Programming Manual

HP 8990A Peak Power Analyzer

SERIAL NUMBERS

Attached to the rear panel of the instrument is a serial number plate. The serial number is in the form: 0000A00000. The first four digits and the letter are the serial number prefix. The last five digits are the suffix. The prefix is the same for identical instruments; it changes only when a configuration change is made to the instrument. The suffix, however, is assigned sequentially and is different for each instrument.

This manual applies to instruments with serial numbers prefixed 3107A and above.

HEWLETT PACKARD

HP Part No. 08990-90002

1501 Page Mill Road, Palo Alto, California
Printed in USA

Introduction

This manual explains how to program the HP 8990A Peak Power Analyzer and lists the commands and queries associated with this instrument. It is divided into eighteen chapters and two appendices.

Chapter 1 introduces you to the programming syntax required to program this instrument and provides some basic programming concepts to get you started programming.

Chapter 2 describes the interface functions and some general concepts of HP-IB.

Chapter 3 describes the operation of instruments that operate in compliance with the **IEEE 488.2** standard. This chapter also describes the **status reporting** features that are available over the HP-IB.

Chapter 4 covers the conventions which are used to program the instrument as well as conventions used in the remainder of this manual. Also included are the following:

- A complete command tree
- A figure showing front panel menus and related HP-IB commands
- A figure showing front panel keys and related HP-IB commands
- An alphabetic command cross-reference.

Chapter 5 lists the **Common Commands** which are the commands defined by IEEE 488.2. These commands

control some functions that are common to all IEEE 488.2 instruments.

Chapter 6 lists the **Root Level Commands** which control many of the basic functions of the instrument.

Chapter 7 lists the **System Subsystem Commands** which control some basic functions of the Peak Power Analyzer.

Chapter 8 lists the **Acquire Subsystem Commands** which set the parameters for acquiring and storing data.

Chapter 9 lists the **Calibrate Subsystem Commands** which are used to set time nulls (channel-to-channel skew), and for zeroing at low power.

Chapter 10 lists the **Channel Subsystem Commands** which control all Y-axis Peak Power Analyzer functions.

Chapter 11 lists the **Display Subsystem Commands** which control how waveforms, amplitude and time markers, graticule, and text are displayed and written on the screen.

Chapter 12 lists the **Frequency Subsystem Command** which is used to specify the carrier frequency of the CW/pulsed source being measured.

Chapter 13 lists the **Function Subsystem Commands** which control the waveform math functions of the Peak Power Analyzer.

Chapter 14 lists the **Hardcopy Subsystem Commands** which control the parameters used during the plotting or printing of waveforms.

Chapter 15 lists the **Measure Subsystem Commands** which select the automatic measurements to be made.

Chapter 16 lists the **Timebase Subsystem Commands** which control all X-axis Peak Power Analyzer functions.

Chapter 17 lists the **Trigger Subsystem Commands** which control the trigger modes and parameters for each trigger mode.

Chapter 18 lists the **Waveform Subsystem Commands** which provide access to waveform data. Data can be active data from channels and functions as well as static data from waveform memories.

Appendix A provides details on how automatic measurements are calculated and offers some tips on how to improve results.

Appendix B contains **example programs** using the command set from the Peak Power Analyzer.

At the end of the manual is a complete **index** for easy reference of commands and functions.

Contents

Figures

Tables

xx This Page Intentionally Left Blank

CERTIFICATION

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

WARRANTY

This Hewlett-Packard instrument product is warranted against defects in material and workmanship for a period of one year from date of shipment. During the warranty period, Hewlett-Packard Company will, at its option, either repair or replace products which prove to be defective.

For warranty service or repair, this product must be returned to a service facility designated by HP. Buyer shall prepay shipping charges to HP and HP shall pay shipping charges to return the product to Buyer. However, Buyer shall pay all shipping charges, duties, and taxes for products returned to HP from another country.

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

LIMITATION OF WARRANTY

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

cont'd

LIMITATION OF WARRANTY (cont'd)

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

EXCLUSIVE REMEDIES

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

ASSISTANCE

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

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

1

Introduction to Programming the HP 8990A Peak Power Analyzer

Introduction

This chapter introduces you to the basic concepts of HP-IB communication and provides information and examples to get you started programming the HP 8990A Peak Power Analyzer. The exact mnemonics for the commands are listed in chapters 5 through 18.

There are four basic operations that can be done with a controller and Peak Power Analyzer via HP-IB. You can do the following:

1. Set up the Peak Power Analyzer and start making measurements
2. Retrieve setup information and measurement results
3. Digitize a waveform and pass the data to the controller
4. Send measurement data to the Peak Power Analyzer

Other more complicated tasks are accomplished with a combination of these four basic functions.

This chapter deals mainly with how to set up the Peak Power Analyzer, how to retrieve setup information and measurement results, how to digitize a waveform, and how to pass data to the controller. This chapter is divided into two sections. The first section concentrates on program syntax, and the second section discusses programming the Analyzer. Refer to the chapter "Measure Subsystem" for information on sending measurement data to the Peak Power Analyzer.

Controllers Other Than Hewlett-Packard

The programming examples in this manual are written in HP BASIC 5.0 for an HP 9000 Series 200/300 Controller. HP BASIC handles some of the redundant miscellaneous overhead associated with IEEE Standard 488.1 (HP-IB). For instance, when a BASIC "OUTPUT" statement is used (by the Active Controller) to send data to an HP-IB device, the following sequence of commands and data are sent over the bus:

```
OUTPUT 701;"Data"
```

1. The unlisten command is sent.
2. The talker's address command is sent (the address of the computer).
3. The listener's address command (01) is sent.
4. The data bytes "D", "a", "t", and "a" are sent.
5. Terminators CR and LF are sent.

All bytes are sent using the HP-IB's interlocking handshake to ensure that the listener has received each byte.

The example clearly shows that the HP BASIC "OUTPUT" statement causes more to take place besides the output of data. So, for controllers other than Hewlett-Packard which are using a programming language other than HP BASIC, additional steps may have to be added to the program examples given in the manual.

For more information, refer to IEEE Standard 488.1 and your programming language reference.

Programming Syntax

Talking to the Peak Power Analyzer

In general, computers acting as controllers communicate with the Peak Power Analyzer by passing messages over a remote interface using the I/O statements provided in the instruction set of the controller's host language. Hence, the messages for programming the Peak Power Analyzer, described in this manual, will normally appear as ASCII character strings imbedded inside the I/O statements of your controller's program. For example, the HP 9000 Series 200/300 BASIC language system uses the OUTPUT statement for sending program messages to the Peak Power Analyzer, and the ENTER statement for receiving response messages from the Peak Power Analyzer. Messages are placed on the bus by using an output command and passing the device selector, program message, and terminator. Passing the device selector ensures that the program message is sent to the correct interface and instrument.

The following command turns the command headers on:

```
OUTPUT < device selector > ;":SYSTEM:HEADER ON"
```

< device selector > represents the address of the device being programmed.

Note

The programming examples in this manual are written in HP Basic 5.0 for an HP 9000 Series 200/300 Controller.

The actual OUTPUT command you use when programming is dependent on the controller and the programming language you are using.

Angular brackets "< >," in this manual, enclose words or characters that symbolize a program code parameter or a bus command.

Information that is displayed in quotes represents the actual message that is sent across the bus. The message terminator (NL or EOI) is the only additional information that is also sent across the bus.

For HP 9000 Series 200/300 controllers, it is not necessary to type in the actual <terminator> at the end of the program message. These controllers automatically supply the program message terminator when the return key is pressed.

Addressing the Peak Power Analyzer

Since HP-IB can address multiple devices through the same interface card, the device selector passed with the program message must include not only the correct interface code, but also the correct instrument address.

Interface Select Code (Selects Interface). Each interface card has a unique interface select code. This code is used by the controller to direct commands and communications to the proper interface. The default is typically "7" for HP-IB controllers.

Instrument Address (Selects Instrument). Each instrument on an HP-IB bus must have a unique instrument address between decimal 0 and 30. The address must not be the address of the controller, which is usually decimal 21. The device address passed with the program message must include not only the correct

instrument address, but also the correct interface select code.

DEVICE SELECTOR = (Interface Select Code * 100) + (Instrument Address)

For example, if the instrument address for the Peak Power Analyzer is 7 and the interface select code is 7, when the program message is passed, the routine performs its function on the instrument at device selector 707.

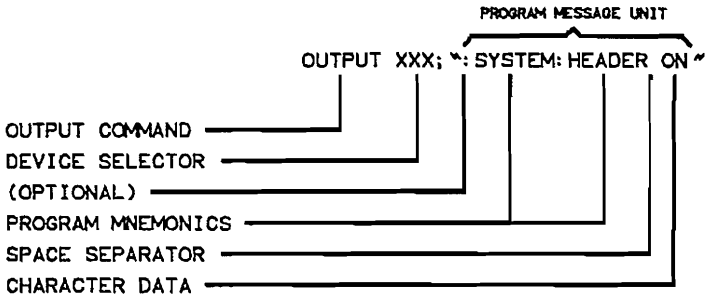
For the Peak Power Analyzer, the instrument address is typically set to "7" at the factory. This address can be changed in the HP-IB menu of the Utility menu.

Note

The program examples in this manual assume the Peak Power Analyzer is at device address 707.

Program Message Syntax

To program the Peak Power Analyzer over the bus, you must have an understanding of the command format and structure expected by the Peak Power Analyzer. The Peak Power Analyzer is remotely programmed with program messages. These are composed of sequences of program message units, with each unit representing a program command or query. A program command or query is composed of a sequence of functional elements that include separators, headers, program data, and terminators. These are sent to the Peak Power Analyzer over the system interface as a sequence of ASCII data messages. For example:



Separator

The <separator> shown in the program message refers to a blank space which is required to separate the program mnemonic from the program data.

Command Syntax

A command is composed of a header, any associated data, and a terminator. The header is the mnemonic or mnemonics that represent the operation to be performed by the Peak Power Analyzer. The different types of headers are discussed in the following paragraphs.

Simple Command Header. Simple command headers contain a single mnemonic. AUTOSCALE and DIGITIZE are examples of simple command headers typically used in the Peak Power Analyzer. The syntax is:

<program mnemonic><terminator>

When program data must be included with the simple command header (for example, :DIGITIZE CHAN1), a separator is added. The syntax is:

<program mnemonic><separator><program data><terminator>

Compound Command Header. Compound command headers are a combination of two or more program mnemonics. The first mnemonic selects the subsystem, and the last mnemonic selects the function within that subsystem. Additional mnemonics appear between the subsystem mnemonic and the function mnemonic when there are additional levels within the subsystem that must be transversed. The mnemonics within the compound message are separated by colons. For example:

To execute a single function within a subsystem, use the following:

:<subsystem>:<function><separator><program data><terminator>

(For example :SYSTEM:LONGFORM ON)

To transverse down a level of a subsystem to execute a subsystem within that subsystem:

:<subsystem>:<subsystem>:<function><separator> <program data><terminator>

(For example :TRIGGER:DELAY:SOURCE CHAN1)

To execute more than one function within the same subsystem a semi-colon is used to separate the functions:

:<subsystem>:<function><separator><data>; <function><separator><data><terminator>

(For example :SYSTEM:LONGFORM ON;HEADER ON)

Identical function mnemonics can be used for more than one subsystem. For example, the function mnemonic RANGE may be used to change the vertical range or to change the horizontal range:

```
:CHANNEL1:RANGE 7E-03
```

- sets the full scale vertical axis to 7 mW

```
:TIMEBASE:RANGE  1
```

- sets the horizontal timebase to 1 second full scale.

CHANNEL1 and TIMEBASE are subsystem selectors that determine which range is being modified.

Common Command Header. Common command headers control IEEE 488.2 functions within the Peak Power Analyzer (such as clear status, etc.). Their syntax is:

*<command header><terminator>

No space or separator is allowed between the asterisk and the command header. *CLS is an example of a common command header.

Query Command

Command headers immediately followed by a question mark (?) are queries. After receiving a query, the Peak Power Analyzer interrogates the requested function and places the answer in its output queue. The output message remains in the queue until it is read or another command is issued. When read, the message is transmitted across the bus to the designated listener (typically a controller). For example, the query :TIMEBASE:RANGE? places the current timebase setting in the output queue. In conjunction with this, the controller input statement:

```
ENTER <device selector>;Range$
```

passes the value across the bus to the controller and places it in the BASIC variable "Range$".

Query commands are used to find out how the Peak Power Analyzer is currently configured. They are also used to get results of measurements made by the Peak Power Analyzer, with the query actually activating the measurement. For example, the command :MEASURE:RISETIME? instructs the Peak Power Analyzer to measure the risetime of your waveform and place the result in the output queue.

Note

The output queue must be read before the next program message is sent. For example, when you send the query :MEASURE:RISETIME? you must follow that query with a program statement like, ENTER 707; Value_risetime$ to read the result of the query and place the result in a BASIC variable (Value_risetime$).

Sending another command before reading the result of the query will cause the output buffer to be cleared and the current response to be lost. This will also generate an error in the error queue.

Program Header Options

Program headers can be sent using any combination of uppercase or lowercase ASCII characters. Instrument responses, however, are always returned in uppercase.

