal is done.

Syntax `RAMD`

RE

Calculates the cal standard values at the frequency of each point in the current sweep and stores it in the trace memory for the active channel.

Syntax `RE{CCSTDI|CDSTDI|2DSTDI}`

Option	Description
`CCSTDI`	Calculates receiver coefficient standard and copies into memory.
`CDSTDI`	Calculates disk Receiver1 standard and copies into memory.
`2DSTDI`	Calculates disk Receiver2 standard and copies into memory.

Example `RECCSTDI`

READ

Outputs the date/time in string format. There is no front-panel equivalent.

Syntax `READ{DATE|TIME}`

Option	Description
`DATE`	Outputs the date in the following string format: DD MMM YYYY.
`TIME`	Outputs the time in the following string format: HH:MM:SS.

Example `READDATE`

READRECT

Reads up to five receiver cal file titles.

Syntax `READRECT`

READSOUT

Reads up to five source cal file titles.

Syntax `READSOUT`

REAL

Selects the real display format.

Syntax `REAL`

RECA

Recalls the indicated internal register. OPC compatible.

Syntax `RECA{1|2|3|4|5}`

Example `RECA1`

RECAREG

Recalls save/recall registers 01 through 31. `RECAREG01` through `RECAREG05` are the same as `RECA1` through `RECA5`.

Syntax `RECAREG`{<*integer*>}

<*integer*> can range from register `01` to `31`.

Example `RECAREG01`

RECO

Recalls previously saved display colors.

Syntax `RECO`

REFD

Completes the reflection calibration subsequence of a 2-port calibration. OPC compatible.

Syntax `REFD`

REFL

Begins the reflection calibration subsequence of a 2-port calibration.

Syntax `REFL`

REFOP

Begins the reflection calibration subsequence for one path, two port calibration.

Syntax REFOP

REFP

Enters the reference position.

Syntax REFP[?| *<number>*]

<number> can range from 0 to 10, where:

0 is the bottom of the graticule.

10 is the top of the graticule.

Query Response *<number>*

REFT

Recall file titles from disk. Requires pass control mode when using the HP-IB port.

Syntax REFT

REFV

Enters the reference line value.

Syntax

`REFV[?|` *<number>*`]`

<number> can range from:

For log magnitude:	±500 dB
For phase:	±500 degrees
For Smith chart and polar:	±500 units
For linear magnitude:	±500 units
For SWR:	±500 units

The scale is always positive, and has minimum values of .001 dB, 10e–12 degrees, 10e–15 seconds, and 10 picounits.

Query Response

<number>

RESC

Resumes cal sequence.

Syntax

`RESC`

RESD

Restores the measurement display after viewing the operating parameters or list values.

Syntax

`RESD`

RESPDONE

Completes the response calibration sequence. OPC compatible.

Syntax `RESPDONE`

REST

Measurement restart.

Syntax `REST`

REV

Calls the reverse calibration classes, during a full 2-port calibration; OPC compatible if there is only one standard in the class.

Syntax `REV{I|M|T}`

Option	Description
`I`	Isolation
`M`	Match
`T`	Transmission

Example `REVI`

RF

Sets RF greater or less than LO in mixer measurements.

Syntax `RF{GTLO|LTLO}`

Option	Description
`GTLO`	Sets RF greater than LO in mixer measurements.
`LTLO`	Sets RF less than LO in mixer measurements.

Example `RFGTLO`

RIG

Selects a plot in the right upper or lower quadrant.

Syntax `RIG{L|U}`

Option	Description
`L`	Selects a plot in the lower right quadrant.
`U`	Selects a plot in the upper right quadrant.

Example `RIGL`

RSCO

Resets colors for the selected group.

Syntax `RSCO`

RST

Presets the instrument. OPC compatible.

Syntax

```
RST
```

S

Selects the S-parameter displayed on the active channel.

Syntax

```
S{11|12|21|22}
```

Option	Description
`11`	Selects the S11 parameter.
`12`	Selects the S12 parameter.
`21`	Selects the S21 parameter.
`22`	Selects the S22 parameter.

Example

```
S22
```

SADD

Adds a new segment to the table, during either a list frequency or limit table edit.

Syntax

```
SADD
```

SAMC

Turns sampler correction on and off. Sampler correction is only turned off to take data for customer calibration coefficients.

Syntax	`SAMC{?|ON|OFF}`
Query Response	`{1|0}`
Example	`SAMCON`

SAV

Completes the port calibration sequence. OPC compatible.

Syntax `SAV{1|2}`

Option	Description
1	Completes the 1-port calibration sequence.
2	Completes the 2-port calibration sequence.

Example `SAV1`

SAVC

Completes the transfer of error correction coefficients back into the instrument. OPC compatible.

Syntax `SAVC`

SAVE

Stores the current instrument state in the indicated internal register. OPC compatible.

Syntax `SAVE{1|2|3|4|5}`

Example `SAVE1`

SAVEOPTK

Save the current definitions of optical standards for the HP 8702A/B/D.

Syntax `SAVEOPTK`

SAVEOPTS

Save the current definitions of optical standards for the HP 8702D.

Syntax `SAVEOPTS`

SAVEREG

Saves to save/recall registers 01 through 31. `SAVEREG01` through `SAVEREG05` are the same as `SAVE1` through `SAVE5`.

Syntax `SAVEREG{<integer>}`

<*integer*> can range from `01` to `31`.

Example `SAVEREG01`

SAVEUSEK

Stores the active calibration kit as the user kit.

Syntax `SAVEUSEK`

SAVU

Selects format for saving to disk.

Syntax `SAVU{ASCI|BINA}`

Option	Description
`ASCI`	Selects ASCII format for saving to disk. Also known as citifile format.
`BINA`	Selects binary format for saving to disk.

Example `SAVUASCI`

SCAL

Sets the trace scale factor.

Syntax `SCAL[?|` *<number>*`]`

<number> can range from:

For log magnitude:	±500 dB
For phase:	±500 degrees
For Smith chart and polar:	±500 units
For linear magnitude:	±500 units
For SWR:	±500 units

The scale is always positive, and has minimum values of .001 dB, 10e–12 degrees, 10e–15 seconds, and 10 picounits.

Query Response *<number>*

SCAP

Selects a full plot, or a plot where the graticule is expanded to P1 and P2.

Syntax `SCAP{FULL|GRAT}`

Example `SCAPFULL`

SDEL

During either a list frequency or a limit table edit, deletes the current segment.

Syntax `SDEL`

SDON

During either a list frequency or a limit table edit, closes a segment after editing.

Syntax `SDON`

SEA

Places the active marker.

Syntax `SEA{L|MAX|MIN|OFF|R|TARG[?|` *<number>*`]}`

Option	Description
`L`	Places the active marker for search left for next occurrence of the target value.
`MAX`	Places the active marker for trace maximum. See also `MARKMAXI`.
`MIN`	Places the active marker for trace minimum. See also `MARKMINI`.
`OFF`	Turns the marker search OFF.
`R`	Places the active marker for search right for next occurrence of the target value.
`TARG`	Places the active marker for arbitrary target amplitude.

<number> can range from:

For log magnitude:	±500 dB
For phase:	±500 degrees
For Smith chart and polar:	±500 units
For linear magnitude:	±500 units
For SWR:	±500 units

The scale is always positive, and has minimum values of .001 dB, 10e–12 degrees, 10e–15 seconds, and 10 picounits.

Query Response <*number*>

SEDI

During either a frequency or a limit table edit, selects the segment for editing.

Syntax `SEDI[?|` <*number*>]

<*number*> can be:

For cal-power meter calibration:	1 to 12
For system-limit testing:	1 to 18
For stimulus menu:	see below

Condition	Range
Frequency sweeps	30 kHz to 3 GHz (to 6 GHz for option 006)
Power sweeps	–15 to 20 dBm in range 0, 25 dB maximum in other ranges (–100 to +100 with power meter cal on)
CW time	0 to 24 hours
Frequency sweep (transform on)	±1/frequency step
CW time sweep (transform on)	±1/time step

Query Response <*number*>

SELL

Selects the learn string revision (LRN) to input to or output from the analyzer.

Syntax `SELL[?|` *<number>*]

<number> can be: 0

Query Response *<number>*

SEQ

Selects sequence 1 through 6.

Syntax `SEQ{1|2|3|4|5|6}`

Example `SEQ1`

SEQWAIT

Tells the instrument to wait a specified number of seconds during a sequence. `NEWSEQ` must precede to ensure that a sequence is currently being created or modified.

Syntax `SEQWAIT[?|` *<number>*]

<number> can range from 01 to 3000 s.

Query Response *<number>*

SETBIT

Sets the specified bit (0 to 7) on the GPIO.

Syntax `SETBIT[?|` <*number*>]

<*number*> can range from 0 to 7.

Query Response <*number*>

SETDATE

Sets the date in the following format: DD MMM YYYY, where DD is the day and must be 2 digits, MMM is the month and must be three alpha characters (JAN, FEB, MAR, APR, MAY, JUN, JUL, AUG, SEP, OCT, NOV, DEC), and YYYY is the year and must be 4 digits.

Syntax `SETDATE` [<*string*>]

<*string*> value is: DD MMM YYYY

SETF

Sets frequency for low pass transform. Not available with Option 110.

Syntax `SETF`

SETTIME

Sets the time in the following format: HH:MM:SS, where HH is the hour, MM is minutes, SS is seconds, and each must be 2 digits.

Syntax `SETTIME` [<*string*>]

<*string*> value is: HH:MM:SS

SETZ

Set the characteristic impedance of the measurement system.

Syntax `SETZ`[`?|` <*number*>]

<*number*> can range from 0.1 to 500Ω.

Query Response <*number*>

SHOM

Displays the desired softkey menu during a sequence. `NEWSEQ` must precede to ensure that a sequence is currently being created or modified.

Syntax `SHOM`

SING

Sets to single sweep. OPC compatible.

Syntax `SING`

SLID

Sliding load done.

Syntax `SLID`

SLIL

Specifies the standard as a sliding load during a standard definition as part of a cal kit modification.

Syntax `SLIL`

SLIS

Sliding load set.

Syntax `SLIS`

SLOPE

Enters the power slope value.

Syntax `SLOPE[?|` <*number*>]

<*number*> can range from –2 to 2 dB/GHz.

Query Response <*number*>

SLOPO

Turns the power slope on and off.

Syntax `SLOPO{?|ON|OFF}`

Query Response `{1|0}`

Example `SLOPOON`

SMIC

Selects Smith chart display format.

Syntax `SMIC`

SMIM

Selects the marker readout format on a Smith chart.

Syntax `SMIM{GB|LIN|LOG|RI|RX}`

Option	Description
`GB`	Selects the G+jB marker readout format on a Smith chart.
`LIN`	Selects the linear marker readout format on a Smith chart.
`LOG`	Selects the log marker readout format on a Smith chart.
`RI`	Selects the real/imaginary pairs marker readout format on a Smith chart.
`RX`	Selects the R+jX marker readout format on a Smith chart.

Example `SMIMGB`

SMOOAPER

Sets the smoothing aperture as a percent of the trace.

Syntax `SMOOAPER[?|` *<number>*]

<number> can range from 0.05 to 20%.

Query Response *<number>*

SMOOO

Turns smoothing on and off.

Syntax `SMOOO{?|ON|OFF}`

Query Response {`1|0`}

Example `SMOOOON`

SO

Calculates the values for source cal standard and stores it in trace memory for the active channel.

Syntax `SO{UCSTDI|UDSTDI|2DSTDI}`

Option	Description
`UCSTDI`	Calculates the source coefficient standard and copies into memory.
`UDSTDI`	Calculates disk cal standard and copies into memory.
`2DSTDI`	Calculates disk cal standard and copies into memory.

Example `SOUCSTDI`

SOFR

Displays the firmware revision on the screen.

Syntax `SOFR`

SOFT

Presses the indicated softkey.

Syntax `SOFT{1|2|3|4|5|6|7|8}`

Example `SOFT1`

SOUP

Turns the source power on and off.

Syntax `SOUP{?|ON|OFF}`

Query Response `{1|0}`

Example `SOUPON`

SPAN

Sets the stimulus span. If a list frequency segment is being edited, sets the span of the segment.

Syntax `SPAN[?|` <*number*>]

<*number*> values may be:

Condition	Range
Frequency sweeps	30 kHz to 3 GHz (to 6 GHz for option 006)
Power sweeps	–15 to 20 dBm in range 0, 25 dB maximum in other ranges (–100 to +100 with power meter cal on)
CW time	0 to 24 hours
Frequency sweep (transform on)	±1/frequency step
CW time sweep (transform on)	±1/time step

Query Response <*number*>

SPEC

Initiates the SPECIFY CLASS part of modifying a cal kit.