Both program command and query headers may be sent in either longform (complete spelling), shortform (abbreviated spelling), or any combination of longform and shortform. Please note, ONLY the longform or shortform of a command will be accepted by the Peak Power Analyzer. Either of the following examples turn the headers on:

:SYSTEM:HEADER ON - longform

:SYST:HEAD ON - shortform

Programs written in longform are easily read and are almost self-documenting. The shortform syntax conserves the amount of controller memory needed for program storage and reduces the amount of I/O activity.

Note

The rules for shortform syntax are shown in the chapter "Programming and Documentation Conventions."

Program Data

Program data is used to convey a variety of types of parameter information related to the command header. At least one space must separate the command header or query header from the program data.

<program mnemonic><separator><data>
<terminator>

When a program mnemonic or query has multiple data parameters a comma separates sequential program data.

<program mnemonic><separator><data>,<data>
<terminator>

For example, :TRIGGER:DELAY TIME,1.23E-01 has two data parameters: TIME (character data) and 1.23E-01 (numeric data).

Character Program Data. Character program data is used to convey parameter information as alpha or alphanumeric strings. For example, the timebase command MODE can be set to auto, trigger, or single. The character program data in this case may be AUTO, TRIGGER, or SINGLE. :TIMEBASE:MODE SINGLE sets the timebase mode to single.

Numeric Program Data. Some command headers require program data to be a number. For example, :TIMEBASE:RANGE requires the desired full scale range to be expressed numerically. The Peak Power Analyzer recognizes integers, real numbers, and scientific notation. For more information see the chapter "Message Communication and System Functions."

Program Message Terminator

The program codes within a data message are executed after the program message terminator is received. The terminator may be either an NL (New Line) character, an EOI (End-Or-Identify) asserted, or a combination of the two. All three ways are equivalent with the exact encodings for the program terminators listed

in the chapter "Message Communication and System Functions." Asserting EOI sets the HP-IB EOI control line low on the last byte of the data message. The NL character is an ASCII linefeed (decimal 10).

Selecting Multiple Subsystems

You can send multiple program commands and program queries for different Peak Power Analyzer subsystems on the same line by separating each command with a semicolon. The colon following the semicolon enables you to enter a new subsystem. For example:

<program mnemonic><separator><data>;:<program mnemonic><separator><data> <terminator>

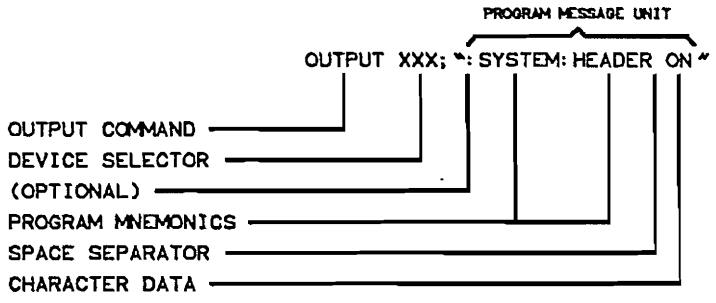
```
:CHANNEL1:RANGE 0.4;:TIMEBASE:RANGE 1
```

Note

Multiple commands may be any combination of compound and simple commands.

Summary

The following illustration summarizes the syntax for programming over the bus.



Programming the Peak Power Analyzer

Initialization

To make sure the bus and all appropriate interfaces are in a known state, begin every program with an initialization statement. For example:

```
CLEAR 707  ! initializes the interface of the
           ! Analyzer.
```

Then, initialize the Peak Power Analyzer to a preset state. For example:

```
OUTPUT 707;"*RST"  ! initializes the instrument
                   ! to a preset state.
```

Note

The actual commands and syntax for initializing the Peak Power Analyzer are discussed in the chapter "Common Commands."

Refer to your controller manual and programming language reference manual for information on initializing the interface.

Autoscale

The AUTOSCALE feature of the Peak Power Analyzer performs a very useful function on unknown waveforms by setting up the vertical channel, timebase, and trigger level of the Peak Power Analyzer.

The syntax for Autoscale is:

```
OUTPUT 707;":AUTOSCALE"
```

Setting Up the Peak Power Analyzer

A typical Peak Power Analyzer setup would set the vertical range, the horizontal range, delay time, delay reference, trigger mode, trigger level, and trigger slope. A typical example of the commands sent to the Analyzer are:

```
OUTPUT 707;":SYSTEM:POWER:UNITS WATTS"

OUTPUT 707;":CHANNEL1:RANGE 7E-03"

OUTPUT 707;":TIM:RANG 1E-06;DEL 20E-09;MODE TRIG"

OUTPUT 707;":TRIGGER:LEVEL .003W;SLOPE POSITIVE"
```

This example sets the display units to watts. With one screen displayed, the full scale display is 7mW or 875 μW per division. The horizontal time is 1 μs full-scale with a 20 ns delay. The timebase mode is set to triggered, and the trigger circuit is programmed to trigger at 0.003 watts on a positive slope.

Returning to Local

When placing the Peak Power Analyzer in local, use the BASIC command LOCAL <interface select code><device selector> and not LOCAL <device selector>. Using only the BASIC command LOCAL <device selector> returns the front panel of the Peak Power Analyzer to local, but the Peak Power Analyzer still accepts remote commands. For example:

```
10  REMOTE 707   !Peak Power Analyzer is
20               !placed in remote.
30  LOCAL 7      !Peak Power Analyzer accepts
40               !remote commands and commands
50               !from the front panel.
60  OUTPUT 707;":MENU FREQ"
70               !Send a command
70               !Peak Power Analyzer still
80               !accepts commands from the
90               !front panel.
100 LOCAL 707    !Peak Power Analyzer accepts
110              !commands from the front panel.
```

Aborting Lengthy Commands

Under certain conditions, some commands, especially the Root Level DIGITIZE command, can take a long time to process. It may be desired to abort the command before it is finished. The following text lists the different ways that can be used to abort the HP-IB command. The command that has minimal impact on the existing instrument state is listed first. Before any of the commands can be used, it is necessary to gain control of the controller by pausing the program or clearing the interface. Clearing the interface is done with a dedicated key (CLEAR I/O) on HP 9000 Series 200 Controllers.

Begin by using the first command, and then use as many as needed to abort the command:

```
ABORT 7      !Minimum change, simply clears the
             !handshake lines. (ABORT message)
```

```
CLEAR 707   !The input and output buffers
            !are cleared, the parser is reset, and any
            !pending commands are cleared. (Selective
            !Device Clear message)
CLEAR 7     !The same actions are taken as with
            !CLEAR 707, except to all instruments
            !on interface select code 7. (Device
            !Clear message)
OUTPUT 707;"*RST"
            !Peak Power Analyzer is reset to a
            !pre-defined state.
```

Receiving Information from the Peak Power Analyzer

After receiving a query (command header followed by a question mark), the Peak Power Analyzer interrogates the requested function and places the answer in its output queue. The answer remains in the output queue until it is read or another command is issued. When read, the message is transmitted across the bus to the controller. The input statement for receiving a response message from an instrument's output queue typically has two parameters; the device address and a format specification for handling the response message. For example, to read the result of the query command :SYSTEM:LONGFORM?, you would execute the statement:

```
ENTER <device selector>;Setting$
```

where <device selector> represents the address of your device. This would enter the current setting for the longform command in the string variable Setting$.

Note

All results for queries sent in a program message must be read before another program message is sent. For example, when you send the query :MEASURE:RISETIME?, you must follow that query with the program statement ENTER 707;Risetime$ to read the result of the query and place the result in a variable (Risetime$).

Sending another command before reading the result of the query will cause the output buffer to be cleared and the current response to be lost. This will also cause an error to be placed in the error queue. Executing an ENTER statement before sending a query will cause the controller to wait indefinitely.

The actual ENTER program statement you use when programming is dependent on the programming language you are using.

The format specification for handling the response message is dependent on both the controller and the programming language.

Response Header Options

The format of the returned ASCII string depends on the current settings of the SYSTEM HEADER and LONGFORM commands. The general format is:

```
<header><separator><data><terminator>
```

The header identifies the data that follows and is controlled by issuing a **:SYSTEM:HEADER ON/OFF** command. If the state of the header command is OFF, only the data is returned by the query. The format of the header is controlled by the **:SYSTEM:LONGFORM ON/OFF** command. If longform is OFF, the header will be in its shortform. The following would be returned from a :MEASURE:FREQUENCY? query:

<data><terminator> (with HEADER OFF)

:MEAS:FREQ<separator><data><terminator>
(with HEADER ON/LONGFORM OFF)

:MEASURE:FREQUENCY<separator><data>
<terminator> (with HEADER ON/LONGFORM ON)

Note

A command or query may be sent in either longform or shortform, or in any combination of longform and shortform. The HEADER and LONGFORM commands only control the format of the returned data and have no effect on the way commands are sent. Common commands never return a header.

Refer to the chapter "System Subsystem" for more information on turning the HEADER and LONGFORM commands on and off.

Response Data Formats

Most data will be returned as exponential or integer numbers. However, query data of some instrument setups is returned as character data. Interrogating the trigger SLOPE? will return one of the following:

:TRIGGER:SLOPE **POSITIVE** <terminator> (with HEADER ON/ LONGFORM ON)

:TRIG:SLOP **POS**<terminator> (with HEADER ON/LONGFORM OFF)

POSITIVE<terminator> (with HEADER OFF/LONGFORM ON)

POS<terminator> (with HEADER OFF/LONGFORM OFF)

Note

Refer to the individual commands in this manual for information on the format (alpha or numeric) of the data returned from each query.

String Variables

If you want to observe the headers for queries, you must bring the returned data into a BASIC string variable. Reading queries into string variables is simple and straightforward, requiring little attention to formatting. For example:

```
ENTER <device selector>;Result$
```

places the output of the query in the string variable Result$.

Note

BASIC string variables are case sensitive and must be expressed exactly the same each time they are used.

The output of the Peak Power Analyzer may be numeric or character data depending on what is queried. Refer to the specific commands for the formats and types of data returned from queries.

Note

For the example programs, assume that the device being programmed is at device selector 707. The actual address will vary according to how you have configured the bus for your own application.

The following example shows the data being returned to a string variable with headers off:

```
10 DIM Rang$[30]
20 OUTPUT 707;":SYSTEM:HEADER OFF"
30 OUTPUT 707;":CHANNEL1:RANGE?"
40 ENTER 707;Rang$
50 PRINT Rang$
60 END
```

After running this program, the controller displays:

```
+7.00000E-03
```

Numeric Variables

If you do not need to see the headers when a numeric value is returned from the Peak Power Analyzer , then you can use a numeric variable. When you enter numeric data from the Peak Power Analyzer into a numeric variable, turn the headers off.

Note

When you enter numeric data from the Peak Power Analyzer into numeric variables, the headers should be turned off. Otherwise the headers may cause misinterpretation of returned data.

The following example shows the data being returned to a numeric variable.

```
10 OUTPUT 707;":SYSTEM:HEADER OFF"
20 OUTPUT 707;":CHANNEL1:RANGE?"
30 ENTER 707;Rang
40 PRINT Rang
50 END
```

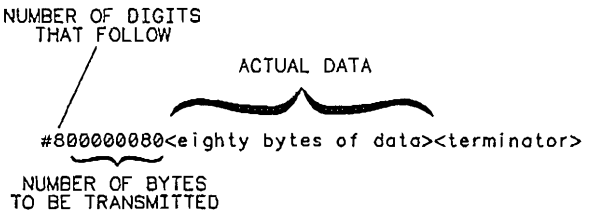
After running this program, the controller displays:

```
.007
```

Definite-Length Block Response Data

Definite-length block response data allows any type of device-dependent data to be transmitted over the system interface as a series of 8-bit binary data bytes. This is particularly useful for sending large quantities of data. The syntax is a pound sign (#) followed by a non-zero digit representing the number of digits in the decimal integer. After the non-zero digit is the decimal integer that states the number of 8-bit data bytes being sent. This is followed by the actual data.

For example, in order to transmit 80 bytes of data, the syntax would be:



The first "8" after the # sign states the number of digits that follow, and "00000080" states the number of bytes (in decimal) to be transmitted.

Multiple Queries

You can send multiple queries to the Peak Power Analyzer within a single program message, but you must also read them back within a single program message. This can be accomplished by either reading the queries back into a string variable or into multiple numeric variables. For example, you could read the result of the query :SYSTEM:HEADER?;LONGFORM? into the string variable Results$ with the command:

```
ENTER 707;Results$
```

When you read the result of multiple queries into string variables, each response is separated by a semicolon. For example, the response of the query :SYSTEM:HEADER?;LONGFORM? with HEADER and LONGFORM on would be:

```
:SYSTEM:HEADER 1;:SYSTEM:LONGFORM 1
```

If you do not need to see the headers when the numeric values are returned, then you can use

the following program message to read the query :SYSTEM:HEADERS?;LONGFORM? into multiple numeric variables:

```
ENTER 707;Result1,Result2
```

Note

When you enter numeric data into numeric variables, the headers should be turned off. Otherwise, the headers may cause formatting errors or misinterpretation of returned data.

Instrument Status

Status registers track the current status of the Peak Power Analyzer. By checking the instrument status, you can find out whether an operation has been completed, whether the Peak Power Analyzer is receiving triggers, and more. The chapter "Message Communication and System Functions" explains how to check the status of the instrument.

Digitize Command

The ACQUIRE and WAVEFORM subsystems are subsystems that affect the DIGITIZE command. The DIGITIZE command is used to capture a waveform in a known format which is specified by the ACQUIRE subsystem. When the DIGITIZE command is sent to the Peak Power Analyzer, the specified channel signal is digitized with the current ACQUIRE parameters. To obtain waveform data, you must specify the WAVEFORM parameters for the waveform data prior to sending the :WAVEFORM:DATA? query.