Syntax

`SPEC{<`*option*>}[<*integer*>,<*integer*>,. . .]

Option	Description
`FWDM`	Forward match
`FWDT`	Forward transmission
`RESP`	Response
`RESI`	Resp & Isol, response
`REVM`	Reverse match
`REVT`	Reverse transmission
`S11A`	S11A (opens)
`S11B`	S11B (shorts)
`S11C`	S11C (loads)
`S22A`	S22A (opens)
`S22B`	S22B (shorts)
`S22C`	S22C (loads)
`TRFM`	TRL, Reflect, Forward, Match
`TRRM`	TRL, Reflect, Reverse, Match
`TLFM`	TRL, Line, Forward, Match
`TLFT`	TRL, Line, Forward, Trans
`TLRM`	TRL, Line, Reverse, Match
`TLRT`	TRL, Line, Reverse, Trans
`TTFM`	TRL, Thru, Forward, Match
`TTFT`	TRL, Thru, Forward, Trans
`TTRM`	TRL, Thru, Reverse, Match
`TTRT`	TRL, Thru, Reverse, Trans

<*integer*> is a standard number.

Description

After issuing each command, sends the analyzer a series of standard numbers to be included in the class, separated by commas. When the class is full, sends `CLAD` to terminate the sequence.

SPEG

Displays the specify gate menu.

Syntax `SPEG`

SPLD

Turns the split display mode on and off.

Syntax `SPLD{?|ON|OFF}`

Query Response `{1|0}`

Example `SPLDON`

SRE

Service request enable. A bit set in D enables the corresponding bit in the status byte to generate an SRQ.

Syntax `SRE[?|` *<number>*]

<number> can range from 0 to 255.

Query Response *<number>*

SSEG

Selects the desired segment of the frequency list for a list frequency sweep.

Syntax

`SSEQ`[`?|` <*number*>]

<*number*> can range from 1 to 30.

Query Response

<*number*>

ST

Sets the frequency stimulus when the time domain transform is on. Results may be viewed using the transform parameter feature. Not available with Option 110.

Syntax

`ST{AF|OF}`

Option	Description
`AF`	Sets the start frequency.
`OF`	Sets the stop frequency.

Example

`STAF`

STAN

Selects a standard from a class during a calibration sequence. All of these commands are OPC compatible.

Syntax

```
STAN{A|B|C|D|E|F|G}
```

Option	Description
`A`	Standard listed under softkey 1.
`B`	Standard listed under softkey 2.
`C`	Standard listed under softkey 3.
`D`	Standard listed under softkey 4.
`E`	Standard listed under softkey 5.
`F`	Standard listed under softkey 6.
`G`	Standard listed under softkey 7.

Description

If a class is requested, as in `CLASS11A` (open, S11 1-port cal), the analyzer will do one of two things. If there is only one standard in the class, it will measure that standard automatically. If there are several standards in the class, then one of the commands must be used to select one, causing it to be measured.

Example

```
STANA
```

STAR

Enters the start stimulus value. If a list frequency segment is being edited, sets the start of the segment.

Syntax `STAR[?|` *<number>*]

<number> values may be:

Condition	Range
Frequency sweeps	30 kHz to 3 GHz (to 6 GHz for option 006)
Power sweeps	–15 to 20 dBm in range 0, 25 dB maximum in other ranges (–100 to +100 with power meter cal on)
CW time	0 to 24 hours
Frequency sweep (transform on)	±1/frequency step
CW time sweep (transform on)	±1/time step

Query Response *<number>*

STB?

Outputs the status byte.

Syntax `STB?`

STDD

Standard done, define standard sequence, while modifying a cal kit.

Syntax `STDD`

STDT

Selects the standard type after the standard number has been entered during a modify cal kit sequence.

Syntax `STDT{ARBI|DELA|LOAD|OPEN|SHOR}`

Option	Description
`ARBI`	Selects the standard type of arbitrary impedance.
`DELA`	Selects the standard type of delay/thru.
`LOAD`	Selects the standard type of load.
`OPEN`	Selects the standard type of open.
`SHOR`	Selects the standard type of short.

Example `STDTARBI`

STDTFRES

Specifies that a Fresnel optical standard is being defined.

Syntax `STDTFRES`

STDTOTHR

Specifies that a thru optical standard is being defined.

Syntax `STDTOTHR`

STDTRECE

Specifies that the receiver coefficient cal standard is being defined.

Syntax `STDTRECE`

STDTREFL

Specifies that a reflector optical standard is being defined.

Syntax `STDTREFL`

STDTSOUR

Specifies that the source coefficient cal standard is being defined.

Syntax `STDTSOUR`

STDTTHRR

Specifies that the thru standard used as part of a thru/receiver is being defined; thru/receiver is a cal standard available during calibration of E/O devices.

Syntax `STDTTHRR`

STOP

Sets the stimulus stop value. If a list frequency segment is being edited, sets the stop of the segment.

Syntax `STOP[?|` *<number>*`]`

<number> values may be

Condition	Range
Frequency sweeps	30 kHz to 3 GHz (to 6 GHz for option 006)
Power sweeps	–15 to 20 dBm in range 0, 25 dB maximum in other ranges (–100 to +100 with power meter cal on)
CW time	0 to 24 hours
Frequency sweep (transform on)	±1/frequency step
CW time sweep (transform on)	±1/time step

Query Response *<number>*

STOR

Stores the indicated file on disk.

Syntax `STOR{1|2|3|4|5}`

Option	Description
`1`	Stores the current instrument state to disk using the file name provided by the preceding `TITF1` command.
`2`	Stores the current instrument state to disk using the file name provided by the preceding `TITF2` command.
`3`	Stores the current instrument state to disk using the file name provided by the preceding `TITF3` command.
`4`	Stores the current instrument state to disk using the file name provided by the preceding `TITF4` command.
`5`	Stores the current instrument state to disk using the file name provided by the preceding `TITF5` command.

Description Used with the `INTD` and `EXTD` commands to designate the internal or external disk. Requires pass control mode when using the HP-IB port.

Example `STOR1`

STORSEQ

Stores the instrument state to the indicated sequence to disk.

Syntax `STORSEQ{1|2|3|4|5|6}`

Description Used with the `INTD` and `EXTD` commands to designate the internal or external disk. Requires pass control mode when using the HP-IB port.

Example `STORSEQ1`

STPSIZE

While editing a list frequency segment, sets step size.

Syntax

`STPSIZE[?|` <*number*>]

<*number*> values may be:

Condition	Range
Frequency sweeps	30 kHz to 3 GHz (to 6 GHz for option 006)
Power sweeps	–15 to 20 dBm in range 0, 25 dB maximum in other ranges (–100 to +100 with power meter cal on)
CW time	0 to 24 hours
Frequency sweep (transform on)	±1/frequency step
CW time sweep (transform on)	±1/time step

Query Response

<*number*>

SVCO

Saves display colors.

Syntax

`SVCO`

SWET

Sets the sweep time. Sets to 0 for auto sweep time.

Syntax `SWET[?|` *<number>*`]`

<number> can range from 0 to 86,400 s.

Query Response *<number>*

SWR

Selects the SWR display format.

Syntax `SWR`

TAKCS

Begins a power meter calibration sweep. Requires pass control.

Syntax `TAKCS`

TALKLIST

Puts the analyzer in talker/listener mode.

Syntax `TALKLIST`

TERI

Specifies the terminal impedance of an arbitrary impedance standard during a cal kit modification.

Syntax `TERI[?|` *<number>*]

<number> can range from 0 to 1 kΩ.

Query Response *<number>*

TESS?

Returns a one on the standard analyzer. (The standard analyzer does not recognize external test sets.) For Option 011, returns a one if an HP 85046A/B S-parameter test set is present. Returns a two if an HP 85047A S-parameter test set is present.

Syntax `TESS?`

TIMDTRAN

Turns the time domain transform on and off. Not available with Option 110.

Syntax `TIMDTRAN{?|ON|OFF}`

Query Response {`1|0`}

Example `TIMDTRANON`

TIMESTAM

Turns on the clock time for prints and plots.

Syntax `TIMESTAM{?|ON|OFF}`

Query Response `{1|0}`

Example `TIMESTAMON`

TINT

Adjusts the tint for the selected display feature.

Syntax `TINT{?|` *<number>*]

<number> can range from 0 to 100.

Query Response *<number>*

TITF

Titles the indicated file number.

Syntax `TITF{1|2|3|4|5}` [*<string>*]

<string> can be a maximum of 10 characters.

TITF0

Titles the SAVE STATE filename in sequence mode.

Syntax `TITF0`

TITL

Enters a new CRT title. A maximum of 48 characters (alphanumeric and mathematical symbols) are allowed.

Syntax `TITL` [<*string*>]

<*string*> can be a maximum of 48 characters.

TITP

Titles the plot file.

Syntax `TITP`

TITR

Titles the indicated internal register.

Syntax `TITR`{`1|2|3|4|5`} [<*string*>]

<*string*> can be a maximum of 10 characters.

TITREG

Titles save/recall registers 01 through 31. `TITREG01` through `TITREG05` are the same as `TITR1` through `TITR5`.

Syntax

`TITREG`{<*integer*>} [<*string*>]

<*integer*> can range from register 01 to 31.

<*string*> can be a maximum of 10 characters.

TITSEQ

Selects the sequence to be titled.

Syntax

`TITSEQ`{`1|2|3|4|5|6`}

Example

`TITSEQ1`

TITT

Sends the title string. `NEWSEQ` must precede to ensure that a sequence is currently being created or modified.

Syntax

`TITT{MEM|PMTR|PERI|PRIN}`

Option	Description
`MEM`	Sends the title string to trace memory.
`PMTR`	Sends the title string to the power meter address.
`PERI`	Sends the title string to the peripheral address.
`PRIN`	Sends the title string to the printer address.

Example

`TITTMEM`

TRACK

Turns marker search tracking on and off.

Syntax `TRACK{?|ON|OFF}`

Query Response `{1|0}`

Example `TRACKON`

TRAD

Completes the transmission calibration subsequence of a 2-port calibration. OPC compatible.

Syntax `TRAD`

TRAN

Begins the transmission calibration subsequence of a 2-port calibration.

Syntax `TRAN`

TRAOP

Begins the transmission calibration subsequence for one path, two port calibration.

Syntax `TRAOP`

TRAP

Displays tranform parameter screen on the instrument display. Not available with Option 110.

Syntax `TRAP`

TRAS

Sets the transform span when the time domain transform is on. The results may be viewed with the transform parameter feature. Not available with Option 110.

Syntax `TRAS`

TRIG

HP-IB trigger. Puts instrument into hold mode.

Syntax `TRIG`

TST?

Causes a self test and returns a zero if the test is passed.

Syntax `TST?`

TSTIO

Defines 3 bits, D0 through D2, on the test set connector I/O for the channel 1 and channel 2 forward or reverse settings. These bits can be set to values of 0 through 7. Be careful that you do not also set a value to ATTP1 and ATTP2 as there is interraction between these commands and they will change the values you have set for D0 through D2 and will couple the channels together. Values for ATTP1 and ATTP2 translate to the values for D0 through D2 as shown in the table below.

Syntax

`TSTIO{FWD|REV}[?| <`*number*`>]`

<*number*> can range from 0 to 7.

ATTP1/ATTP2	D0 - D2
0 dB	7
10 dB	6
20 dB	5
30 dB	4
40 dB	3
50 dB	2
60 dB	1
70 dB	0

Query Response <*number*>

Example

```
TSTIOFWD3
```

TSTPP

Selects test port 1 or 2 for non-S-parameter measurements.

Syntax `TSTPP{1|2}`

Example `TSTPP1`

TTL

Sets the rear panel TEST SEQ BNC output at end of sweep.

Syntax `TTL{HPULS|LPULS|OH|OL}`

Option	Description
`HPULS`	Sets the rear panel TEST SEQ BNC output normally low, high pulse at end of sweep.
`LPULS`	Sets the rear panel TEST SEQ BNC output normally high, low pulse at end of sweep.
`OH`	Sets the rear panel TEST SEQ BNC output continuously high.
`OL`	Sets the rear panel TEST SEQ BNC output continuously low.

Example `TTLHPULS`

UCONV

Selects up converter for mixer measurements.

Syntax `UCONV`

UP

Increments the value in the active entry area (up key).

Syntax `UP`

USEPASC

Puts the analyzer in pass control mode.

Syntax `USEPASC`

USESENS

Selects the sensor input being used with the HP 438A power meter. For the HP 436A or 437B, the A sensor is always used.