The number of data points comprising a waveform varies according to the number requested in the ACQUIRE subsystem. The ACQUIRE subsystem determines the number of data points, type of acquisition, and number of averages used by the DIGITIZE command. This allows you to specify exactly what the digitized information will contain. A typical setup is:

```
OUTPUT 707;":ACQUIRE:TYPE AVERAGE"
```

```
OUTPUT 707;":ACQUIRE:COMPLETE 100"
OUTPUT 707;":WAVEFORM:SOURCE CHANNEL1"
OUTPUT 707;":WAVEFORM:FORMAT ASCII"
OUTPUT 707;":ACQUIRE:COUNT 4"
OUTPUT 707;":ACQUIRE:POINTS 500"
OUTPUT 707;":DIGITIZE CHANNEL1"
OUTPUT 707;":WAVEFORM:DATA?"
```

This setup places the Peak Power Analyzer into the average mode with four averages and defines the data record to be 500 points. This means that when the DIGITIZE command is received, the waveform is not stored in memory until 500 points have been averaged at least four times.

After receiving the **:WAVEFORM:DATA?** query, the Peak Power Analyzer starts outputting the waveform information when addressed to talk.

Digitized waveforms are transferred from the Peak Power Analyzer to the controller by sending a numerical representation of each digitized point. The format of the numerical representation is controlled with the **:WAVEFORM:FORMAT** command and may be selected as **ASCII**, **WORD**, **BYTE**, or **COMPRESSED**.

The easiest method of entering a digitized waveform from the Peak Power Analyzer is to use the ASCII format and place the information in an integer array. The data point is represented by signed six-digit integers whose values range from 0 to 32,640. You must scale the integers with values from a waveform header (separately queried) to determine the amplitude value of each point. These integers are passed starting with the leftmost point on the Peak Power Analyzer's display. For more information, refer to the chapter "Waveform Subsystem."

2

Interface Functions

Introduction

This section describes the interface functions and some general concepts of the HP-IB. In general, these functions are defined by IEEE 488.1. They deal with general bus management issues, as well as messages which can be sent over the bus as bus commands.

Interface Capabilities

The interface capabilities of the Peak Power Analyzer, as defined by IEEE 488.1 are SH1, AH1, T5, L4, SR1, RL1, PP1, DC1, DT1, C0, and E2.

Command and Data Concepts

The HP-IB has two modes of operation: command mode and data mode. The bus is in command mode when the ATN line is true. The command mode is used to send talk and listen addresses and various bus commands, such as a group execute trigger (GET). The bus is in the data mode when the ATN line is false. The data mode is used to convey device-dependent messages across the bus. The device-dependent messages include all of the instrument commands and responses found in chapters 5 through 18 of this manual.

Addressing

By using the front-panel menus, the Peak Power Analyzer can be placed in either talk-only mode or addressed (talk/listen) mode (see your Operating Manual).

Addressed Mode

Addressed mode is used when the Peak Power Analyzer will operate in conjunction with a controller. When the Peak Power Analyzer is in the addressed mode, the following is true:

- Each device on the HP-IB bus resides at a particular address, ranging from 0 to 30. Address 21 is usually reserved for the controller.
- The active controller specifies which devices will talk and which will listen.
- An instrument, therefore, may be talk-addressed, listen-addressed, or unaddressed by the controller.

If the controller addresses the instrument to talk, it will remain configured to talk until it receives an interface clear message (IFC), another instrument's talk-address (OTA), its own listen address (MLA), or a universal untalk command (UNT).

If the controller addresses the instrument to listen, it remains configured to listen until it receives an interface clear message (IFC), its own talk-address (MTA), or a universal unlisten command (UNL).

Refer to the BASIC I/O documentation for more information on these messages.

Talk-Only Mode

Talk-only mode is used when you want to make a hardcopy of the Peak Power Analyzer display using an HP-IB listen-only printer.

Remote, Local and Local Lockout

The local, remote, and remote with local lockout modes may be used for various degrees of front-panel control while a program is running. The Peak Power Analyzer will accept and execute bus commands while in local mode, and the front panel will also be entirely active. If the Peak Power Analyzer is in remote mode, all controls (except the power switch and the LOCAL key) are entirely locked out. Local control can only be restored by the controller or pressing the front-panel LOCAL key.

Note

Cycling the power will also restore local control, but this will also reset certain HP-IB states.

The Peak Power Analyzer is placed in remote mode by first setting the REN bus control line true, and then addressing the instrument to listen. The Peak Power Analyzer can be placed in local lockout mode by sending the local lockout command (LLO). The Peak Power Analyzer is returned to local mode by either setting the REN line false, sending the instrument the go-to-local command (GTL), or simulating a front-panel LOCAL key press using the SYSTEM:KEY command.

Bus Commands

The following commands are IEEE 488.1 bus commands (ATN true). IEEE 488.2 defines many of the actions which are taken when these commands are received by the Peak Power Analyzer.

Device Clear

The device clear (DCL) or selected device clear (SDC) commands clear the input and output buffers, reset the parser, and clear any pending commands.

Group Execute Trigger (GET)

The group execute trigger command arms the trigger (which is the same action produced by sending the RUN command).

Interface Clear (IFC)

This command halts all bus activity. This includes unaddressing all listeners and the talker, disabling serial poll on all devices, and returning control to the system controller.

Status Annunciators

The Peak Power Analyzer will display the HP-IB status on the CRT. The message will indicate whether the instrument is in remote mode, whether talk or listen is addressed, and whether the Peak Power Analyzer has requested service. When the Peak Power Analyzer is in local mode only the SRQ annunciator may be displayed.

3

Message Communication and System Functions

Introduction

This chapter describes the operation of instruments that operate in compliance with the IEEE 488.2 standard. Instruments that are compatible with IEEE 488.2 must also be compatible with IEEE 488.1. However IEEE 488.1 compatible instruments may or may not conform to the IEEE 488.2 standard. The IEEE 488.2 standard defines the message exchange protocols by which the instrument and the controller will communicate. It also defines some common capabilities, which are found in all IEEE 488.2 instruments. This chapter also contains a few items which are not specifically defined by IEEE 488.2, but deal with message communication or system functions.

Protocols

The protocols of IEEE 488.2 define the overall scheme used by the controller and the instrument to communicate. This includes defining when it is appropriate for devices to talk or listen and what happens when the protocol is not followed.

Functional Elements

Before proceeding with the description of the protocol, a few system components should be understood.

Input Buffer. The input buffer of the instrument is the memory area where commands and queries are stored prior to being parsed and executed. It allows a controller to send a string of commands to the instrument which could take some time to execute, and then proceed to talk to another instrument while the first is parsing and executing commands. The Peak Power Analyzer's input buffer will hold 300 characters, or bytes of data.

Output Queue. The output queue of the instrument is the memory area where all output data (<response messages>) are stored until read by the controller. The Peak Power Analyzer's output queue will hold 300 characters. However, the Peak Power Analyzer will handle block data of greater than 300 characters where appropriate.

Parser. The instrument's parser is the component that interprets the commands sent to the instrument and decides what actions should be taken. "Parsing" refers to the action taken by the parser to achieve this goal. Parsing and executing of commands begins when either the instrument sees a <program message terminator> (defined later in this chapter) or the input buffer becomes full. If you wish to send a long sequence of commands to be executed and talk to another instrument while they are executing, you should send all of the commands before sending the <program message terminator>.

Protocol Overview

The instrument and controller communicate by using <program message>s and <response message>s. These messages serve as the containers into which sets of program commands or instrument responses are placed. <program message>s are sent by the controller to the instrument, and <response message>s are sent from the instrument to the controller in response to a query message. A "query message" is defined as being a <program message> which contains one or more queries.

The instrument will only talk when it has received a valid query message, and, therefore, has something to say. The controller should only attempt to read a response after sending a complete query message, but before sending another <program message>. The basic rule to remember is that the instrument will only talk when prompted to, and the output queue must be read before the instrument is told to do something else.

Protocol Operation

When the instrument is powered on or when it receives a device clear command, the input buffer and output queue are cleared, and the parser is reset to the root level of the command tree.

The instrument and the controller communicate by exchanging complete <program message>s and <response message>s. This means that the controller should always terminate a <program message> before attempting to read a response. The instrument will terminate <response message>s except in talk-only mode during a hardcopy output.

If a query message is sent, the next message passing over the bus should be the <response message>. The controller should always read the complete <response message> associated with a query message before sending another <program message> to the same instrument.

The instrument allows the controller to send multiple queries in one query message. This is referred to as sending a "compound query." As will be noted later in this chapter, multiple queries in a query message are separated by semicolons. The responses to each of the queries in a compound query will also be separated by semicolons.

Commands are executed in the order that they are received. This also applies to the reception of the group execute trigger (GET) bus command. The group execute

trigger command should not be sent in the middle of a <program message>.

Protocol Exceptions

If an error occurs during the information exchange, the exchange may not be completed in a normal manner. Some of the protocol exceptions are shown below.

Addressed to talk with nothing to say. If the instrument is addressed to talk before it receives a query, it will indicate a "query unterminated" error and will not send any bytes over the bus; in addition, the bus will hang. If the instrument has nothing to say because queries requested were unable to be executed because of some error, the device will not indicate a query error, it will simply wait to receive the next message from the controller.

Addressed to talk with no listeners on the bus. If the instrument is addressed to talk, and there are no listeners on the bus, the instrument will wait for a listener to listen or for the controller to take control.

Command Error. A command error will be reported if the instrument detects a syntax error or an unrecognized command header.

Execution Error. An execution error will be reported if a parameter is found to be out of range, or if the current settings do not allow execution of a requested command or query.

Note

When a command is found to have a parameter out of range, a serial poll immediately following the command occasionally may not immediately indicate an error. Processing of the next command begins before the previous error is reported. If it is necessary to know that an error has occurred after each message unit, a wait statement of one milli-second can be included in the program to allow the Peak Power Analyzer time to report the error.

The Peak Power Analyzer was designed to operate without the wait statement in order to increase overall bus throughput.

Device-specific Error. A device-specific error will be reported if the instrument is unable to execute a command for a strictly device-dependent reason.

Query Error. A query error will be reported if the proper protocol for reading a query is not followed. This includes the interrupted and unterminated conditions described below.

Unterminated Condition. If the controller attempts to read a <response message> before terminating the <program message>, a query error will be generated. The parser will reset itself, and the response will be cleared from the output queue of the instrument without being sent over the bus. This condition also happens if the instrument is addressed to talk and it has nothing to say.

Interrupted Condition. If the controller does not read the entire <response message> generated by a query message and then attempts to send another <program message>, the device will generate a query error. The unread portion of the response will then be discarded by the instrument. The interrupting <program message> will not be affected.

Buffer Deadlock. The instrument may become deadlocked if the input buffer and output queue both become full. This condition can occur if a very long <program message> is sent containing queries that generate a great deal of response data. The instrument cannot accept any more bytes, and the controller cannot read any of the response data until it has completed sending the entire <program message>. Under this condition, the instrument will break the deadlock by clearing the output queue and continuing to discard responses until it comes to the end of the current <program message>. The query error bit will also be set.

Syntax Diagrams

The syntax diagrams in this chapter are similar to the syntax diagrams in the IEEE 488.2 standard. Commands and queries are sent to the instrument as a sequence of data bytes. The allowable byte sequence for each functional element is defined by the syntax diagram that is shown with the element description.

The allowable byte sequence can be determined by following a path in the syntax diagram. The proper path through the syntax diagram is any path that follows the direction of the arrows. If there is a path around an element, that element is optional. If there is a path from right to left around one or more elements, that element or those elements may be repeated as many times as desired.

Syntax Overview

This overview is intended to give a quick glance at the syntax defined by IEEE 488.2. It should allow you to understand many of the things about the syntax that you need to know. This chapter also contains the details of the IEEE 488.2 defined syntax.

IEEE 488.2 defines the blocks used to build messages which are sent to the instrument. A whole string of commands can, therefore, be broken up into individual components.

Figure 3-1 shows a breakdown of an example <program message>. There are a few key items to notice:

1. A semicolon separates commands from one another. Each <program message unit> serves as a container for one command. The <program message unit>s are separated by a semicolon.
2. A <program message> is terminated by a <NL> (new line), a <NL> with EOI asserted, or EOI being asserted on the last byte of the message. The recognition of the <program message terminator>, or <PMT>, by the parser serves as a signal for the parser to begin execution of commands. The <PMT> also affects command tree traversal (see the Programming and Documentation Conventions chapter).
3. Multiple data parameters are separated by a comma.
4. The first data parameter is separated from the header with one or more spaces.
5. The header MEAS:SOURCE is a compound header. It places the parser in the measure subsystem until the <NL> is encountered.
6. A colon after a semicolon places the parser at the root level of the command tree.

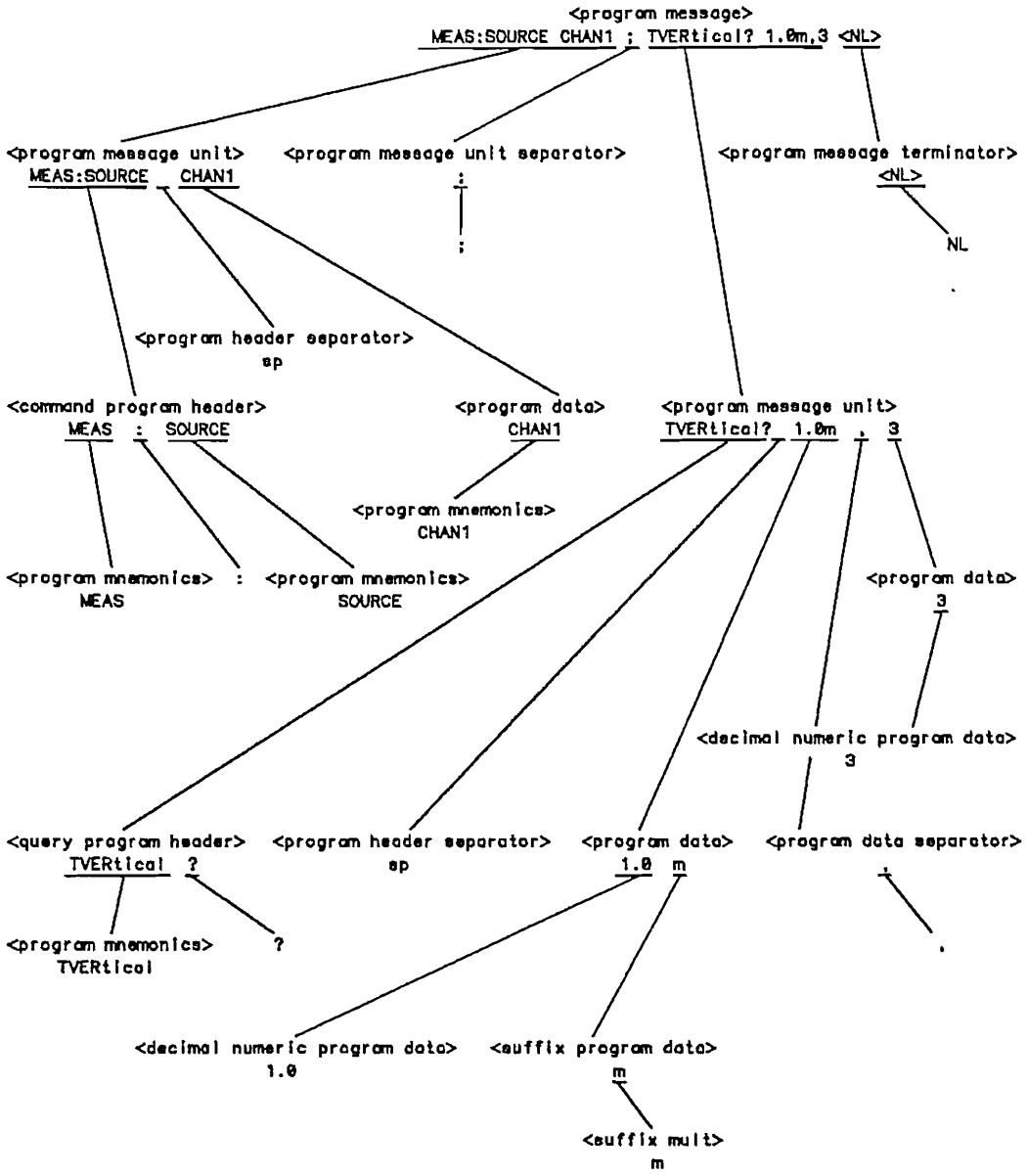


Figure 3-1. <program message> Parse Tree

Device Listening Syntax

The listening syntax of IEEE 488.2 is designed to be more forgiving than the talking syntax. This allows greater flexibility in writing programs, as well as allowing them to be easier to read.

Upper/Lower Case Equivalence. Upper and lower case letters are equivalent. The mnemonic RANGE has the same semantic meaning as the mnemonic range.

<white space>. <white space> is defined to be one or more characters from the ASCII set of 0—32 decimal, excluding 10 decimal (NL). <white space> is used by several instrument listening components of the syntax.

It is usually optional, and can be used to increase the readability of a program.

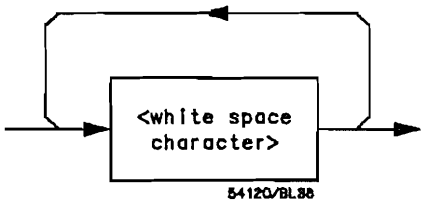


Figure 3-2. <white space>

<program message>. The <program message> is a complete message to be sent to the instrument. The instrument will begin executing commands once it has a complete <program message>, or when the input buffer becomes full. The parser is also repositioned to the root of the command tree after executing a complete <program message>. Refer to the Tree Traversal Rules in the Programming and Documentation Conventions chapter for more details.

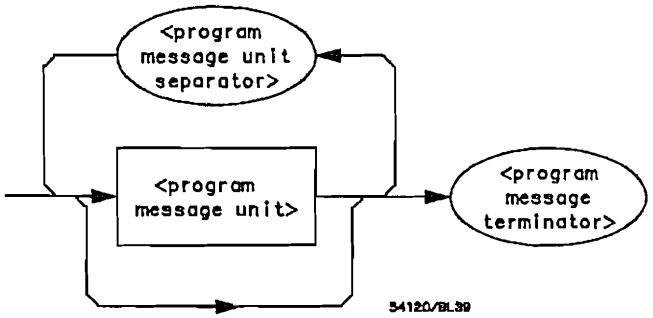


Figure 3-3. <program message>

<program message unit>. The <program message unit> is the container for individual commands within a <program message>.

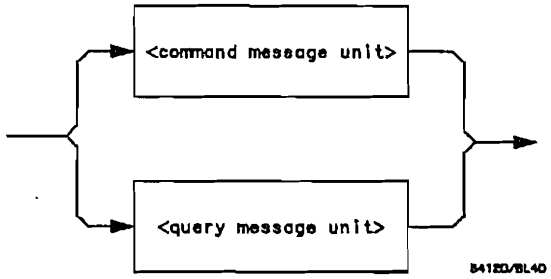


Figure 3-4. <program message unit>

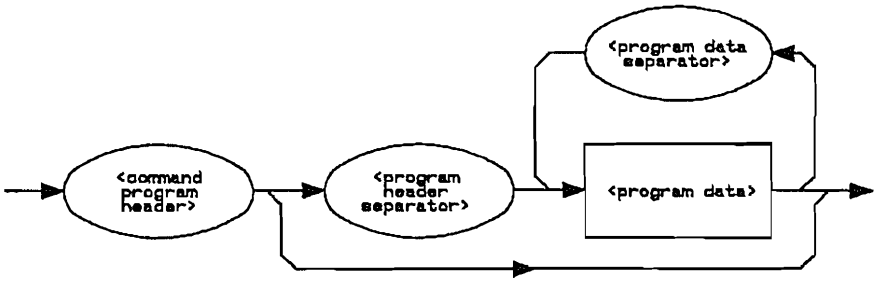


Figure 3-5. <command message unit>

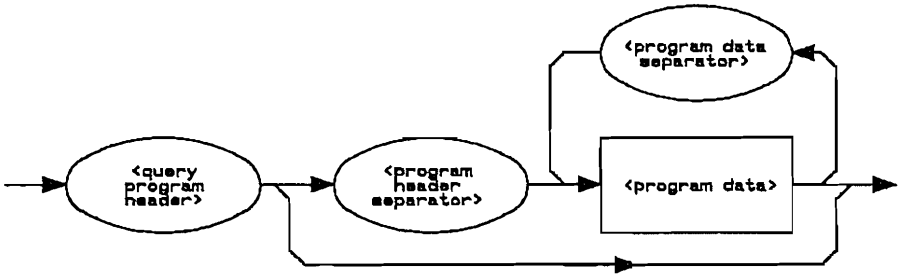


Figure 3-6. <query message unit>

<program message unit separator>. A semicolon separates <program message unit>s, or individual commands.

<command program header>/<query program header>. These elements serve as the headers of commands or queries. They represent the action to be taken.

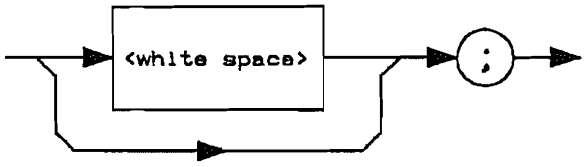
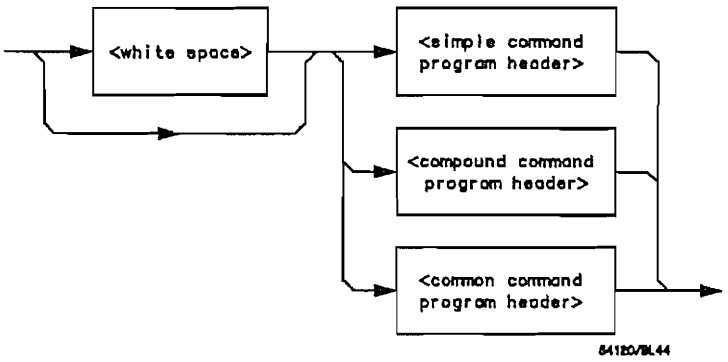
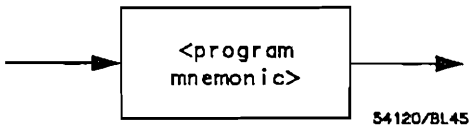


Figure 3-7. <program message unit separator>



Where <simple command program header> is defined as



Where <compound command program header> is defined as

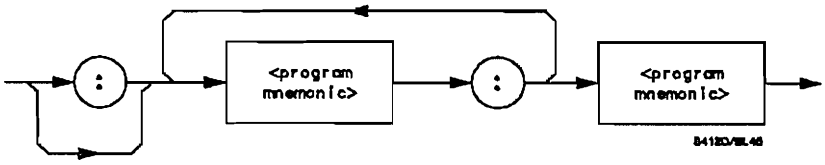
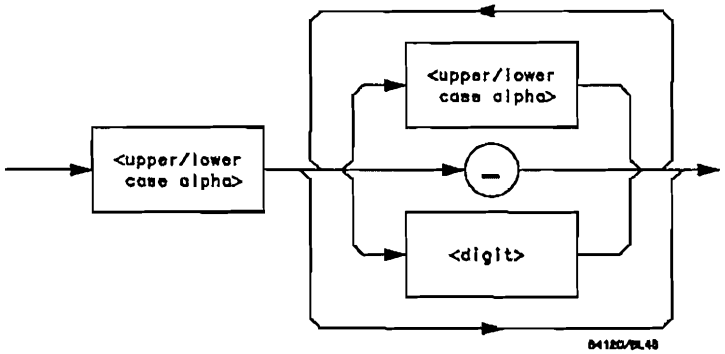


Figure 3-8. <command program header>

Where <common command program header> is defined as

Where <program mnemonic> is defined as

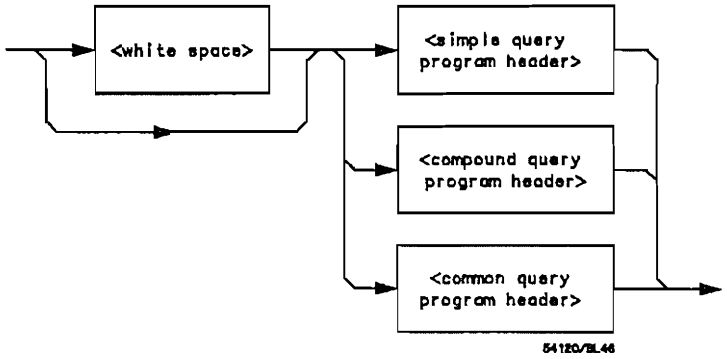


Where <upper/lower case alpha> is defined as a single ASCII encoded byte in the range 41–5A, 61–7A (65–90, 97–122 decimal).

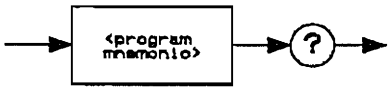
Where <digit> is defined as a single ASCII encoded byte in the range 30–39 (48–57 decimal).

Where (_) represents an "underscore", a single ASCII-encoded byte with the value 5F (95 decimal).

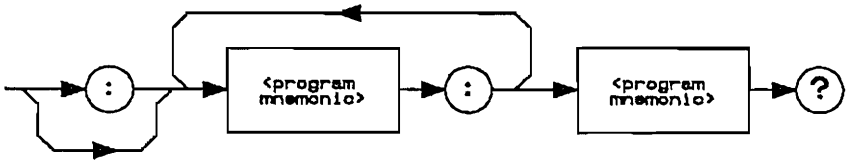
Figure 3-8. <command program header> (continued)



Where <simple query program header> is defined as



Where <compound query program header> is defined as



Where <common query program header> is defined as

Figure 3-9. <query program header>

<program data>. The <program data> element represents the possible types of data which may be sent to the instrument. The Peak Power Analyzer will accept the following data types: <character program data>, <decimal numeric program data>, <suffix program data>, <string program data>, and <arbitrary block program data>.