Syntax `USESENS{A|B}`

Example `USESENSA`

VELOFACT

Enters the velocity factor of the transmission medium.

Syntax `VELOFACT[?|` *<number>*]

<number> can range from 0 to 10.

Query Response *<number>*

VIEM

Displays the measurement trace (ON) or the mixer setup (OFF).

Syntax `VIEM{?|ON|OFF}`

Query Response {`1|0`}

Example `VIEMON`

VOFF

Sets the frequency offset value.

Syntax `VOFF`[`?|` *<number>*]

<number> can range from 0 to 3 GHz (0 to 6 GHz for Option 006)

Query Response *<number>*

WAIT

Waits for a clean sweep. OPC compatible.

Syntax `WAIT`

WAVE

Specifies a waveguide standard while defining a standard as part of a cal kit modification.

Syntax `WAVE`

WIDT

Turns the bandwidth search on and off.

Syntax `WIDT{?|ON|OFF}`

Query Response `{1|0}`

Example `WIDTON`

WIDV

Enters the width search parameter.

Syntax `WIDV[?|` *<number>*]

<number> can range from:

For log magnitude:	±500 dB
For phase:	±500 degrees
For Smith chart and polar:	±500 units
For linear magnitude:	±500 units
For SWR:	±500 units

The scale is always positive, and has minimum values of .001 dB, 10e–12 degrees, 10e–15 seconds, and 10 picounits.

Query Response *<number>*

WIND

Sets the window for the time domain transform. Not available with Option 110.

Syntax `WIND{MAXI|MINI|NORM}`

Option	Description
`MAXI`	Sets the window to maximum for the transform.
`MINI`	Sets the window to minimum for the transform.
`NORM`	Sets the window to normal for the transform.

Example `WINDMAXI`

WINDOW

Enters arbitrary window for the time domain transform. Not available with Option 110.

Syntax `WINDOW[?|` *<number>*]

<number> value is state-dependent.

Query Response *<number>*

WINDUSEM

Turns the trace memory ON as the window shape. Not available with Option 110.

Syntax `WINDUSEM{?|ON|OFF}`

Query Response {`1|0`}

Example `WINDUSEMON`

WRSK

Enters new softkey labels into the indicated softkey positions.

Syntax `WRSK{1|2|3|4|5|6|7|8}` [*<string>*]

<string> can be a maximum of 10 characters.

4

Graphics Language Reference

Graphics Language Reference

This chapter is the reference for all display graphics programming commands. Commands are listed alphabetically. Refer to “Drawing Graphics on the Display” on page 1-33 for more information.

Table 4-1. Notation Conventions and Definitions

Convention	Description
< >	Angle brackets indicate values entered by the programmer.
|	”Or” indicates a choice of one element from a list.
[ ]	Square brackets indicate that the enclosed items are optional.
{}	When several items are enclosed by braces, one, and only one of these elements must be selected.
<number>	A numerical operand.
<integer>	An ASCII string representing an integer. This is defined by the IEEE 488.2 <NR1> format.
<string>	A character-string operand which must be enclosed by quotes.

AF

Erases the user graphics display.

Syntax `AF`

See Also `PG`

CS

Turns off the measurement display.

Syntax `CS`

See Also `RS`

DF

Sets the default values.

Syntax DF

Description The following table shows the default graphics state set by this command.

Table 4-2. Default Graphics State

Feature	Description	Equivalent Command
Displayed graphics	erased	AF
Measurement display	off	CS
Pen intensity	normal	SP2
Pen state	up	PU
Line style	normal	LT1
Label direction	left to right	DI1,0
Point interpretation	absolute	PA
Character size	medium	SI0.25,0.30

DI

Specifies the direction in which characters are lettered.

Syntax `DI<run>,<rise>`

When the DI and SI instructions are used together, the DI instruction establishes a label's direction and the SI instruction establishes its size. As shown in the following table, text can only be drawn at 90° increments from horizontal. Use the LB command to actually draw the text.

Table 4-3. Text Options

Text Direction	<run>,<rise>
0 degrees	1,0
90 degrees	0,1
180 degrees	–1,0
270 degrees	0,–1

Example `DI0,1`

See Also `LB`, `SI`

LB

Writes text on the display starting at the current pen position.

Syntax `LB<string><etx>`

Description The label string sent with the command must be terminated with the ASCII end of text (ETX) character. The decimal value of the ETX character is 3 (not the character 3). Refer to the manual for the text editor you are using to determine how to enter the ETX character. The following are some hints that may help you to enter this character. On a PC, the ETX character is displayed as a heart symbol (♥) and can sometimes be entered by pressing the control key with the letter “c”. If you are programming using HP BASIC, you can use the `CHR$(3)` string to enter the ETX character. This is shown in the example for this command.

Use the SI command to specify the size of the text.

Example `“LBHello World!” + CHR$(3)`

See Also `SI`

LT

Specifies the style of line that will be drawn with the PA and PR commands.

Syntax `LT{0|1|2|3}`

Option	Line Style
0	solid
1	solid
2	short dashes
3	long dashes

See Also PA, PR

OP

Outputs the scaling limits of the display.

Syntax `OP`

Description The X and Y coordinates for two points are returned to the computer. These points represent the extreme limits for drawing on the display. The first point is located at the origin (0, 0) which is the display's lower left corner. The second point is (5850, 4095) which is the display's upper right corner.

Query Response `0,0,5850,4095`

PA

Draws a line to an absolute point on the display.

Syntax `PA[<x>,<y>, . . . ]`

Description This command establishes absolute drawing mode and moves the pen to the specified point(s). You can include as many X- and Y-coordinate pairs as needed. Each value must be separated by a comma character. When you include more than one coordinate pair, the pen moves to each point in the order given, using the current up or down pen status. Use the pen up instruction, PU, to move to a point without drawing, or the pen down instruction, PD, to draw to a point.

The following example draws a small rectangle on the display.

Example

```
PU;PA100,200;PD;PA200,200,200,300,100,300,100,200;
```

PD

Places the imaginary pen down.

Syntax PD

Description When the pen is down, text and lines can be drawn using the LB, PA, and PR commands. If these items are not displayed after the PD command is given, make sure that the pen value has not been set to zero using the SP command.

See Also PU

PG

Erases the user graphics display.

Syntax PG

See Also AF

PR

Draws a line to a point on the display that is relative to the current pen position.

Syntax

```
PR[<x>,<y>, . . . ]
```

Description

This command establishes relative drawing mode and moves the pen to the specified point(s), relative to the current pen position. You can include as many X- and Y-coordinate pairs as needed. Each value must be separated by a comma character. When you include more than one coordinate pair, the pen moves to the first point using the current up or down pen status. This point pen becomes the current pen position for the next coordinate pair. In this manner, the pen moves to each point in the order given.

Use the pen up instruction, `PU`, to move to a point without drawing, or the pen down instruction, `PD`, to draw to a point.

As the example for this command shows, negative values can be entered. This example draws a small rectangle on the display.

Example

```
PU;PA100,200;PD;PR100,0,0,100,-100,0,0,-100;
```

PU

Places the imaginary pen up.

Syntax

```
PU
```

Description

When the pen is in the up position, nothing will be drawn when using the `LB`, `PA`, and `PR` commands.

See Also

`PD`

RS

Turns on the measurement display.

Syntax `RS`

See Also `CS`

SI

Sets the size of characters drawn using the `LB` command.

Syntax `SI<height>,<width>`

Description Use this command to set character height in width in centimeters. Four settings are available as shown in the following table. Alternate settings are not allowed.

When the `DI` and `SI` instructions are used together, the `DI` instruction establishes a label's direction and the `SI` instruction establishes its size. Use the `LB` command to actually draw the text.

Table 4-4. Text Size Options

Height	Width
0.16	0.20
0.25	0.30
0.33	0.39
0.41	0.49

See Also `LB`, `DI`

SP

Selects the intensity of the pen.

Syntax

```
SP{0|1|2|3}
```

Option	Brightness
`0`	blank
`1`	bright
`2`	normal
`3`	dim

Description

The pen setting sets the intensity that text and lines are drawn with. When drawing a graphic, you can change pens in order to change the intensity of different objects.

5

Tables and Charts

Tables and Charts

What you'll find in this chapter

HP-IB Requirements

Table 5-1. HP-IB Requirements

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

HP-IB operational capabilities

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

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

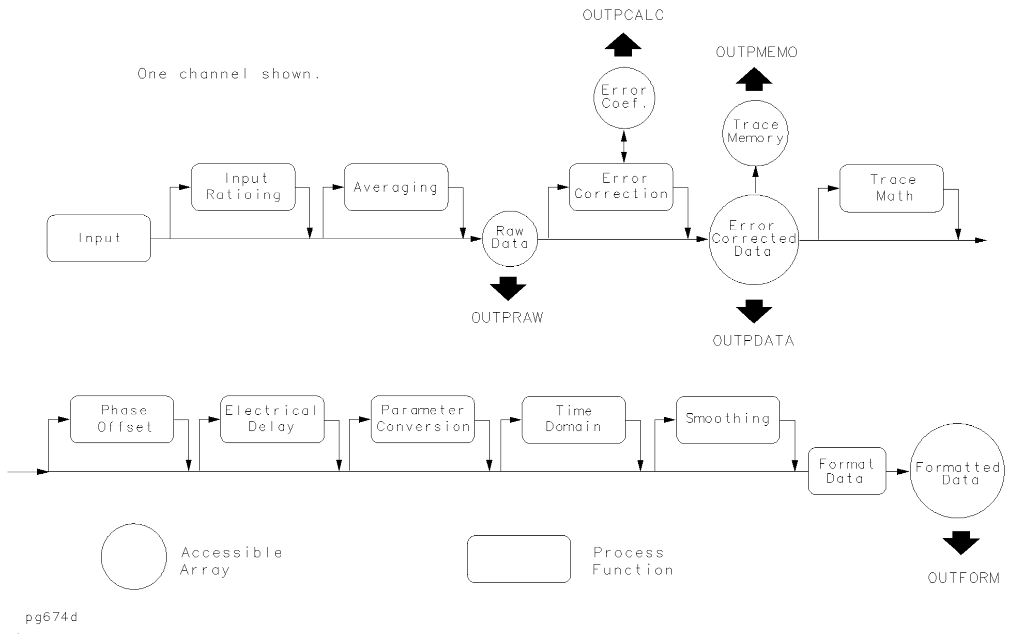


Figure 5-1. Data processing chain

Table 5-2. Units as a Function of Display Format

Display Format	Marker Mode	OUTPMARK		OUTPFORM		MARKER READOUT[a]	
		value 1	value 2	value 1	value 2	value	aux value
LOG MAG		dB	[b]	dB	[b]	dB	[b]
PHASE		degrees	[b]	degrees	[b]	degrees	[b]
DELAY		seconds	[b]	seconds	[b]	seconds	[b]
SMITH CHART	LIN MKR	lin mag	degrees	real	imag	lin mag	degrees
	LOG MKR	dB	degrees	real	imag	dB	degrees
	Re/Im	real	imag	real	imag	real	imag
	R + jX[c]	real ohms	imag ohms	real	imag	real ohms	imag ohms
	G + jB	real Siemens	imag Siemens	real	imag	real Siemens	imag Siemens
POLAR	LIN MKR	lin mag	degrees	real	imag	lin mag	degrees
	LOG MKR	dB	degrees	real	imag	dB	degrees
	Re/Im	real	imag	real	imag	real	imag
LIN MAG		lin mag	[b]	lin mag	[b]	lin mag	[b]
REAL		real	[b]	real	[b]	real	[b]
SWR		SWR		SWR	[b]	SWR	[b]

a. The marker readout values are the marker values displayed in the upper right-hand corner of the display. They also correspond to the value and auxiliary value associated with the fixed marker.

b. Value 2 is not significant in this format, though it is included in data transfers.

c. To read the OUTPFORM for R + jX data, use the following key presses: FORMAT, *POLAR*, MEAS, *MORE*, *CONVERSION*, *Z REFL*, or the following commands over HP-IB: POLAj CONVZREFj.