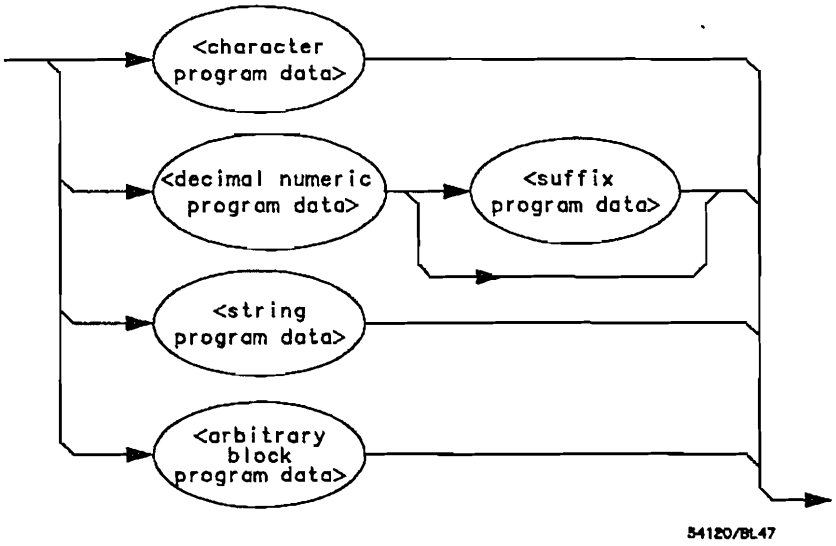
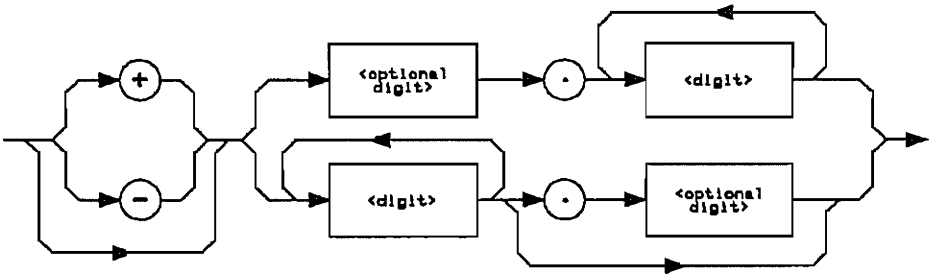


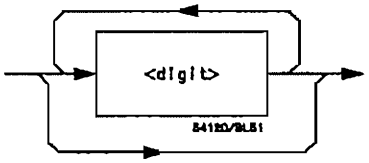
Figure 3-10. <program data>

Figure 3-11. <character program data>

Where <mantissa> is defined as



Where <optional digits> is defined as



Where <exponent> is defined as

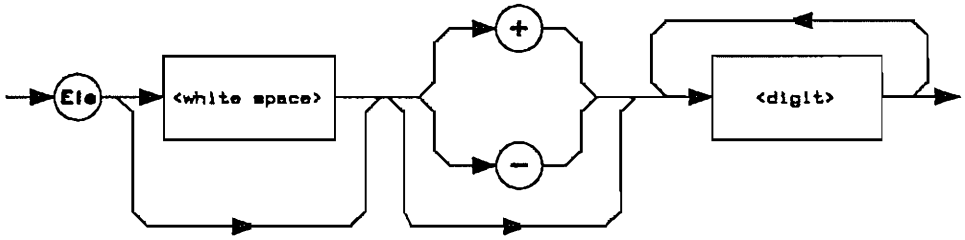


Figure 3-12. <decimal numeric program data>

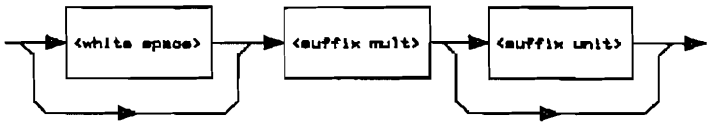


Figure 3-13. <suffix program data>

Suffix Multiplier. The suffix multipliers that the Peak Power Analyzer will accept are shown in table 3-1.

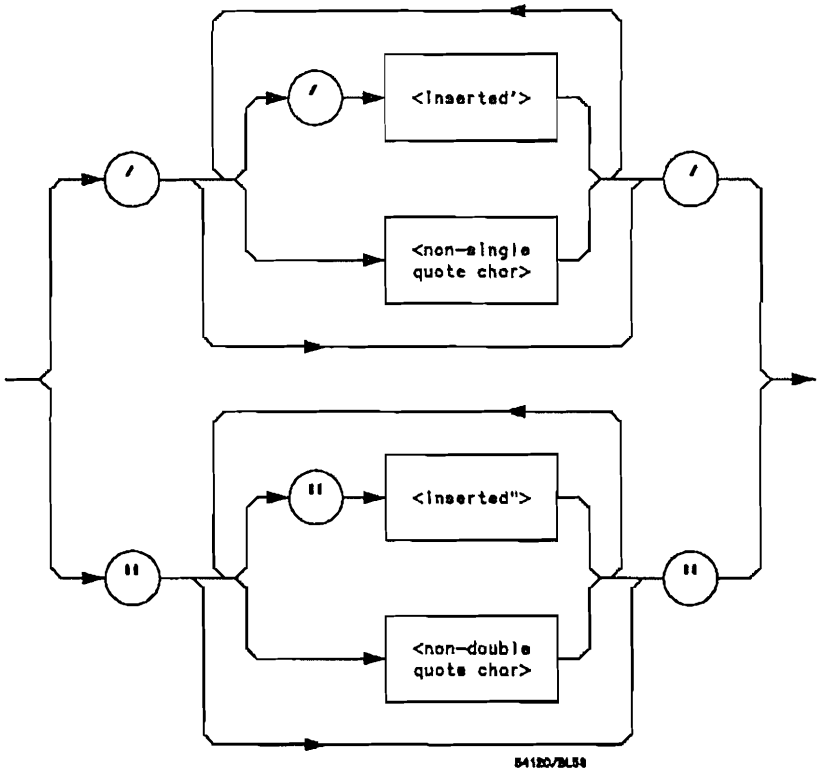
Table 3-1. <suffix mult>

Value	Mnemonic	Value	Mnemonic
1E15	PE	1E-3	M
1E12	T	1E-6	U
1E9	G	1E-9	N
1E6	MA	1E-12	P
1E3	K	1E-15	F
		1E-18	A

Suffix Unit. The suffix units that the Peak Power Analyzer will accept are shown in table 3-2.

Table 3-2. <suffix unit>

Suffix	Referenced Unit
V	Volt
S	Second
W	Watt
DBM	dBm
DB	dB
HZ	Hertz



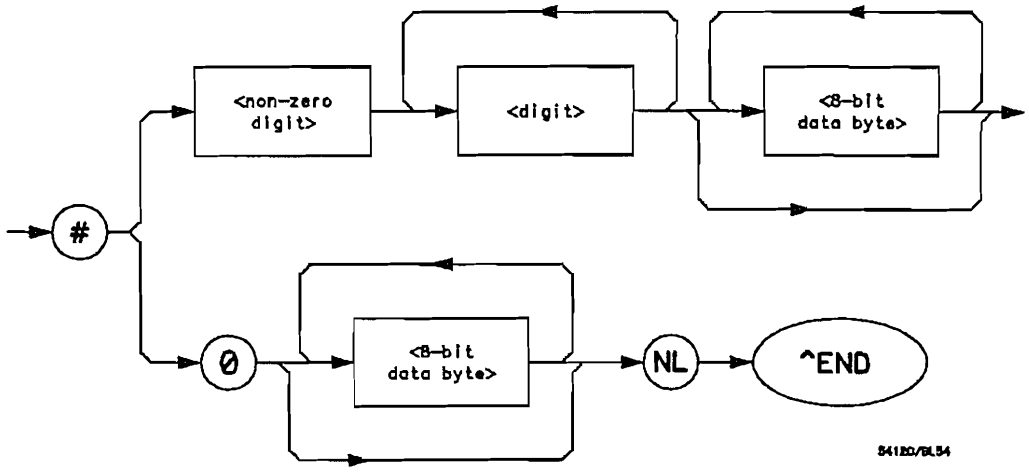
Where <inserted'> is defined as a single ASCII character with the value 27 (39 decimal).

Where <non-single quote char> is defined as a single ASCII character of any value except 27 (39 decimal).

Where <inserted"> is defined as a single ASCII character with the value 22 (34 decimal).

Where <non-double quote char> is defined as a single ASCII character of any value except 22 (34 decimal).

Figure 3-14. <string program data>



Where <non-zero digit> is defined as a single ASCII encoded byte in the range 31—39 (49—57 decimal).

Where <8-bit byte> is defined as an 8-bit byte in the range 00—ff (0—255 decimal).

Figure 3-15. <arbitrary block program data>

<program data separator>. A comma separates multiple data parameters of a command from one another.

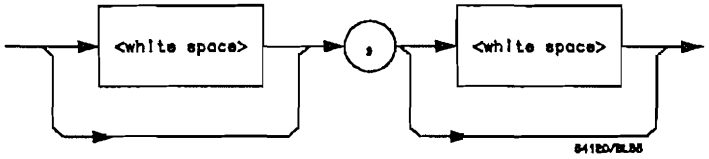
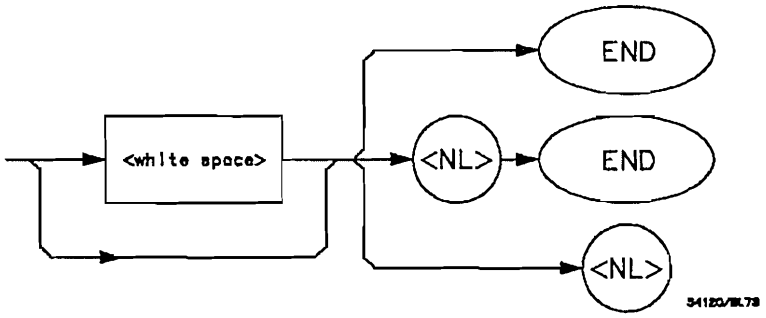


Figure 3-16. <program data separator>

<program header separator>. A space (ASCII decimal 32) separates the header from the first or only parameter of the command.

Figure 3-17. <program header separator>

<program message terminator>. The <program message terminator> or <PMT> serves as the terminator to a complete <program message>. When the parser sees a complete <program message> it will begin execution of the commands within that message. The <PMT> also resets the parser to the root of the command tree.



While <NL> is defined as a single ASCII-encoded byte 0A (10 decimal).

Figure 3-18. <program message terminator>

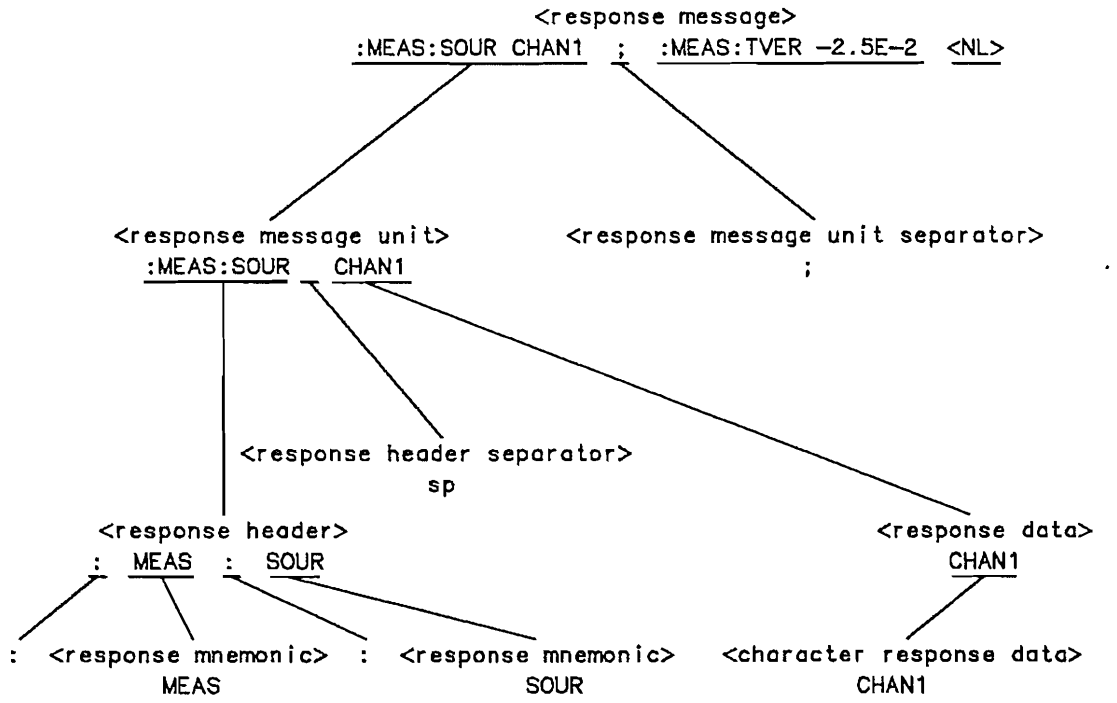


Figure 3-19. <response message Tree>

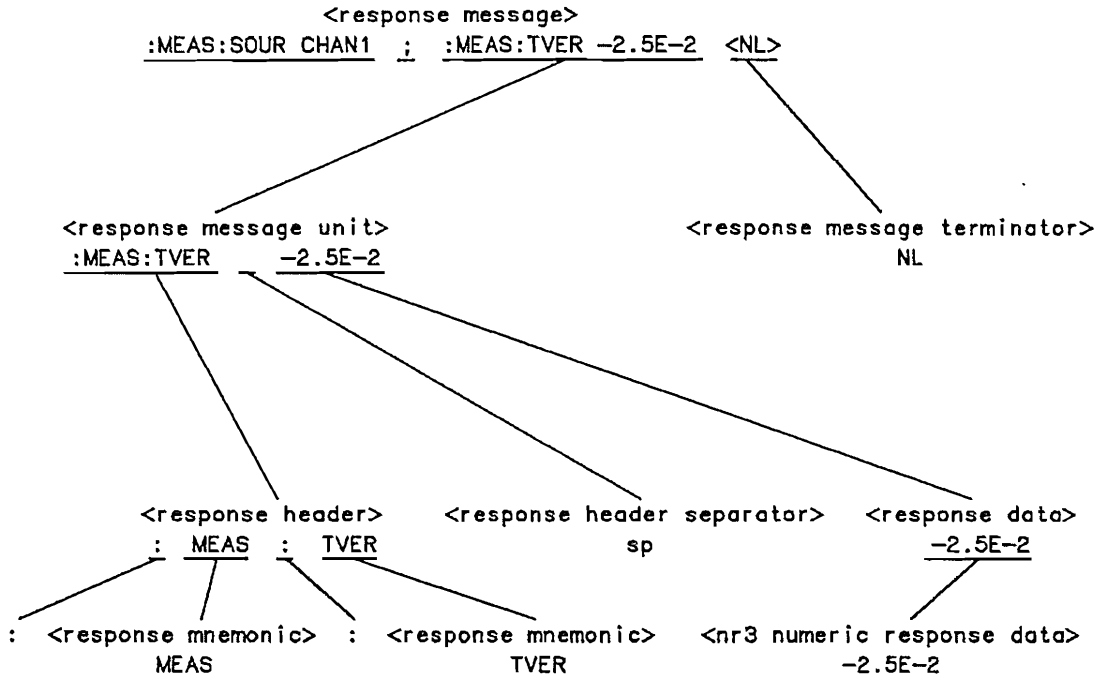


Figure 3-19. <response message Tree> (continued)

Device Talking Syntax

The talking syntax of IEEE 488.2 is designed to be more precise than the listening syntax. This allows the programmer to write routines which can more easily interpret and use the data the instrument is sending. One of the implications of this is the absence of <white space> in the talking formats. The instrument will not pad messages which are being sent to the controller with spaces.