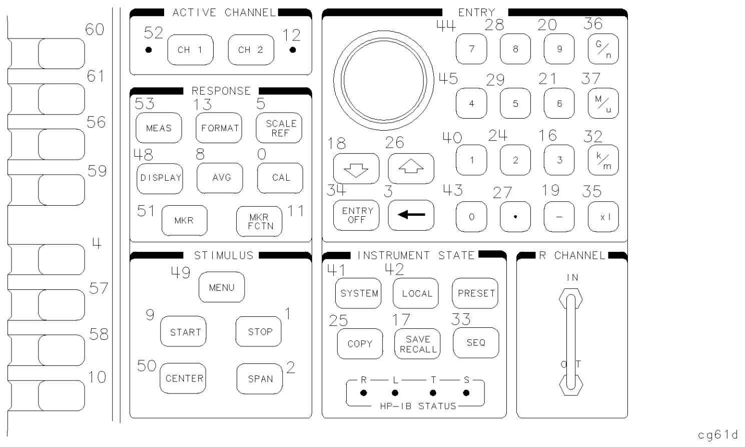


Figure 5-2. Key codes

Note 1:	Key code 63 is invalid key.
Note 2:	`OUTPUT`; reports a knob turn as a –1.
Note 3:	If the two byte integer sent back from `KOR?` is negative, it is a knob count. Decode the knob count and direction by setting bit 15 equal to bit 14. A positive result indicates counterclock wise rotation.

Programming Commands by Functional Group

Table 5-3. Programming Commands (1 of 18)

Command	Description
Averaging	
AVERFACT	Sets the averaging factor on the active channel.
AVERO	Turns averaging ON and OFF on the active channel.
AVERREST	Restarts the averaging on the active channel.
IFBW	Sets the IF bandwidth.
SMOOAPER	Sets the smoothing aperture as a percent of the trace.
SMOOO	Turns smoothing ON and OFF.
CAL-error correction, calibration	
ALTAB	Places the analyzer in the alternate inputs measurement mode, where inputs A and B are measured on alternate sweeps.
CALI	Begins a calibration sequence.
CALN	Turns calibration type to off. Calibration: none.
CHOPAB	Places the analyzer in the cop measurement mode.
CLASS	Calls reflection standard class during a calibration sequence.
CORI	Turns interpolative error correction ON and OFF.
CORR	Turns error correction ON and OFF.
CSWI	Selects test set continuous switching (ON) or test set hold (OFF) when there is a 2-port calibration active.
DONE	Completes a class of standards, during a calibration.
FWD	Selects a forward calibration class, during a 2-port calibration sequence.
ISO	Completes an isolation subsequence.
OMII	Omits the isolation step of a calibration sequence.
PORE	Turns port extensions ON and OFF.
PORT	Sets the port extension length for a test port.

Table 5-3. Programming Commands (2 of 18)

Command	Description
RAI	Calls the response or isolation class for the response and isolation calibration. OPC-compatible if only one standard in class.
RAID	Completes the response and isolation calibration sequence.
REFD	Completes the reflection calibration subsequence of a 2-port calibration.
REFL	Begins the reflection calibration subsequence of a 2-port calibration.
REFOP	Begins the reflection calibration subsequence for one path, 2-port calibration.
RESC	Resumes cal sequence.
RESPDONE	Completes the response calibration sequence.
REV	Selects a reverse calibration class, during a 2-port calibration sequence.
SAV	Completes a 1-port or 2-port calibration sequence.
SETZ	Sets the characteristic impedance of the measurement system.
SLID	Sliding load done.
SLIS	Sliding load set.
STAN	Selects a standard during a calibration sequence.
TRAD	Completes the transmission calibration subsequence of a 2-port calibration.
TRAN	Begins the transmission calibration subsequence of a 2-port calibration. OPC-compatible.
TRAOP	Begins the transmission calibration subsequence for one path, 2-port calibration.
VELOFACT	Enters the velocity factor of the transmission medium.
CAL-calibration kits	
C	Selects open capacitance values of an open circuit while it is being defined as a calibration standard.
CALK	Selects a default calibration kit.
CLAD	Class done, modify calibration kit, specify class.
COAX	Selects coaxial offsets instead of waveguide while defining a standard during a calibration kit modification.
DEFS	Begins standard definition during calibration kit modification.
FIXE	Specifies a fixed load, as opposed to a sliding load, when defining a standard during a calibration kit modification.

Table 5-3. Programming Commands (3 of 18)

Command	Description
KITD	Calibration kit done: the last step in modifying a calibration kit.
LAB	Enters a label during a cal kit modification.
MAXF	Sets the maximum valid frequency of a standard being defined during a calibration kit modification.
MINF	Sets the minimum valid frequency of a standard being defined during a calibration kit modification.
MODI1	Begins the modify calibration kit sequence.
OFS	Specifies an offset value for a standard being defined during a calibration kit modification.
SAVEUSEK	Stores the active calibration kit as the user kit.
SLIL	Specifies the standard as a sliding load during a standard definition as part of a cal kit modification.
SPEC	Initiates a class for modifying a calibration kit.
STDD	Standard done, defines standard sequence, while modifying a calibration kit.
STDT	Selects the standard after the standard number has been entered during a modify calibration kit sequence.
TERI	Specifies the terminal impedance of an arbitrary impedance standard during a calibration kit modification.
WAVE	Specifies a waveguide standard while defining a standard as part of a calibration kit modification.
CAL-power meter calibration	
CALFCALF	Sets the calibration factor.
CALFFREQ	Selects the frequency for the calibration factor correction.
CALFSEN	Edits the sensor A or sensor B calibration factor table.
CLEL	Clears the desired list.
EDITDONE	Done editing list frequency or limit table.
NUMR	Sets the number of power meter readings per point used during a power meter calibration.
POWLFREQ	Selects the frequency for which a power loss correction is entered. This must be followed by a POWLLOSS command, which sets the value.
POWLLIST	Begins editing a power loss list for a power meter calibration.

Table 5-3. Programming Commands (4 of 18)

Command	Description
POWLLOSS	Sets the loss value for a particular frequency, POWLFREQ, in the power loss list.
PWMC	Selects the type of power meter calibration desired.
PWRLOSS	Selects whether or not to use the power loss table for a power meter calibration.
PWRMCAL	Displays the power meter calibration menu and sets the drive port calibration power.
SADD	Adds a new segment to the table during either a list frequency or limit table edit.
SDEL	Deletes the current segment during either a list frequency or a limit table edit.
SDON	Closes a segment after editing during either a list frequency or a limit table edit.
SEDI	Selects a segment for editing during either a frequency or a limit table edit.
TAKCS	Begins a power meter calibration sweep. Requires pass control.
USESENS	Selects the sensor input being used with the HP 438A power meter. For the HP 436A or 437B, the A sensor is always used.
CHANNEL	
CHAN	Makes channel 1 or channel 2 the active channel.
COPY	
DEFLPRINT	Sets the printer default setup conditions.
DFLT	Sets the plotter default setup conditions.
FULP	Selects full page plotting, as opposed to plotting in one of the four quadrants.
LEF	Selects a plot in the left lower or upper quadrant.
LINT	Enters the line type for plotting data or memory.
LISV	Activates the list values function.
NEXP	Displays the next page of the operating parameters list.
OPEP	Presents a list of key operating parameters.
OUTP	Outputs to the HP-IB ports. Can be directed to a plotter, or read into the computer.
PCOL	Selects the color for the indicated feature.
PDATA	Selects whether trace data is plotted.
PENN	Selects the pen for plotting the indicated display feature.
PGRAT	Selects whether the graticule is plotted.
PLOS	Selects the pen speed for plotting.

Table 5-3. Programming Commands (5 of 18)

Command	Description
PLOT	Initiates a plot. Requires pass control mode when using the HP-IB port.
PLTTRAUTF	Turns the plotter auto feed on or off.
PLTTRFORF	Sends a form feed to the plotter.
PMEM	Selects whether memory is plotted.
PMKR	Selects whether markers are plotted.
PRIC	Selects color print.
PRIN	Begins printing the format selected. Requires pass control mode when using the HP-IB port.
PRIS	Selects standard (monochrome) print.
PRNTRAUTF	Turns the printer auto feed on or off.
PRNTRFORF	Sends a form feed to the printer.
PTEXT	Selects whether text is plotted.
RESD	Restores the measurement display after viewing the operating parameters or list values.
RIG	Selects a plot in the right lower or upper quadrant.
SCAP	Selects a full plot, or a plot where graticule is expanded to P1 and P2.
DISPLAY	
BACI	Sets the background intensity of the display.
BEEP	Turns sound on or off for completion of functions.
CBRI	Adjusts the color brightness of the selected display feature.
COLO	Selects a display feature for color modification.
COLOR	Adjusts the color saturation for the selected display feature.
D1DIVD2	Divides the data in channel 1 by the data in channel 2 and displays the result on channel 2.
DATI	Stores trace in channel memory.
DEFC	Sets the default colors for all display features.
DISP	Displays data and/or memory on the active channel.
DUAC	Turns dual channel display on or off.

Table 5-3. Programming Commands (6 of 18)

Command	Description
FOCU	Adjusts display focus.
FREO	Turns frequency notation off, frequency blank.
INTE	Sets the display intensity.
RECO	Recalls previously saved display colors.
RSCO	Resets colors for the selected group.
SPLD	Turns the split display mode on or off.
SVCO	Saves display colors.
TINT	Adjusts the tint for the selected display feature.
TITL	Enters a new CRT title.
Entry of Data	
DOWN	Decrements the value in the active entry area (down key).
ENTO	Turns the active entry area off.
UP	Increments the value in the active entry area (up key).
Format of Display	
DELA	Displays the data formatted as group delay.
IMAG	Selects the imaginary display format.
LIN	Selects the linear value.
LOG	Selects the log value.
PHAS	Selects the phase display format.
POLA	Selects the polar display format.
REAL	Selects the real display format.
SMIC	Selects Smith chart display format.
SWR	Selects the SWR display format.
Graphics	
INPUDATA	Accepts error-corrected data.
Input of Data	
INPUCALC	Inputs individual calibration coefficient arrays.

Table 5-3. Programming Commands (7 of 18)

Command	Description
INPUCALK	Inputs a cal kit read out with OUTPCALK.
INPUDATA	Inputs error corrected data array using current format.
INPUFORM	Inputs a formatted data array using current format.
INPULEAS	Inputs a learn string read out by OUTPLEAS.
INPUPMCAL	Inputs power meter cal array into the instrument.
INPURAW	Inputs a raw data array using current format.
Interface Bus, Printing, and Plotting Control	
ADDR	Plotter HP-IB address.
DEBU	Turns the HP-IB debug mode on or off.
DISCUNIT	Specifies which disk in a multiple-disk drive is to be used for disk registers.
DISCVOLU	Specifies which volume of a multiple-volume disk drive is to be used for disk registers.
PARAL	Selects use of the parallel port for general purpose I/O or for the copy function.
PLTHNDSHK	Selects the plotter handshake mode as either Xon-Xoff or DTR-DSR, for serial (RS-232) bus.
PLTPRT	Sets the plotter port.
PLTTRAUTF	Turns on and off the plotter auto feed.
PLTTRBAUD	Sets the plotter baud rate, for serial (RS-232) bus.
PLTTRFORF	Sends a form feed to the plotter.
PLTTYP	Selects the plotter type.
POWM	Selects whether the HP 436A (on) or the HP 437B/438A (off) is to be used as the power meter in service procedures.
PRNHNDSHK	Selects the printer handshake mode as either Xon-Xoff or DTR-DSR, for serial (RS-232) bus.
PRNPRT	Sets the printer port.
PRNTRAUTF	Turns on and off the printer auto feed.
PRNTRBAUD	Sets the printer baud rate.
PRNTRFORF	Sends a form feed to the printer.
PRNTYP	Selects the printer type.

Table 5-3. Programming Commands (8 of 18)

Command	Description
TALKLIST	Puts the analyzer in talker listener mode.
USEPASC	Puts the analyzer in pass control mode.
Measurement Functions	
AB	Measures and displays A/B on the active channel.
ANAI	Measures and displays the data at the auxiliary input (ANALOG IN).
AR	Measures and displays A/R on the active channel.
BR	Measures and displays B/R on the active channel.
CONV	Converts S-parameter data.
MEAS	Measures and displays the selected input on the active channel.
S	Selects S-parameter displayed on the active channel.
TSTPP	Selects test port 1 or 2 for non-S-parameter measurements.
MENU (stimulus)	
ASEG	Uses all segments for list frequency sweep.
ATTP	Selects the amount of attenuation of port 1 or port 2 (option 011 with test set only).
CENT	Sets the center stimulus value.
CLEL	Clears the desired list.
CONT	Sets continuous sweep trigger mode.
COUC	Couples and uncouples the stimulus between the channels.
COUP	Couples the power when coupled channels is turned off.
CWFREQ	Sets the CW frequency for power sweep and CW frequency modes. While the list frequency table segment is being edited, it sets the center frequency of the current segment.
CWTIME	Selects the CW time sweep type.
EDITDONE	Done editing list frequency or limit table.
EDITLIST	Begins editing list frequency table.
EXTT	Sets the external trigger mode.
HOLD	Puts the sweep trigger into hold.
LIN	Selects a linear frequency sweep.

Table 5-3. Programming Commands (9 of 18)

Command	Description
LISFREQ	Selects the list frequency sweep mode.
LOG	Selects a log frequency sweep.
MANTRIG	Sets the external trigger to manual trigger on point.
NUMG	Activates the number of groups of sweeps. A group is whatever is needed to update the current parameter once.
POIN	Sets the number of points in the sweep. If a list frequency segment is being edited, sets the number of points in the segment.
PORTP	Selects either coupled or uncoupled for the port powers for a given channel.
POWE	Sets the output power level.
POWS	Selects power sweep, from the sweep type menu.
POWT	Turns power trip off, which clears a power trip after an overload condition is detected at one of the input ports.
PRAN	Selects the power range when in manual power range. Used with PWRR and PMAN.
PWRR	Selects the source power range auto or manual mode.
REST	Measurement restart.
SADD	Adds a new segment to the table.
SDEL	Deletes the current segment.
SDON	Closes a segment after editing.
SEDI	Selects the segment for editing.
SING	Sets to single sweep.
SLOPE	Enters the power slope value.
SLOPO	Turns the power slope on and off.
SOUP	Turns the source power on or off.
SPAN	Sets the stimulus span. If a list frequency segment is being edited, sets the span of the segment.
SSEG	Selects the desired segment of the frequency list for a list frequency sweep.
STAR	Enters the start stimulus value. If a list frequency segment is being edited, sets the start of the segment.

Table 5-3. Programming Commands (10 of 18)

Command	Description
STOP	Sets the stimulus stop value. If a list frequency segment is being edited, sets the stop of the segment.
STPSIZE	Sets the step size.
SWET	Sets the sweep time. Sets to 0 for auto sweep time.
Markers	
DELO	Turns the delta marker mode off.
DELR	Makes the indicated marker the delta reference.
DELRFIXM	Fixed marker.
DISM	Displays the response and stimulus values for all markers that are turned on.
MARK	Makes the indicated marker active and sets its stimulus.
MARKCONT	Places the markers continuously on the trace, not on discrete sample points.
MARKCOUP	Couples the markers between the channels.
MARKDISC	Places the markers in discrete placement mode.
MARKFAUV	Sets the auxiliary value of the fixed marker position. Works in coordination with MARKFVAL and MARKFSTI.
MARKFSTI	Sets the stimulus position of the fixed marker.
MARKFVAL	Sets the value of the fixed marker position.
MARKOFF	Turns all markers and marker functions off.
MARKUNCO	Uncouples the markers between channels.
MARKZERO	Places the fixed marker at the active marker position and makes it the delta reference.
POLM	Selects the marker readout format for polar display.
SMIM	Selects the marker readout format on a Smith chart.
Marker Functions	
MARKCENT	Enters the marker stimulus as the center stimulus.
MARKDELA	Sets electrical length so group delay is zero at the marker stimulus.
MARKREF	Enters the marker amplitude as the reference value.
MARKSPAN	Enters the span between the active marker and the delta reference as the sweep span.
MARKSTAR	Enters the marker stimulus as the start stimulus.

Table 5-3. Programming Commands (11 of 18)

Command	Description
MARKSTOP	Enters the marker stimulus as the stop stimulus.
MEASTAT	Turns trace statistics on and off.
SEA	Places the active marker.
TRACK	Turns marker search tracking on and off.
WIDT	Turns the bandwidth search on and off.
WIDV	Enters the widths search parameter.
Output Functions	
OUTMEMO	Outputs the memory trace from the active channel.
OUTP	Outputs to the HP-IB ports.
OUTPACTI	Outputs the value of the active function, or the last active function if the active area is OFF.
OUTPCALC	Outputs the error coefficient array for the active calibration on the active channel. Each array comes out in the current output format.
OUTPCALK	Outputs the active calibration kit as a less than 1000 byte string in form 1.
OUTPDATA	Outputs the error corrected data from the active channel in the current format.
OUTPERRO	Outputs the oldest error message in the error queue.
OUTPFORM	Outputs the formatted display data array from the active channel in the current format.
OUTPICALC	Outputs, over HP-IB, interpolated calibration coefficient array for the active calibration on the active channel.
OUTPIDEN	Outputs the identification string for the analyzer.
OUTPIPMCAL	Outputs, over HP-IB, the interpolated power meter calibration array for channel 1 or channel 2.
OUTPKEY	Outputs the key code of the last key pressed.
OUTPLEAS	Outputs the learn string, which contains the entire front panel state, the limit table, and the list frequency table, in form 1.
OUTPLIM	Outputs the limit test results.
OUTPMARK	Outputs the marker values.
OUTPMWID	Outputs the marker bandwidth search results: bandwidth, center, and Q, in that order.

Table 5-3. Programming Commands (12 of 18)

Command	Description
OUTPMWIL	Outputs the marker bandwidth search results: bandwidth, center, loss, and Q, in that order.
OUTPMSTA	Outputs the marker statistics: mean, standard deviation, and peak-to-peak deviation, in that order.
OUTPPMCAL	Outputs the power meter calibration array for channel 1 or channel 2. Values are sent as 100 times the source power in dB.
OUTPPRNALL	Prints all list values or operating and marker parameters in text mode to HP-IB.
OUTPRAW	Outputs the raw measurement data.
OUTPRFFR	Outputs the external source RF frequency, when in external source instrument mode.
OUTPSEQ	Outputs a sequence listing over HP-IB.
OUTPSTAT	Outputs the status byte.
OUTPTITL	Outputs the display title.
READ	Outputs the date/time in string format.
Output Formats	
FORM	Sets the data format for array transfers in and out of the instrument.
Save and Recall Registers	
CLEA	Clears the indicated save/recall registers.
CLEARALL	Clears all the save/recall registers.
CLEAREG	Clears save/recall registers 01 through 31.
RECA	Recalls the indicated internal register.
RECAREG	Recalls save/recall registers 01 through 31.
SAVE	Stores the current instrument state in the indicated internal register.
SAVEREG	Saves to save/recall registers 01 through 31.
TITF0	Titles the SAVE STATE filename in sequence mode.
TITP	Titles the plot file.
TITR	Titles the indicated internal register.
TITREG	Titles save/recall registers 01 through 31.
Menu Control	

Table 5-3. Programming Commands (13 of 18)

Command	Description
MENU	Activates the selected menu.
	Miscellaneous
COPYFRFT	Copies file titles defined by TITF into register titles.
COPYFRRT	Copies register titles defined by TITR into file titles.
DEFLTCPIO	Sets up a default state for copy.
EXTT	Sets the external trigger mode.
IDN?	Outputs the identification string.
KEY	Sends a key code, equivalent to actually pressing the key.
KOR?	Outputs a two byte key code/knob count.
MARKBUCK	Places the marker on a specific sweep point (bucket).
OPC	Operation complete. Reports the completion of the next command received by setting bit 0 in the event-status register, or by replying to an interrogation if OPC? is issued.
PRES	Presets the analyzer to the factory preset state.
PSOFT	Controls whether softkeys are included in the hardcopy print or plot.
SAMC	Turns sampler correction on and off. Sampler correction is only turned off to take data for customer calibration coefficients.
SELL	Selects the learn string revision (LRN) to input to or output from the analyzer.
SOFR	Displays the firmware revision on the screen.
TESS?	Returns a one on the standard analyzer.
WAIT	Waits for a clean sweep.
	Save and Recall Disk Files
DIRS	Sets the number of files in the directory at disk initialization.
EXTD	Selects the external disk as the active storage device.
EXTM	Stores data on disk.
FORMAT	Selects the disk format.
INI	Initializes the internal or external disk. All information on the disk will be destroyed.
INTD	Selects the internal disk as the active storage device.
INTM	Selects the internal memory as the active storage device.

Table 5-3. Programming Commands (14 of 18)

Command	Description
LOAD	Loads the file from disk with the name indicated by the previous TITF command. The actual file recalled depends on the file title in the file position specified.
PURG	Purges the indicated file from disk. Requires pass control mode when using the HP-IB port.
REFT	Recalls file titles from disk. Requires pass control mode when using the HP-IB port.
SAVU	Selects format for saving to disk.
STOR	Stores the indicated file on disk. Used with the INTD and EXTD commands to designate the internal or external disk.
TITF	Titles the indicated file numbers.
TITF0	Titles the SAVE STATE filename in sequence mode.
TITP	Titles the plot file.
Scale Ref	
AUTO	Auto scales the active channel.
ELED	Sets the electrical delay offset.
MARKDELA	Sets electrical length so group delay is zero at the marker stimulus.
MARKREF	Enters the marker amplitude as the reference value.
PHAO	Sets the phase offset.
REFP	Enters the reference position. 0 is the bottom, 10 is the top of the graticule.
REFV	Enters the reference line value.
SCAL	Sets the trace scale factor.
Seq-sequencing	
ADDR	Sets the HP-IB address for the specified device.
ASSS	Asserts the sequence status bit.
CLEABIT	Clears the specified bit on the GPIO.
CLEASEQ	Clears the sequence from the internal registers.
CONS	Continues the paused sequence.
DECRLOOC	Decrements the sequencing loop counter by 1. NEWSEQ must precede to ensure that a sequence is currently being created or modified.
DONM	Done modifying a test sequence.

Table 5-3. Programming Commands (15 of 18)

Command	Description
DOSEQ	Begins execution of the selected sequence.
DUPLSEQ<X>SEQ<Y>	Duplicates sequence X to sequence Y.
EMIB	Sends out a beep during a sequence. NEWSEQ must precede to ensure that a sequence is currently being created or modified.
GOSUB	Invokes a sequence as a subroutine.
IFBI	Tests the specified input GPIO bit (PARAIN).
IFLC	Branches an executing sequence to a new sequence if condition is satisfied. NEWSEQ must precede to ensure that a sequence is currently being created or modified.
IFLT	Branches an executing sequence to a new sequence if condition is satisfied. NEWSEQ must precede to ensure that a sequence is currently being created or modified.
INCRLOOC	Increments the sequencing loop counter by 1. NEWSEQ must precede to ensure that a sequence is currently being created or modified.
LOADSEQ	Loads the indicated sequence from disk. Requires pass control mode when using the HP-IB port.
LOOC	Sets the value of the sequencing loop counter. NEWSEQ must precede to ensure that a sequence is currently being created or modified.
MARKCW	Sets the CW frequency to the marker frequency.
NEWSEQ	Begins modifying a sequence.
PARAIN	Specifies the input GPIO bit to be used by the IFBI test.
PARAOUT	Programs all GPIO output bits (0 to 255) at once.
PAUS	Inserts a pause into a sequence. NEWSEQ must precede to ensure that a sequence is currently being created or modified.
PMTRTTIT	Reads power meter/HP-IB value into title string. NEWSEQ must precede to ensure that a sequence is currently being created or modified.
PRINSEQ	Begins printing the sequence selected. Requires pass control mode when using the HP-IB port.
PTOS	Pauses the sequence to be followed by selection one of the six sequences.
SEQ	Selects sequence 1 through 6.
SEQWAIT	Tells the instrument to wait a specified number of seconds during a sequence. NEWSEQ must precede to ensure that a sequence is currently being created or modified.

Table 5-3. Programming Commands (16 of 18)

Command	Description
SETBIT	Sets the specified bit (0 to 7) on the GPIO.
SHOM	Displays the desired softkey menu during a sequence. NEWSEQ must precede to ensure that a sequence is currently being created or modified.
STORSEQ	Stores the instrument state to the indicated sequence to disk. Used with the INTD and EXTD commands to designate the internal or external disk. Requires pass control mode when using the HP-IB port.
TITSEQ	Selects the sequence to be titled.
TITT	Sends the title string. NEWSEQ must precede to ensure that a sequence is currently being created or modified.
TSTIO	Defines 3 bits, D0 through D2, on the test set connector I/O for the channel 1 and channel 2 forward or reverse settings.
TTL	Sets the rear panel “TEST SEQ” BNC output at end of sweep.
Softkeys	
SOFT	Presses the indicated softkey.
WRSK	Enters new softkey labels into the indicated softkey positions.
Status Reporting	
CLES	Clears the status byte.
ESB?	Outputs event-status register B.
ESE	Enables the selected event-status register bits to be summarized by bit 5 of the status byte.
ESNB	Enables the selected event-status register B bits to be summarized by bit 2 of the status byte.
ESR?	Outputs the value of the event-status register.
OUTPSTAT	Outputs the status byte.
SRE	Service request enable.
Stimulus	
CENT	Sets the center stimulus value. If a list frequency segment is being edited, sets the center of the list segment.
SPAN	Sets the stimulus span. If a list frequency segment is being edited, sets the span of the segment.