<response message>. This element serves as a complete response from the instrument. It is the result of the instrument executing and buffering the results from a complete <program message>. The complete <response message> should be read before sending another <program message> to the instrument.

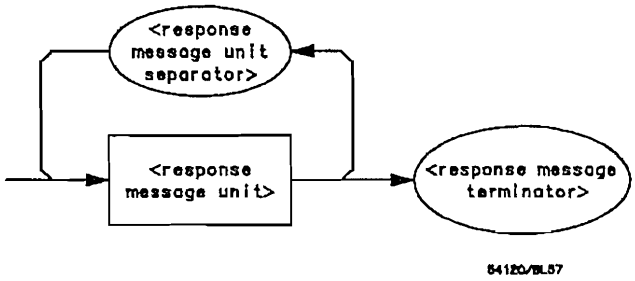
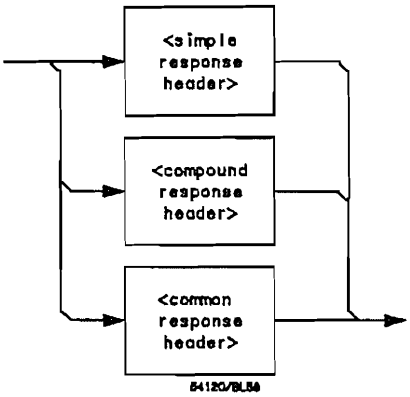


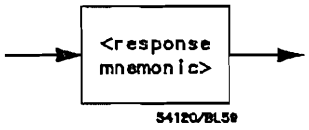
Figure 3-20. <response message>

<response message unit>. This element serves as the container of individual pieces of a response. Typically, a <query message unit> will generate one <response message unit>, although a <query message unit> may generate multiple <response message unit>s.

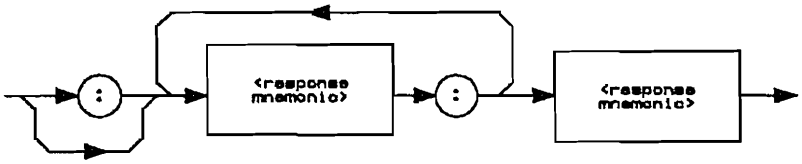
<response header>. The <response header>, when returned, indicates what the response data represents.



Where <simple response mnemonic> is defined as



Where <compound response header> is defined as



Where <common response header> is defined as

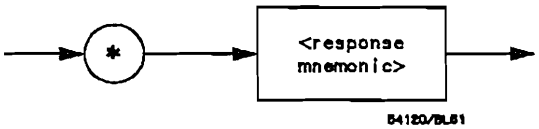
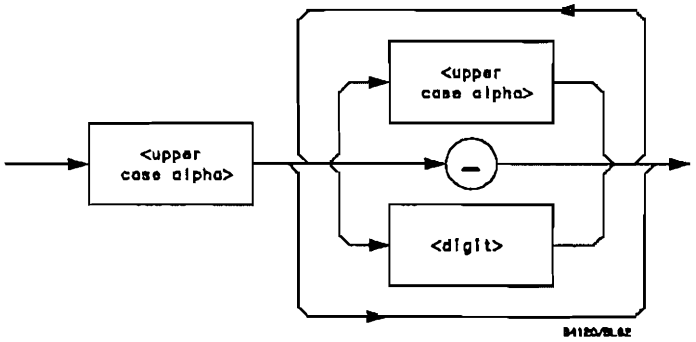


Figure 3-21. <response message unit>

Where <response mnemonic> is defined as



Where <uppercase alpha> is defined as a single ASCII encoded byte in the range 41—5A (65—90 decimal).

Where (_) represents an "underscore", a single ASCII encoded byte with the value 5F (95 decimal).

Figure 3-21. <response message unit> (continued)

<response data>. The <response data> element represents the various types of data which the instrument may return. These types include: <character response data>, <nr1 numeric response data> (integer), <nr3 numeric response data> (exponential), <string response data>, <definite length arbitrary block response data>, and <arbitrary ASCII response data>.

Figure 3-22. <character response data>

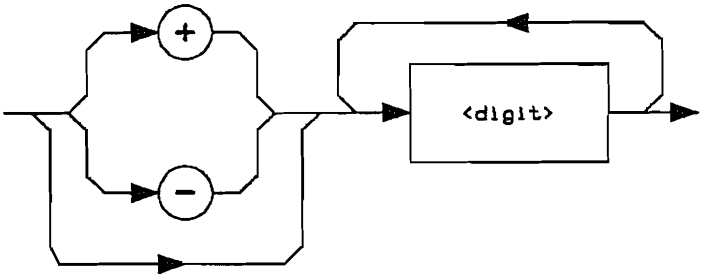


Figure 3-23. <nrl numeric response data>

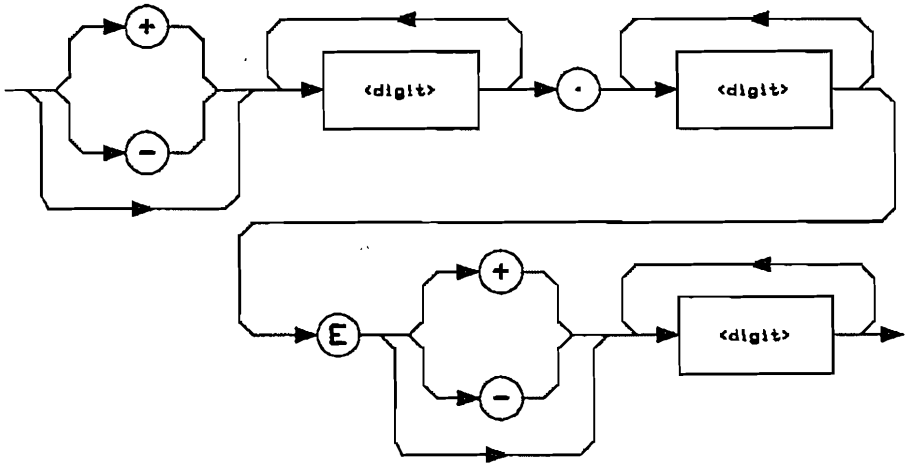


Figure 3-24. <nr3 numeric response data>

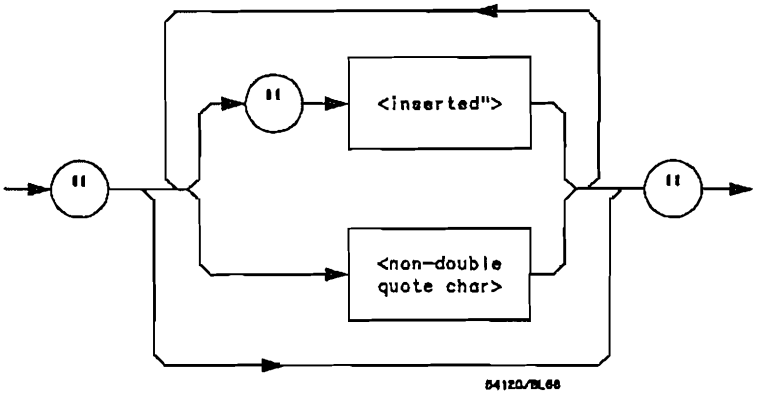


Figure 3-25. <string response data>

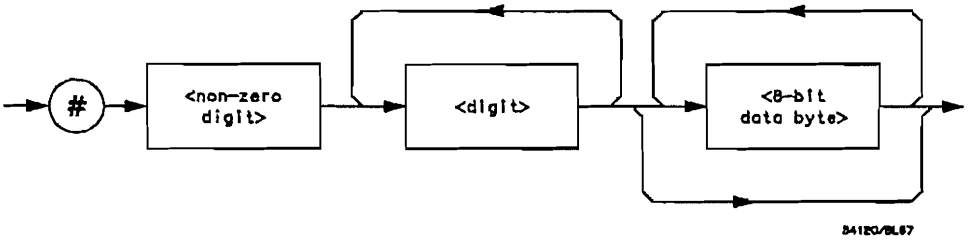
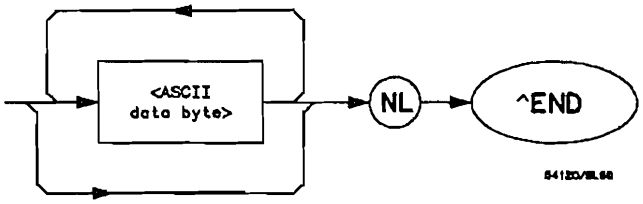


Figure 3-26. <definite length arbitrary block>



Where <ASCII data type> represents any ASCII-encoded byte except <NL> (0A, 10 decimal).

1. The END message provides an unambiguous termination to an element that contains arbitrary ASCII characters.

2. The IEEE 488.1 END message serves the dual function for terminating this element as well as terminating the <RESPONSE MESSAGE>. It is only sent once with the last byte of the indefinite block data. The NL is presented for consistency with the <RESPONSE MESSAGE TERMINATOR>.

Figure 3-27. <arbitrary ASCII response data>

<response data separator>. A comma separates multiple pieces of response data within a single <response message unit>.

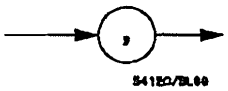


Figure 3-28. <response data separator>

<response header separator>. A space (ASCII decimal 32) delimits the response header, if returned, from the first or only piece of data.

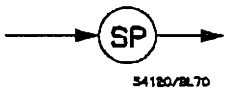


Figure 3-29. <response header separator>

<response message unit separator>. A semicolon delimits the <response message unit>s if multiple responses are returned.

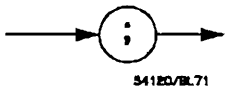


Figure 3-30. <response message unit separator>

<response message terminator>. A <response message terminator> (NL) terminates a complete <response message>. It should be read from the instrument along with the response itself.

Note

If you do not read the <response message terminator>, the Peak Power Analyzer will produce an interrupted error on the next message.

Common Commands

IEEE 488.2 defines a set of common commands. These commands perform functions which are common to any type of instrument. They can therefore be implemented in a standard way across a wide variety of instrumentation. All the common commands of IEEE 488.2 begin with an asterisk. There is one key difference between the IEEE 488.2 common commands and the rest of the commands found in the Peak Power Analyzer. The IEEE 488.2 common commands do not affect the parser's position within the command tree. More information about the command tree and tree traversal can be found in the Programming and Documentation Conventions chapter.

Table 3-3.
HP 8990A's IEEE 488.2 Common Commands

Command	Command Name
*CLS	Clear Status Command
*ESE	Event Status Enable Command
*ESE?	Event Status Enable Query
*ESR?	Event Status Register Query
*IDN?	Identification Query
*IST?	Individual Status Query
*LRN?	Learn Device Setup Query
*OPC	Operation Complete Command
*OPT?	Option Identification Query
*PRE	Parallel Poll Enable Register Enable Command
*PRE?	Parallel Poll Enable Register Enable Query
*OPC?	Operation Complete Query
*RCL	Recall Command
*RST	Reset Command
*SAV	Save Command
*SRE	Service Request Enable Command
*SRE?	Service Request Enable Query
*STB?	Read Status Byte Query
*TRG	Trigger Command
*TST?	Self-Test Query
*WAI	Wait-to-Continue Command

Status Reporting

The status reporting features which are available over the HP-IB include the serial and parallel polls. IEEE 488.2 defines data structures, commands, and common bit definitions for each. There are also instrument defined structures and bits. The bits in the status byte act as summary bits for the data structures residing behind them. In the case of queues, the summary bit is set if the queue is not empty. For registers, the summary bit is set if any enabled bit in the event register is set. The events are enabled via the corresponding event-enable-register. Events captured by an event register remain set until the register is read or cleared. Registers are read with their associated commands. For example, the "*CLS" command clears all event registers and all queues except the output queue. If "*CLS" is sent immediately following a <program message terminator>, the output queue will also be cleared.