Table 5-3. Programming Commands (17 of 18)

Command	Description
STAR	Enters the start stimulus value. If a list frequency segment is being edited, sets the start of the segment.
STOP	Sets the stimulus stop value. If a list frequency segment is being edited, sets the stop of the segment.
System	
ANAB	Enables the analog bus for service use.
DCONV	Selects down converter for mixer measurements.
FREQOFFS	Activates the frequency offset instrument mode.
HARM	Sets the harmonic measurement mode (option 002).
INSM	Selects the instrument mode.
RF	Sets RF greater or less than LO for mixer measurement.
SETDATE	Sets the date in the following format: DD MMM YYYY, where DD is the day and must be 2 digits, MMM is the month and must be three alpha characters, and YYYY is the year and must be 4 digits.
SETTIME	Sets the time in the following format: HH:MM:SS, where HH is the hour, MM is minutes, SS is seconds, and each must be 2 digits.
TIMESTAM	Turns on the clock time for prints and plots.
UCONV	Selects up converter for mixer measurements.
VIEM	Displays the measurement trace (ON) or the mixer setup (OFF).
VOFF	Sets the frequency offset value.
System-limit testing	
BEEP	Controls the warning beeper for a limit test failure.
CLEAL	Clears the desired list.
EDITDONE	Done editing list frequency or limit line table.
EDITLIML	Begins editing limit table.
LIM	Sets the limits for limit lines.
LIMILINE	Turns the display of the limit lines on and off.
LIMIMAOF	Marker to limit offset. Centers the limit lines about the current marker position using the limit amplitude offset function.

Table 5-3. Programming Commands (18 of 18)

Command	Description
LIMISTIO	Enters the stimulus offset of the limit lines.
LIMITEST	Turns limit testing on and off.
LIMT	Sets the limit segment value.
MARKMIDD	Makes the marker amplitude the limit segment middle value, during a limit segment edit.
MARKSTIM	During a limit segment edit, enters the marker stimulus as the limit stimulus break point.
SADD	Adds a new segment to the table, during either a list frequency or limit line table edit.
SDEL	Deletes the current segment, during either a list frequency or limit line table edit.
SDON	Closes a segment after editing, during either a list frequency or limit line table edit.
SEDI	Selects the segment for editing, during either a list frequency or limit line table edit.
System-transform	
BANDPASS	Selects the time domain bandpass mode.
DEMO	Controls transform demodulation.
GATE	Sets the time domain gate time.
GATEO	Turns the time domain gate on or off.
GATS	Sets the gate shape.
LOW	Turns ON the low pass transform (option 010).
SETF	Sets frequency for low pass transform (option 010).
SPEG	Displays the specify gate menu.
TIMDTRAN	Turns the option 010 (time domain) transform on and off.
WIND	Sets the window for a transform.
WINDOW	Enters arbitrary window.
WINDUSEM	Turns the trace memory on as the window shape.

Key Definitions

The following table lists the softkey functions alphabetically, and the equivalent programming command.

Table 5-4. Keys versus Programming Commands (1 of 32)

Softkey	Equivalent Programming Command
↓	`DOWN`
Δ *MODE MENU*	
Δ *MODE OFF*	`DELO`
Δ *REF= 1*	`DELR1`
Δ *REF= 2*	`DELR2`
Δ *REF= 3*	`DELR3`
Δ *REF= 4*	`DELR4`
Δ *REF=* Δ *FIXED MKR*	`DELRFIXM`
1/S	`CONVIDS`
A	`MEASA`
A/B	`AB`
A/R	`AR`
ACTIVE ENTRY	
ACTIVE MKR MAGNITUDE	—
ADD	`SADD`
ADDRESS: 8702	

Table 5-4. Keys versus Programming Commands (2 of 32)

Softkey	Equivalent Programming Command
ADDRESS: CONTROLLER	ADDRCONT
ADDRESS: DISK	ADDRDISC
ADDRESS: P MTR/HPIB	ADDRPOWM
ADJUST DISPLAY	
ADJUSTMENT TESTS	
ALC ON off	
ALL SEGS SWEEP	ASEG
ALTERNATE A AND B	ALTAB
AMPLITUDE OFFSET	LIMIAMPO
ANALOG BUS on OFF	ANAB
ANALOG IN Aux Input	ANAI
ARBITRARY IMPEDANCE	STDTARBI
ASCII	SAVUASCI
ASSERT SRQ	ASSS
AUTO FEED ON off (Plotter)	PLTTRAUTFON or PLTTRAUTFOFF
AUTO FEED ON off (Printer)	PRNTRAUTFON or PRNTRAUTOFF
AUTO SCALE	AUTO
AVERAGING FACTOR	AVERFACT
AVERAGING on OFF	AVERON or AVEROFF
AVERAGING RESTART	AVERREST
AVG	MENUAVG
B	MEASB
B/R	BR

Table 5-4. Keys versus Programming Commands (3 of 32)

Softkey	Equivalent Programming Command
BACKGROUND INTENSITY	`BACI`
BACK SPACE	
BANDPASS	`BANDPASS`
BEEP DONE ON off	`BEEPDONEON` or `BEEPDONEOFF`
BEEP FAIL on OFF	`BEEPFAILON` or `BEEPFAILOFF`
BEEP WARN on OFF	`BEEPWARNON` or `BEEPWARNOFF`
BRIGHTNESS	`CBRI`
C0	`C0`
C1	`C1`
C2	`C2`
C3	`C3`
CAL	`MENUCAL`
CAL FACTOR	`CALFCALF`
CAL FACTOR SENSOR A	`CALFSENA`
CAL FACTOR SENSOR B	`CALFSENB`
CAL KIT: 3.5mmC	`CALK35MM`
CAL KIT: 3.5mmD	`CALK35MD`
CAL KIT: 7mm	`CALK7MM`
CAL KIT: N 50W	`CALKN50`
CAL KIT: N 75W	`CALKN75`
CAL KIT: STD KIT	
CAL KIT: USER KIT	`CALKUSED`
CAL KITS & STDS	

Table 5-4. Keys versus Programming Commands (4 of 32)

Softkey	Equivalent Programming Command
CAL STD: SRC COEFF	
CAL STD: SRC DISK	
CALIBRATE MENU	
CALIBRATE: NONE	`CALN`
CANCEL	
CENTER	`CENT`
CH 1	`CHAN1`
CH1 DATA [ ]	`PCOLDATA1`
CH1 DATA LIMIT LN	`COLOCH1D`
CH1 MEM	`COLOCH1M`
CH1 MEM [ ]	`PCOLMEMO1`
CH 2	`CHAN2`
CH2 DATA [ ]	`PCOLDATA2`
CH2 DATA LIMIT LN	`COLOCH2D`
CH2 MEM [ ]	`PCOLMEMO2`
CH2 MEM REF LINE	`COLOCH2M`
CHAN PWR [COUPLED]	`CHANPCPLD`
CHAN PWR [UNCOUPLED]	`CHANPUNCPLD`
CHOP A AND B	`CHOPAB`
CLASS DONE	`CLAD`
CLEAR BIT	`CLEABIT`
CLEAR LIST	`CLEAL`
CLEAR SEQUENCE	`CLEASEQn`

Table 5-4. Keys versus Programming Commands (5 of 32)

Softkey	Equivalent Programming Command
COAX	`COAX`
COEFF A	
COEFF B	
COEFF C	
COEFF D	
COEFF E	
COEFF F	
COEFF G	
COEFF H	
COEFF I	
COEFF STD⟶MEMORY	
COLOR	`COLOR`
CONFIGURE EXT DISK	
CONTINUE SEQUENCE	`CONS`
CONTINUE TEST	
CONTINUOUS	`CONT`
CONVERSION [ ]	`CONVOFF`
COPY	`MENUCOPY`
CORRECTION on OFF	`CORRON` or `CORROFF`
COUPLED CH ON off	`COUCON` or `COUCOFF`
CW FREQ	`CWFREQ`
CW TIME	`CWTIME`
D2/D1 to D2 on OFF	`D1DIVD2ON` or `D1DIVD2OFF`

Table 5-4. Keys versus Programming Commands (6 of 32)

Softkey	Equivalent Programming Command
DATA AND MEMORY	DISPDATM
DATA ARRAY on OFF	EXTMDATAON or EXTMDATAOFF
DATA/MEM	DISPDDM
DATA – MEM	DISPDMM
DATA → MEMORY	DATI
DATA ONLY ON off	EXTMDATOON or EXTMDATOOFF
DECISION MAKING	
DECR LOOP COUNTER	DECRLOOC
DEFAULT COLORS	DEFC
DEFAULT PLOT SETUP	DFLT
DEFAULT PRNT SETUP	DEFLPRINT
DEFINE DISK-SAVE	
DEFINE PLOT	
DEFINE PRINT	
DEFINE:REFLECTOR	
DEFINE STANDARD	DEFS
DELAY	DELA
DELAY/LOAD	
DELAY/THRU	
DELETE	SDEL
DELETE ALL FILES	
DELETE FILE	
DELTA LIMITS	LIMD

Table 5-4. Keys versus Programming Commands (7 of 32)

Softkey	Equivalent Programming Command
DEMOD: AMPLITUDE	`DEMOAMPL`
DEMOD: OFF	`DEMOOFF`
DEMOD: PHASE	`DEMOPHAS`
DETECTOR OFFSET DAC	
DIR SIZE:DEFAULT	
DIRECTORY SIZE (LIF)	`DIRS`
DISK STD → MEMORY	
DISK UNIT NUMBER	`DISCUNIT`
DISP MKRS ON off	`DISM`
DISPLAY	`MENUDISP`
DISPLAY: DATA	`DISPDATA`
DISPLAY TESTS	
DO SEQUENCE	`DOSEQn`
DONE	`EDITDONE`
DONE (Segment)	`SDON`
DONE 1-PORT CAL	`SAV1`
DONE 2-PORT CAL	`SAV2`
DONE RESP ISOL'N CAL	`RAID`
DONE RESPONSE	`RESPDONE`
DONE SEQ MODIFY	`DONM`
DOWN CONVERTER	`DCONV`
DUAL CHAN on OFF	`DUACON` or `DUACOFF`
DUPLICATE SEQUENCE	`DUPLSEQxSEQy`

Table 5-4. Keys versus Programming Commands (8 of 32)

Softkey	Equivalent Programming Command
EACH SWEEP	`PWMCEACS`
EDIT	`SEDI`
EDIT LIMIT LINE	`EDITLIML`
EDIT LIST	`EDITLIST`
ELECTRICAL [ ]	
ELECTRICAL DELAY	`ELED`
ELECTRICAL PARAMETERS	
EMIT BEEP	`EMIB`
END OF LABEL	
END SWEEP HIGH PULSE	`TTLHPULS`
END SWEEP LOW PULSE	`TTLLPULS`
ENTER SRC COEFF	
ENTRY OFF	`ENTO`
ERASE TITLE	
EXECUTE TEST	
EXT SOURCE AUTO	`INSMEXSA`
EXT SOURCE MANUAL	`INSMEXSM`
EXT TRIG ON POINT	`EXTTPOIN`
EXT TRIG ON SWEEP	`EXTTON`
EXTENSION INPUT A	`PORTA`
EXTENSION INPUT B	`PORTB`
EXTENSION INPUT R	
EXTENSION PORT 1	`PORT1`

Table 5-4. Keys versus Programming Commands (9 of 32)

Softkey	Equivalent Programming Command
EXTENSION PORT 2	`PORT2`
EXTENSIONS on OFF	`POREON` or `POREOFF`
EXTERNAL DISK	`INTM`
EXTERNAL TESTS	
FILENAME FILE0	`TITF0`
FILE UTILITIES	
FIRMWARE REVISION	
FIXED	`FIXE`
FIXED MKR AUX VALUE	`MARKFAUV`
FIXED MKR POSITION	`DELRFIXM`
FIXED MKR STIMULUS	`MARKFSTI`
FIXED MKR VALUE	`MARKFVAL`
FLAT LINE	`LIMTFL`
FORM FEED	
FORMAT	`MENUFORM`
FORMAT ARY on OFF	`EXTMFORMON` or `EXTMFORMOFF`
FORMAT: DOS	`FORMATDOS`
FORMAT: LIF	`FORMATLIF`
FORMAT DISK	
FORMAT EXT DISK	`INIE`
FORMAT INT DISK	`INID`
FORMAT INT MEMORY	`INTM`
FORWARD:OPEN	

Table 5-4. Keys versus Programming Commands (10 of 32)

Softkey	Equivalent Programming Command
FRACN TUNE on OFF	
FREQ OFFS MENU	
FREQ OFFS ON off	FREQOFFSON or FREQOFFSOFF
FREQUENCY	CALFFREQ
FREQUENCY BLANK	FREO
FREQUENCY: CW	LOFREQ
FRESNEL	
FULL 2-PORT	CALIFUL2
FULL PAGE	FULP
FWD ISOL'N ISOL'N STD	FWDI
FWD MATCH (Label Class)	LABEFWDM and LABETTFM
FWD MATCH (Specify Class)	SPECFWDM and SPECTTFM
FWD MATCH THRU	FWDM
FWD TRANS (Label Class)	LABEFWDT and LABETTFT
FWD TRANS (Specify Class)	SPECFWDT and SPECTTFT
FWD TRANS THRU	FWDT
G+jB MKR	SMIMGB
GATE: CENTER	GATECENT
GATE: SPAN	GATESPAN
GATE: START	GATESTAR
GATE: STOP	GATESTOP
GATE on OFF	GATEOON or GATEOOFF
GATE SHAPE MAXIMUM	GATSMAXI

Table 5-4. Keys versus Programming Commands (11 of 32)

Softkey	Equivalent Programming Command
GATE SHAPE MINIMUM	`GATSMINI`
GATE SHAPE NORMAL	`GATSNORM`
GATE SHAPE WIDE	`GATSWIDE`
GET SEQ TITLES	
GOSUB SEQUENCE	`GOSUBSEQ`
GRAPHICS on OFF	`EXTMGRAPON` or `EXTMGRAPOFF`
GRATICULE [ ]	`PCOLGRAT`
GRATICULE:TEXT	`COLOGRAT`
GRATICULE:WARNING	
GUIDED CAL	
HARMONIC MEAS	
HARMONIC OFF	`HARMOFF`
HARMONIC SECOND	`HARMSEC`
HARMONIC THIRD	`HARMTHIR`
HB FLTR SW on OFF	
HOLD	`HOLD`
HP-IB DIAG on OFF	`DEBUON` or `DEBUOFF`
IF BW [ ]	`IFBW`
IF GAIN:AUTO	
IF GAIN:OFF	
IF GAIN:ON	
IF LIMIT TEST FAIL	`IFLTFAIL`
IF LIMIT TEST PASS	`IFLTPASS`

Table 5-4. Keys versus Programming Commands (12 of 32)

Softkey	Equivalent Programming Command
IF LOOP COUNTER = 0	`IFLCEQZE`
IF LOOP COUNTER < > 0	`IFLCNEZE`
IMAGINARY	`IMAG`
INCR LOOP COUNTER	`INCRLOOC`
INDEX of REFRACTION	
INPUT PORTS	
INSTRUMENT MODE	
INTENSITY	`INTE`
INTERNAL DISK	`INTD`
INTERNAL MEMORY	`INTM`
INTERNAL TESTS	
INTERPOL on OFF	`CORION` or `CORIOFF`
ISOLATION (2-Port)	`ISOL`
ISOLATION (One-Path 2-Port)	`ISOOP`
ISOLATION DONE	`ISOD`
ISOL'N STD	`RAIISOL`
ISTATE CONTENTS	
KIT DONE (MODIFIED)	`KITD`
LABEL	
LABEL KIT	`LABK`
LABEL STD	`LABS`
LEFT LOWER	`LEFL`
LEFT UPPER	`LEFU`

Table 5-4. Keys versus Programming Commands (13 of 32)

Softkey	Equivalent Programming Command
LIGHTWAVE PARAMETERS	
LIMIT LINE OFFSETS	
LIMIT LINE on OFF	`LIMILINEON` or `LIMILINEOFF`
LIMIT MENU	
LIMIT TEST on OFF	`LIMITESTON` or `LIMITESTOFF`
LIMIT TEST RESULT	
LIMIT TYPE	
LIMITS [NORM]	
LIN FREQ	`LINFREQ`
LIN MAG	`LINM`
LIN MKR	`POLMLIN`
LINE TYPE DATA	`LINTDATA`
LINE TYPE MEMORY	`LINTMEMO`
LIST	
LIST FREQ	`LISTFREQ`
LIST VALUES	`LISV`
LO CONTROL on OFF	`LOCONTON` or `LOCONTOFF`
LO SOURCE ADDRESS	`ADDRLSRC`
LOAD	`STDTLOAD`
LOAD SEQ FROM DISK	`LOADSEQn`
LOAD SEQ SEQ1	
LOAD SEQ SEQ2	
LOAD SEQ SEQ3	

Table 5-4. Keys versus Programming Commands (14 of 32)

Softkey	Equivalent Programming Command
LOAD SEQ SEQ4	
LOAD SEQ SEQ5	
LOAD SEQ SEQ6	
LOAD SRC DISK	
LOCAL	
LOG FREQ	`LOGFREQ`
LOG MAG	`LOGM`
LOG MKR	`SMIMLOG`
LOG OFFSET DAC	
LOOP COUNTER	`LOOC`
LOSS	`POWLLOSS`
LOSS/SENSR LISTS	
LOW PASS IMPULSE	`LOWPIMPU`
LOW PASS STEP	`LOWPSTEP`
LOWER LIMIT	`LIML`
MAIN POWER DAC	
MANUAL TRG ON POINT	`MANTRIG`
MARKER	`MENUMARK`
MARKER → AMP. OFS.	`LIMIMAOF`
MARKER → CENTER	`MARKCENT`
MARKER → CW	`MARKCW`
MARKER → DELAY	`MARKDELA`
MARKER → MIDDLE	`MARKMIDD`

Table 5-4. Keys versus Programming Commands (15 of 32)

Softkey	Equivalent Programming Command
MARKER → REFERENCE	`MARKREF`
MARKER → SPAN	`MARKSPAN`
MARKER → START	`MARKSTAR`
MARKER → STIMULUS	`MARKSTIM`
MARKER → STOP	`MARKSTOP`
MARKER 1	`MARK1`
MARKER 2	`MARK2`
MARKER 3	`MARK3`
MARKER 4	`MARK4`
MARKER all OFF	`MARKOFF`
MARKER FCTN	`MENUMRKF`
MARKER MODE MENU	
MARKERS: CONTINUOUS	`MARKCONT`
MARKERS: COUPLED	`MARKCOUP`
MARKERS: DISCRETE	`MARKDISC`
MARKERS: UNCOUPLED	`MARKUNCO`
MAXIMUM FREQUENCY	`MAXF`
MEAS	`MENUMEAS`
MEASURE RESTART	`REST`
MEMORY	`DISPMEMO`
MENU	
MIDDLE VALUE	`LIMM`
MINIMUM FREQUENCY	`MINF`

Table 5-4. Keys versus Programming Commands (16 of 32)

Softkey	Equivalent Programming Command
MKR SEARCH [OFF]	SEAOFF
MKR ZERO	MARKZERO
MODIFY [ ]	MODI1
MODIFY COLORS	
MODIFY THRU/RCVR	
MORE	
MORE (F—I)	
MORE RANGES	
NETWORK ANALYZER	INSMNETA
NEWLINE	
NEW SEQ/MODIFY SEQ	NEWSEQn
NUMBER OF GROUPS	NUMG
NUMBER OF POINTS	POIN
NUMBER OF READINGS	NUMR
OFFSET DELAY	OFSD
OFFSET LENGTH	
OFFSET LOSS	OFSL
OFFSETn	
OFFSET Z0	OFSZ
OMIT ISOLATION	OMII
ONE-PATH 2-PORT	CALIONE2
ONE SWEEP	PWMCONES
OP PARMS (MKRS etc)	OPEP

Table 5-4. Keys versus Programming Commands (17 of 32)

Softkey	Equivalent Programming Command
OPEN	
OPTICAL [ ]	
P MTR/HPIB TO TITLE	PMTRTTIT
PAGE	NEXP
PARALL IN BIT NUMBER	PARAIN
PARALL IN IF BIT H	IFBIHIGH
PARALL IN IF BIT L	IFBILOW
PARALLEL [ ]	PARALCPY or PARALGPIO
PARALLEL OUT ALL	PARAOUT
PAUSE	PAUS
PAUSE TO SELECT	PTOS
PEEK	
PEEK/POKE	
PEEK/POKE ADDRESS	
PEN NUM DATA	PENNDATA
PEN NUM GRATICULE	PENNGRAT
PEN NUM MARKER	PENNMARK
PEN NUM MEMORY	PENNMEMO
PEN NUM TEXT	PENNTEXT
PERIPHERAL HPIB ADDR	
PHASE	PHAS
PHASE OFFSET	PHAO
PLL AUTO ON off	

Table 5-4. Keys versus Programming Commands (18 of 32)

Softkey	Equivalent Programming Command
PLL DIAG on OFF	
PLL PAUSE [CONT]	
PLOT	PLOT
PLOT DATA ON off	PDATAON or PDATAOFF
PLOT GRAT ON off	PGRATON or PGRATOFF
PLOT MEM ON off	PMEMON or PMEMOFF
PLOT MKR ON off	PMKRON or PMKROFF
PLOT SPEED [ ]	PLOSFAST or PLOSSLOW
PLOT TEXT ON off	PTEXTON or PTEXTOFF
PLOTTER BAUD RATE	PLTTRBAUD
PLOTTER FORM FEED	PLTTRFORF
PLOTTER PORT	
PLTR PORT: DISK	PLTPRTDISK
PLTR PORT: HPIB	PLTPRTHPIB
PLTR PORT: PARALLEL	PLTPRTPARA
PLTR PORT: SERIAL	PLTPRTSERI
PLTR TYPE [ ]	PLTTYPPLTR or PLTTYPHPGL
POKE	
POLAR	POLA
POLAR MKR MENU	
PORT EXTENSIONS	
PORT PWR [COUPLED]	PORTPCPLD
PORT PWR [UNCOUPLED]	PORTPUNCPLD

Table 5-4. Keys versus Programming Commands (19 of 32)

Softkey	Equivalent Programming Command
POWER	`POWE`
POWER CAL [OFF]	
POWER: FIXED	`LOPOWER`
POWER LOSS	
POWER MTR	`POWM`
POWER MTR: [ ]	`POWMON` or `POWMOFF`
POWER RANGE AUTO man	`PWRRAUTO`
POWER RANGE auto MAN	`PWRRMAN`
POWER RANGES	`PWRR`
POWER SWEEP	`POWS`
POWER: SWEEP	`LOPSWE`
PRESET	`RST and PRES`
PRESET: FACTORY	`RST and PRES`
PRESET: USER	
PREVIOUS RANGES	
PRINT ALL COLOR	`PRINTALL`
PRINT ALL MONOCHROME	`PRINTALL`
PRINT: COLOR	`PRIC`
PRINT COLOR	
PRINT COLORS	`PRINALL`
PRINT: MONOCHROME	`PRIS`
PRINT MONOCHROME	`PRINALL`
PRINT SEQUENCE	`PRINSEQn`

Table 5-4. Keys versus Programming Commands (20 of 32)

Softkey	Equivalent Programming Command
PRINTER BAUD RATE	PRNTRBAUD
PRINTER FORM FEED	PRNTRFORF
PRINTER PORT	
PRNTR PORT HPIB	PRNPRTHPIB
PRNTR PORT PARALLEL	PRNPRTPARA
PRNTR PORT SERIAL	PRNPRTSERI
PRNTR TYPE [ ]	PRNTYPDJ/EP/LJ/PJ/TJ
PURGE SEQUENCES	
PURGE SEQ SEQ1	
PURGE SEQ SEQ2	
PWR LOSS on OFF	PWRLOSSON or PWRLOSSOFF
PWR RANGE AUTO man	PWRRPAUTO or PWRRPMAN
PWRMTR CAL [ ]	PWRMCAL
PWRMTR CAL [OFF]	PWRMMCALOFF
R	MEASR
R+jX MKR	SMIMRX
RANGE 0 –15 TO +10	PRAN0
RANGE 1 –25 TO 0	PRAN1
RANGE 2 –35 TO –10	PRAN2
RANGE 3 –45 TO –20	PRAN3
RANGE 4 –55 TO –30	PRAN4
RANGE 5 –65 TO –40	PRAN5
RANGE 6 –75 TO –50	PRAN6

Table 5-4. Keys versus Programming Commands (21 of 32)