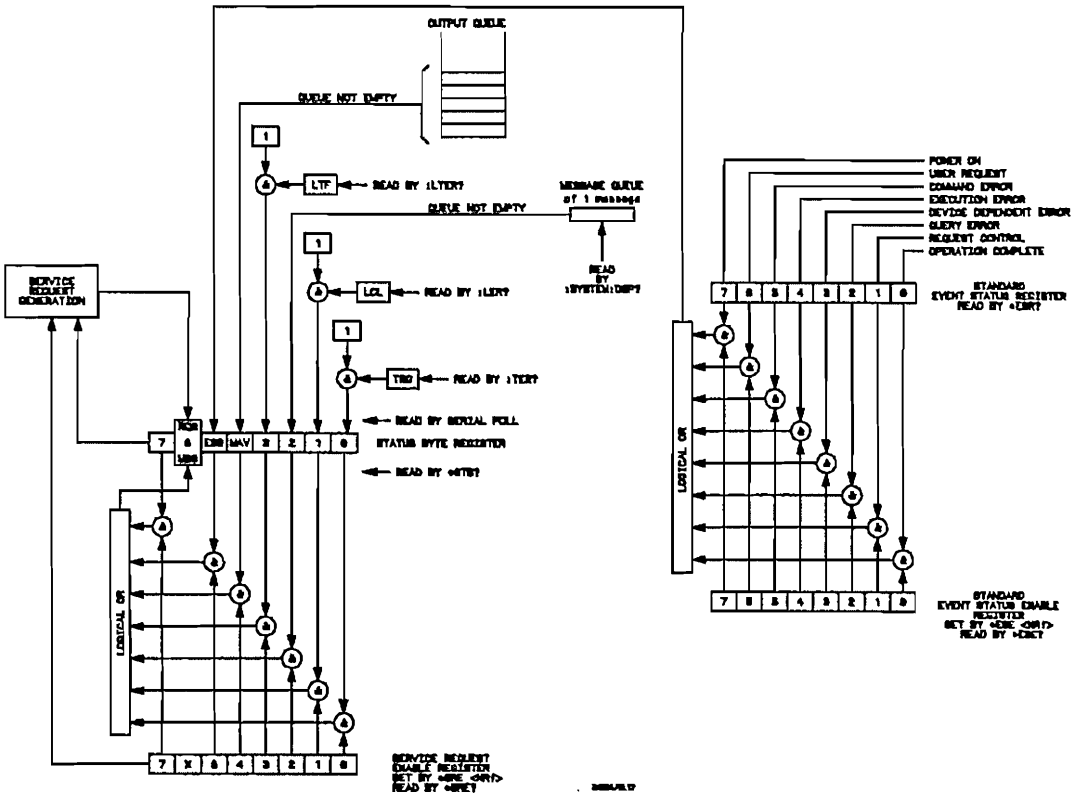


Figure 3-31. Status Reporting Data Structures

Bit Definitions

CME - command error. Indicates whether the parser detected an error.

DDE - device dependent error. Indicates whether the device was unable to complete an operation for device dependent reasons.

ESB - event status bit. Indicates if any of the conditions in the Standard Event Status Register are set and enabled.

EXE - execution error. Indicates whether a parameter was out of range, or inconsistent with current settings.

LCL - local. Indicates whether a remote to local transition has occurred.

LTF - limit test failure. Indicates whether a limit test failure has occurred.

MAV - message available. Indicates whether there is a response in the output queue.

MSG - Message. Indicates whether there is a message in the message queue.

MSS - master summary status. Indicates whether the device has a reason for requesting service. This bit is returned for the *STB? query.

OPC - operation complete. Indicates whether the device has completed all pending operations.

PON - power on. Always 0 in the Peak Power Analyzer.

QYE - query error. Indicates whether the protocol for queries has been violated.

RQC - request control. Indicates whether the device is requesting control. The Peak Power Analyzer will never request control.

RQS - request service. Indicates if the device is requesting service. This bit is returned during a serial poll. RQS will be set to 0 after being read via a serial poll (MSS is not reset by *STB?).

TRG - trigger. Indicates whether a trigger has been received.

URQ - user request. Indicates whether a front panel key has been pressed.

Key Features

A few of the most important features of Status Reporting are shown below.

Operation Complete. The IEEE 488.2 structure provides one technique which can be used to find out if any operation is finished. The *OPC command, when sent to the instrument after the operation of interest, will set the OPC bit in the Standard Event Status Register. If the OPC bit and the RQS bit have been enabled, a service request will be generated.

```
OUTPUT 707;"*SRE 32 ; *ESE 1"
    !Enables an OPC service
    !request.
OUTPUT 707;":DIG CHAN1 ; *OPC"
    !Initiates data acquisition,
    !and will generate an SRQ when
    !the acquisition is complete.
```

The Trigger Bit. The TRG bit indicates if the device has received a trigger. The TRG event register will stay set after receiving a trigger until it is cleared by reading it or by using the *CLS command. If your application needs to detect multiple triggers, the TRG event register must be cleared after each trigger.

If you are using the Service Request to interrupt a program or controller operation when the trigger bit is set, then you must clear the event register after each time it has been set.

```
OUTPUT 707;"*SRE 1"
    !Enables a trigger service request.
```

```
    !The next trigger will generate an SRQ.
OUTPUT 707;":TER?"
    !Queries the TRG event register, thus
    !clearing it.
ENTER 707;A
    !The next trigger can now generate an
    !SRQ
```

Status Byte. If the device is requesting service (RQS set), and the controller serial polls the device, the RQS bit is cleared. The MSS bit (read with *STB?) will not be cleared by reading it. With the exception of the RQS bit, the status byte is not cleared when read.

Serial Poll

The Peak Power Analyzer supports the IEEE 488.1 serial poll feature. When a serial poll of the instrument is requested, the RQS bit is returned on bit 6 of the status byte.

Using Serial Poll. This example will show how to use the service request by conducting a serial poll of all instruments on the bus. In this example, assume that there are two instruments on the bus; an Peak Power Analyzer at address 7 and a printer at address 1. These address assumptions are made throughout this manual. It is also assumed that we are operating on controller HP-IB interface select code 7.

The program command for serial poll using HP BASIC 5.0 is Stat=SPOLL(707). The address 707 is the address of the Peak Power Analyzer in this example. The command for checking the printer is Stat=SPOLL(701) because the address of that instrument is 01 on bus address 7. This command reads the contents of the HP-IB Status Register into the variable called Stat. At that time bit 6 of the variable Stat can be tested to see if it is set (bit 6=1).

The serial poll operation is conducted in the following manner.

1. Enable interrupts on the bus. This allows the controller to "see" the SRQ line.
2. If the SRQ line is high (some instrument is requesting service), check the instrument at address 1 to see if bit 6 of its status register is high.
3. Disable interrupts on the bus.
4. To check whether bit 6 of an instrument's status register is high, use the following command line: IF BIT (Stat, 6) then
5. If bit 6 of the instrument at address 1 is not high, then check the instrument at address 7 to see if bit 6 of its status register is high.
6. As soon as the instrument with status bit 6 high is found, check the rest of the status bits to determine what is required.

The SPOLL(707) command causes more to happen on the bus than simply reading the register. This command clears the bus, automatically addresses the talker and listener, sends SPE (serial poll enable) and SPD (serial poll disable) bus commands, and reads the data. For more information about serial poll, refer to your controller manual and programming language reference manuals.

After the serial poll is completed, the RQS bit in the Peak Power Analyzer's Status Byte Register will be reset if it was set. Once a bit in the Status Byte Register is set, it will remain set until the status is cleared with a *CLS command or the instrument is reset. If these bits are not reset, they cannot generate another SRQ.

Parallel Poll

Parallel poll is a controller initiated operation which is used to obtain information from several devices simultaneously. When a controller initiates a Parallel Poll, each device returns a Status Bit via one of the DIO data lines. Device DIO assignments are made by the controller using the PPC (Parallel Poll Configure) sequence. Devices respond either individually, each on a separate DIO line; collectively, on a single DIO line; or in any combination of these two ways. When responding collectively, the result is a logical AND (True High) or a logical OR (True Low) of the groups of the status bits.

Figure 3-32 shows the Parallel Poll Data Structure. The summary bit is sent in response to a Parallel Poll. This summary bit is the "ist" (Individual Status) local message.

The Parallel Poll Enable Register determines which events are summarized in the individual status local register. The *PRE (Parallel Poll Register Enable) command is used to write to the enable register and the *PRE? query is used to read the register. The *IST? query can be used to read the "ist" without doing a parallel poll.

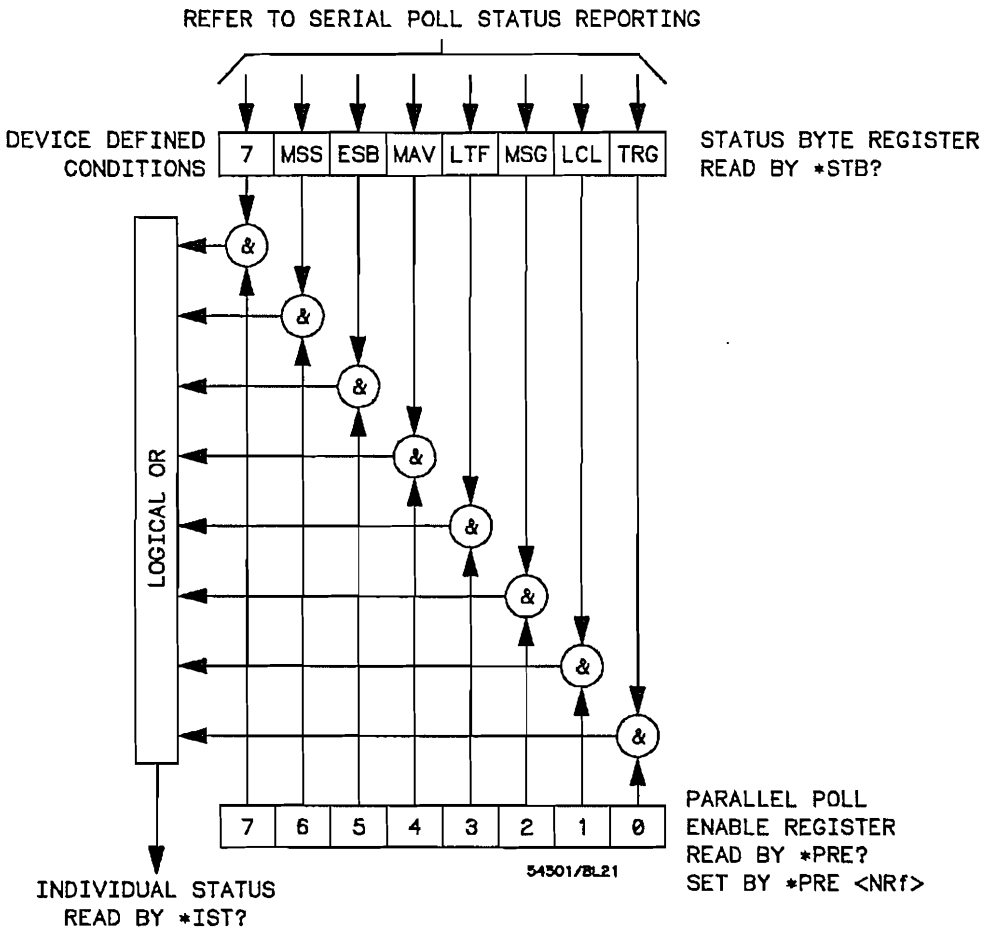


Figure 3-32. Parallel Poll Data Structure

Polling HP-IB Devices.

Parallel Poll is the fastest means of gathering device status when several devices are connected to the bus. Each device (with this capability) can be programmed to respond with one bit of status when parallel polled. This makes it possible to obtain the status of several devices in one operation. If a device responds affirmatively to a parallel poll, more information about its specific status can be obtained by conducting a serial poll of the device.

Configuring Parallel Poll Responses

Certain devices, including the Peak Power Analyzer, can be remotely programmed by a controller to respond to a parallel poll. A device which is currently programmed for a parallel poll responds to the poll by placing its current status on one of the bus data lines. The response and the data-bit number can be programmed by the PPC (Parallel Poll Configure) statement. Multiple listeners cannot be specified in this statement. If more than one device is to respond on a single bit, each device must be configured with a separate PPC statement.

Example:

```
ASSIGN @Device to 707
PPOLL CONFIGURE @Device;Mask
```

The value of Mask (any numeric expression can be specified) is first rounded and then used to configure the device's parallel response. The least significant 3 bits (bits 0 through 2) of the expression are used to determine which data line the device is to respond on (place its status on). Bit 3 specifies the "true" state of the parallel poll response bit of the device. A value of 0 implies that the device's response is 0 when its status bit message is true.