Softkey	Equivalent Programming Command
RANGE 7 −85 TO −60	PRAN7
RAW ARRAY on OFF	EXTMRAWON or EXTMRAWOFF
Re/Im MKR	POLMRI
READ FILE TITLES	REFT
READ SEQ FILE TITLS	
REAL	REAL
RECALL COLORS	RECO
RECALL KEYS MENU	
RECALL KEYS on OFF	
RECALL REG1	RECA1
RECALL REG2	RECA2
RECALL REG3	RECA3
RECALL REG4	RECA4
RECALL REG5	RECA5
RECALL REG6	RECA6
RECALL REG7	RECA7
RECALL STATE	
RECEIVER	
RECEIVER [ ]	
RECORD on OFF	
REFERENCE POSITION	REFP
REFERENCE VALUE	REFV
Refl: E S11 (A/R)	

Table 5-4. Keys versus Programming Commands (22 of 32)

Softkey	Equivalent Programming Command
Refl: E S11 FWD	
Refl: E S22 (B/R)	
Refl: E S22 REV	
Refl: O (A/R)	
Refl: O (B/R)	
Refl: O (PORT 1→2)	
Refl O (PORT 2→1)	
Refl: FWD S11 (A/R)	`S11`
Refl: REV S22 (B/R)	`S22`
REFLECTED POWER	
REFLECTION	`REFOP`
RENAME FILE	
REPEAT on OFF	
RE-SAVE STATE	
RESET COLOR	`RSCO`
RESET MEMORY	
RESPONSE (Calibrate)	`CALIRESP`
RESPONSE (Label Class)	`LABERESP`
RESPONSE (Specify Class)	`SPECRESP`
RESPONSE & ISOL'N (Calibrate)	`CALIRAI`
RESPONSE & ISOL'N (Label Class)	`LABERESI`
RESPONSE & ISOL'N (Specify Class)	`SPECRESI`
RESPONSE & MATCH	

Table 5-4. Keys versus Programming Commands (23 of 32)

Softkey	Equivalent Programming Command
RESTORE DISPLAY	`RESD`
RESUME CAL SEQUENCE	`RESC`
RETURN	
REV ISOL'N ISOL'N STD	`REVI`
REV MATCH (Label Class)	`LABEREVM` and `LABETTRM`
REV MATCH (Specify Class)	`SPECREVM` and `SPECTTRM`
REV MATCH THRU	`REVM`
REV TRANS (Label Class)	`LABEREVT` and `LABETTRT`
REV TRANS (Specify Class)	`SPECREVT` and `SPECTTRT`
REV TRANS THRU	`REVT`
REVERSE:OPEN	
RF > LO	`RFGTLO`
RF < LO	`RFLTLO`
RIGHT LOWER	`RIGL`
RIGHT UPPER	`RIGU`
ROUND SECONDS	
S11 1-PORT	`CALIS111`
S11A RE FW MTCH (Label Class)	`LABES11A` and `LABETRFM`
S11A RE FW MTCH (Specify Class)	`SPECS11A` and `SPECTRFM`
S11B LN FW MTCH (Label Class)	`LABES11B` and `LABETRRM`
S11B LN FW MTCH (Specify Class)	`SPECS11B` and `SPECTRRM`
S11C LN FW TRAN (Label Class)	`LABES11C` and `LABETLFT`
S11C LN FW TRAN (Specify Class)	`SPECS11C` and `SPECTLFT`

Table 5-4. Keys versus Programming Commands (24 of 32)

Softkey	Equivalent Programming Command
S22 1-PORT	`CALIS221`
S22A RE RV MTCH (Label Class)	`LABES22A` and `LABETRRM`
S22A RE RV MTCH (Specify Class)	`SPECS22A` and `SPECTRRM`
S22B LN RV MTCH (Label Class)	`LABES22B` and `LABETLRM`
S22B LN RV MTCH (Specify Class)	`SPECS22B` and `SPECTLRM`
S22C LN RV TRAN (Label Class)	`LABES22C` and `LABETLRT`
S22C LN RV TRAN (Specify Class)	`SPECS22C` and `SPECTLRT`
SAMPLER COR ON off	
SAVE COLORS	`SVCO`
SAVE/RECALL	
SAVE SRC COEFF	
SAVE STATE	
SAVE USER KIT	`SAVEUSEK`
SAVE USING ASCII	
SAVE USING BINARY	`SAVUBINA`
SCALE/DIV	`SCAL`
SCALE PLOT [ ]	`SCAPFULL` or `SCAPGRAT`
SCALE REF	`MENUSCAL`
SEARCH LEFT	`SEAL`
SEARCH RIGHT	`SEAR`
SEARCH: MAX	`SEAMAX`
SEARCH: MIN	`SEAMIN`
SEARCH: OFF	`SEAOFF`

Table 5-4. Keys versus Programming Commands (25 of 32)

Softkey	Equivalent Programming Command
SEARCH: TARGET	
SECOND	`HARMSEC`
SEGMENT	
SEGMENT: CENTER	`CENT`
SEGMENT: SPAN	`SPAN`
SEGMENT: START	`STAR`
SEGMENT: STOP	`STOP`
SEL QUAD [ ]	
SELECT DISK	
SELECT LETTER	
SELECT SEQ SEQ4	
SELF DIAGNOSE	
SEQ	
SEQUENCE 1 SEQ1	`SEQ1`
SEQUENCE 2 SEQ2	`SEQ2`
SEQUENCE 3 SEQ3	`SEQ3`
SEQUENCE 4 SEQ4	`SEQ4`
SEQUENCE 5 SEQ5	`SEQ5`
SEQUENCE 6 SEQ6	`SEQ6`
SERVICE MENU	
SERVICE MODES	
SET ADDRESSES	
SET BIT	`SETBIT`

Table 5-4. Keys versus Programming Commands (26 of 32)

Softkey	Equivalent Programming Command
SET CLOCK	
SET DATE	`SETDATE`
SET DAY	
SET FREQ LOW PASS	`SETF`
SET HOUR	
SET MINUTES	
SET MONTH	
SET TIME	`SETTIME`
SET YEAR	
SET Z0	`SETZ`
SHORT	
SHOW MENUS	`SHOM`
SINGLE	`SING`
SINGLE POINT	`LIMTSP`
SINGLE SEG SWEEP	`SSEG`
SLIDE is SET	
SLIDING	`SLIL`
SLIDING LOAD DONE	
SLOPE	`SLOPE`
SLOPE DAC	
SLOPE OFFSET DAC	
SLOPE on OFF	`SLOPON` or `SLOPOFF`
SLOPING LINE	`LIMTSL`

Table 5-4. Keys versus Programming Commands (27 of 32)

Softkey	Equivalent Programming Command
SMITH CHART	`SMIC`
SMITH MKR MENU	
SMOOTHING APERTURE	`SMOOAPER`
SMOOTHING ON off	`SMOOON` or `SMOOOFF`
SOURCE	
SOURCE [ ]	
SOURCE PLL ON off	
SOURCE PWR on OFF	`SOUPON` or `SOUPOFF`
SPACE	
SPAN	`SPAN`
SPECIAL FUNCTIONS	
SPECIFY	
SPECIFY GATE	`SPEG`
SPECIFY OFFSET	
SPLIT DISP ON off	`SPLDON` or `SPLDOFF`
SPUR AVOID ON off	
SPUR TEST on OFF	
SQUARE LAW LINEAR DAC	
SRC ADJUST DACS	
SRC ADJUST MENU	
SRC COEFF	
SRC TUNE FREQ	
SRC TUNE on OFF	

Table 5-4. Keys versus Programming Commands (28 of 32)

Softkey	Equivalent Programming Command
STANDARDS DONE	
START	LOFSTAR
START	STAR
STATS on OFF	MEASTATON or MEASTATOFF
STD DONE (DEFINED)	STDD
STD OFFSET DONE	
STD TYPE: ARBITRARY IMPEDANCE	STDTARBI
STD TYPE: DELAY/THRU	STDTDELA
STD TYPE: LOAD	STDTLOAD
STD TYPE: OPEN	STDTOPEN
STD TYPE: SHORT	STDTSHOR
STEP SIZE	STPSIZE
STIMULUS OFFSET	LIMISTIO
STIMULUS VALUE	LIMS
STOP	LOFSTOP
STOP	STOP
STORE EEPR on OFF	
STORE SEQ SEQ4	
STORE SEQ TO DISK	STORSEQn
SWEEP	LOFSWE
SWEEP TIME [AUTO]	SWET
SWEEP TYPE MENU	
SWR	SWR

Table 5-4. Keys versus Programming Commands (29 of 32)

Softkey	Equivalent Programming Command
SYSTEM	MENUSYST
SYSTEM CONTROLLER	Take Calibration Sweep
SYS VER TESTS	
TALKER/LISTENER	TALKLIST
TARGET	SEATARG
TERMINAL IMPEDANCE	TERI
TEST OPTIONS	
TESTPORT (1) 2	TSTPP1
TESTPORT 1 (2)	TSTPP2
TESTS	
TESTSET I/O FWD	TSTIOFWD
TESTSET I/O REV	TSTIOREV
TESTSET SW CONT hld	CSWION or CSWIOFF
TEXT	COLOTEXT
TEXT [ ]	PCOLTEXT
THRU	
THRU/RCVR	
THRU/RCVR:RECEIVER	
THRU/RCVR:THRU	
THRU/SRC	
TIME STAMP ON off	TIMESTAMON or TIMESTAMOFF
TINT	TINT
TITLE	TITL

Table 5-4. Keys versus Programming Commands (30 of 32)

Softkey	Equivalent Programming Command
TITLE FILE1	`TITF1`
TITLE FILE2	`TITF2`
TITLE FILE3	`TITF3`
TITLE FILE4	`TITF4`
TITLE FILE5	`TITF5`
TITLE SEQUENCE	`TITSEQn`
TITLE TO MEMORY	`TITTMEM`
TITLE TO P MTR/HPIB	`TITTPMTR`
TITLE TO PERIPHERAL	`TITTPERI`
TITLE TO PRINTER	`TITTPRIN`
TRACKING on OFF	`TRACKON` or `TRACKOFF`
TRANS DONE	`TRAD`
TRANS: E/E S12 (A/R)	
TRANS: E/E S12 REV	`S12`
TRANS: E/E S21 (B/R)	
TRANS: E/E S21 FWD	`S21`
TRANS: E/O (A/R)	
TRANS: E/O (B/R)	
TRANS: E/O (PORT 1→2)	
TRANS: E/O (PORT 2→1)	
TRANS: O/E (A/R)	
TRANS: O/E (B/R)	
TRANS: O/E (PORT 1→2)	

Table 5-4. Keys versus Programming Commands (31 of 32)

Softkey	Equivalent Programming Command
TRANS: O/E (PORT 2→1)	
TRANS: O/O (A/R)	
TRANS: O/O (B/R)	
TRANS: O/O (PORT 1→2)	
TRANS: O/O (PORT 2→1)	
TRANSFORM MENU	
TRANSFORM on OFF	`TIMDTRANON` or `TIMDTRANOFF`
TRANSMISSION	`FWDT`
TRIGGER MENU	
TRIGGER: TRIG OFF	`EXTTOFF`
TRL/LRL* 2-PORT*	`CALITRL2`
TTL I/O	
TTL OUT HIGH	`TTLOH`
TTL OUT LOW	`TTLOL`
TUNED RECEIVER	`INSMTUNR`
UNCOUPLED	`UNCPLD`
UP CONVERTER	`UCONV`
UPPER LIMIT	`LIMU`
USE MEMORY on OFF	`WINDUSEMON` or `WINDUSEMOFF`
USE PASS CONTROL	`USEPASC`
USE SENSOR A/B	`ENSA` or `ENSB`
VELOCITY FACTOR	`VELOFACT`
VIEW MEASURE	`VIEM`

Table 5-4. Keys versus Programming Commands (32 of 32)

Softkey	Equivalent Programming Command
VOLUME NUMBER	`DISCVOLU`
WAIT x	`SEQWAIT`
WARNING	`COLOWARN`
WARNING [ ]	`PCOLWARN`
WAVEGUIDE	`WAVE`
WIDTH VALUE	`WIDV`
WIDTHS on OFF	`WIDTON` or `WIDTOFF`
WINDOW	`WINDOW`
WINDOW: MAXIMUM	`WINDMAXI`
WINDOW: MINIMUM	`WINDMINI`
WINDOW: NORMAL	`WINDNORM`
WRITE EEPROM	
XMIT CNTRL [Xon-Xoff]	`PRNHNDSHKXON`
XMIT CNTRL [DTR-DSR]	`PRNHNDSHKDTR`
XMIT CNTRL [Xon-Xoff]	`PLTHNDSHKXON`
XMIT CNTRL [DTR-DSR]	`PLTHNDSHKDTR`
Y: Refl	`CONVYREF`
Y: Trans	`CONVYTRA`
Z: Refl	`CONVZREF`
Z: Trans	`CONVZTRA`

Index