Example:

The following statement configures the device at address 07 on the interface select 7 to respond by placing a 0 on DIO4 when its status response is "true."

```
PPOLL CONFIGURE 707;4
```

Conducting a Parallel Poll

The PPOLL (Parallel Poll) function returns a single byte containing up to 8 status bit messages for all devices on the bus capable of responding to the poll. Each bit returned by the function corresponds to the status bit of the devices(s) configured to respond to the parallel poll (one or more devices can respond on a single line). The PPOLL function can only be executed by the controller. It is initiated by the simultaneous assertion of the ATN and EOI on the HP-IB interface.

Example:

```
Response = PPOLL(7)
```

Disabling Parallel Poll Responses

The PPU (Parallel Poll Unconfigure) statement gives the controller the capability of disabling the parallel poll response of one or more devices on the bus.

Examples:

The following statement disables device five only:

```
PPOLL UNCONFIGURE 705
```

This statement disables all devices on interface select code eight from responding to a parallel poll:

```
PPOLL UNCONFIGURE 8
```

If no primary address is specified, all bus devices are disabled from responding to a parallel poll. If a primary address is specified, only the specified devices (which have the parallel poll configure capability) are disabled.

HP-IB Commands

The following paragraphs describe what the actual HP-IB commands do to perform the functions of the BASIC commands shown in the previous examples.

Parallel Poll Unconfigure Command. The parallel poll unconfigure command (PPU) resets all parallel poll devices to the idle state (unable to respond to a parallel poll).

Parallel Poll Configure Command. The parallel poll configure command (PPC) causes the addressed listener to be configured according to the parallel poll enable secondary command PPE.

Parallel Poll Enable Command. The parallel poll enable secondary command (PPE) configures the devices which have received the PPC command to respond to a parallel poll on a particular HP-IB DIO line with a particular level.

Parallel Poll Disable Command. The Parallel Poll disable secondary command (PPD) disables the devices which have received the PPC command from responding to parallel poll.

Table 3-4. Parallel Poll Commands

Command	Mnemonic	Decimal Code	ASCII/ISO Character
Parallel Poll Unconfigure (Multiline Command)	PPU	21	NAK
Parallel Poll Configure (Secondary Command)	PPC	05	ENQ
Parallel Poll Enable (Secondary Command)	PPE	96–111	I–O
Parallel Poll Disable (Secondary Command)	PPD	112	P

4

Programming and Documentation Conventions

Introduction

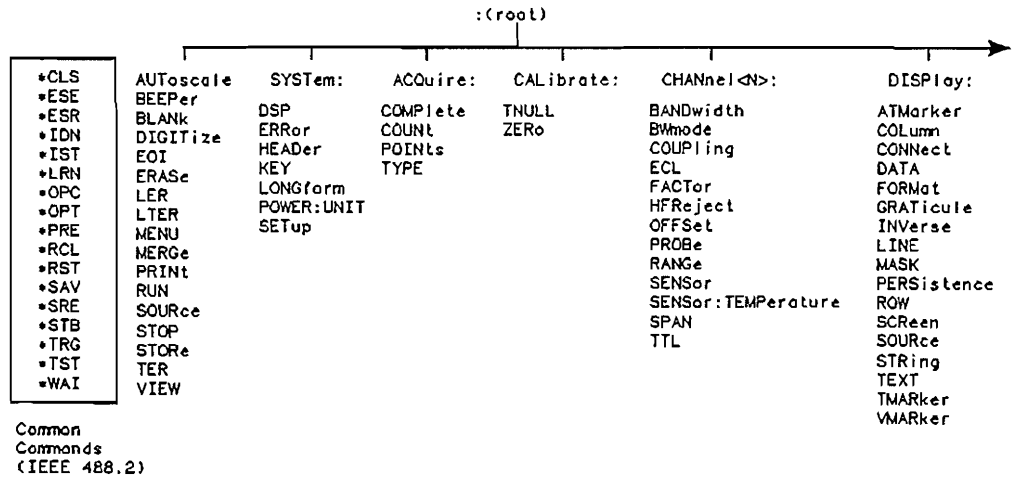
This chapter covers conventions which are used in programming the HP 8990A Peak Power Analyzer, as well as conventions used in the remainder of this manual. Included is a detailed description of the command tree and command tree traversal. Figures have been included which show how the HP-IB commands relate to the front panel and menus. For more information on command syntax, refer to the chapter "Message Communication and System Functions."

Truncation Rules

The truncation rules for the mnemonics used in headers and alpha arguments are as follows:

- If the length of the keyword is exactly four characters, the keyword is the mnemonic.
- The mnemonic is the first four characters of the keyword, unless the fourth character of the keyword is a vowel. The mnemonic is then the first three characters of the keyword.

Exceptions to these rules are described in the appropriate subsystem. Some examples of how the truncation rule is applied to various commands are shown in Table 4-1.



NOTE: THIS INSTRUMENT CONTAINS FOUR IDENTICAL CHANNEL SUBSYSTEMS AND TWO IDENTICAL FUNCTION SUBSYSTEMS. THE 'N' IN THE CHANNEL HEADER MUST BE 1 THROUGH 4, AND THE 'N' IN THE FUNCTION HEADER MUST BE 1 OR 2.

Figure 4-1. The HP 8990A Command Tree

Table 4-1. Mnemonic Truncation

Longform	Shortform
RANGE	RANG
PATTERN	PATT
TIME	TIME
DELAY	DEL

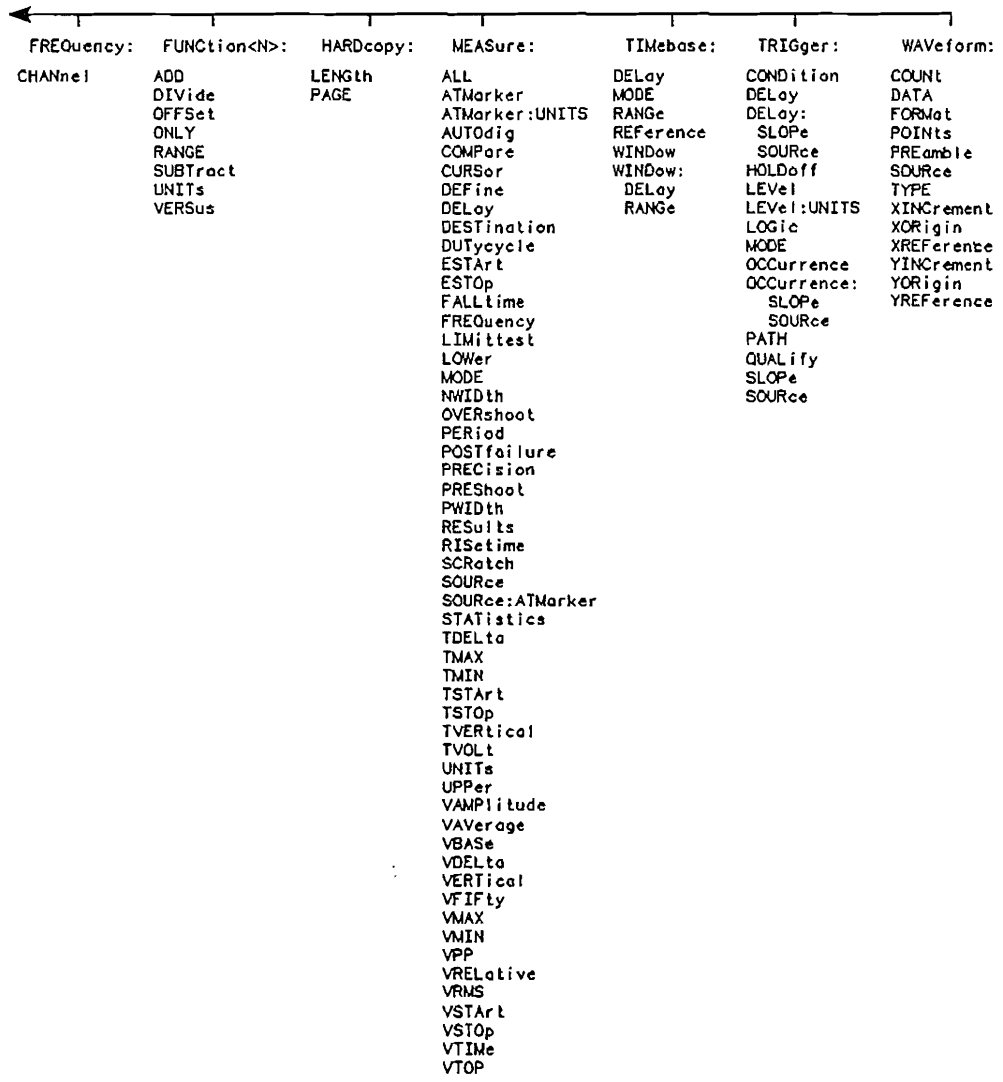


Figure 4-1. The HP 8990A Command Tree (continued)

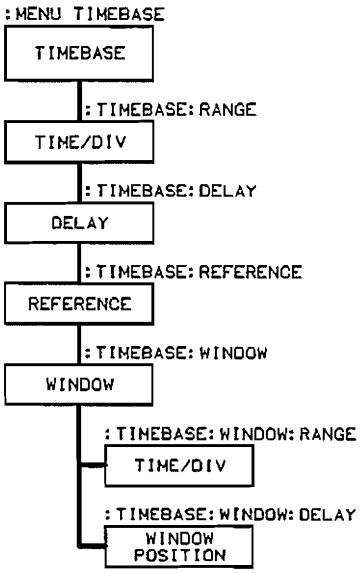


Figure 4-2. Front Panel Menus and Related HP-IB Commands

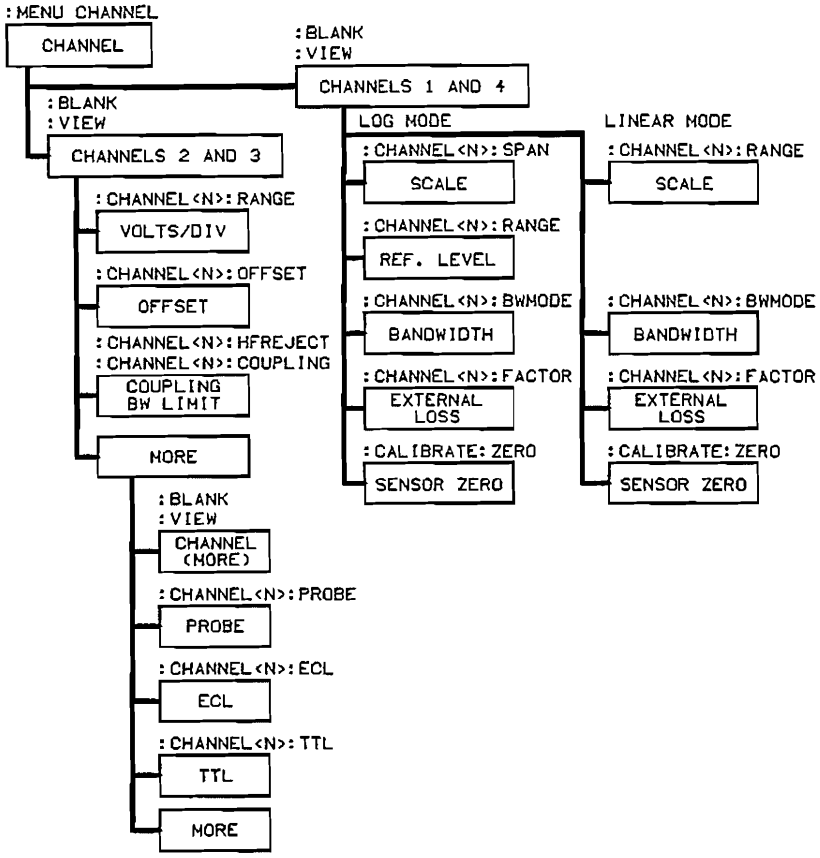


Figure 4-2. Front Panel Menus and Related HP-IB Commands (continued)

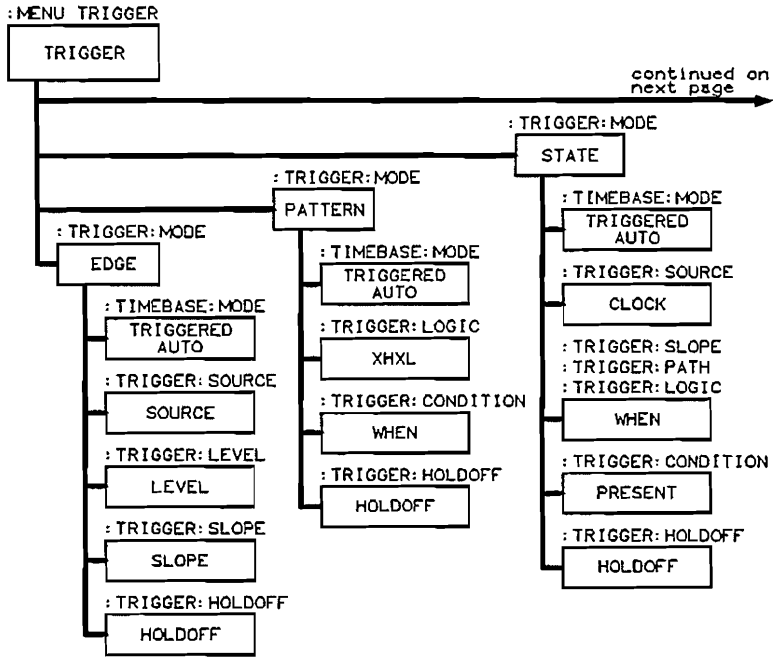


Figure 4-2. Front Panel Menus and Related HP-IB Commands (continued)

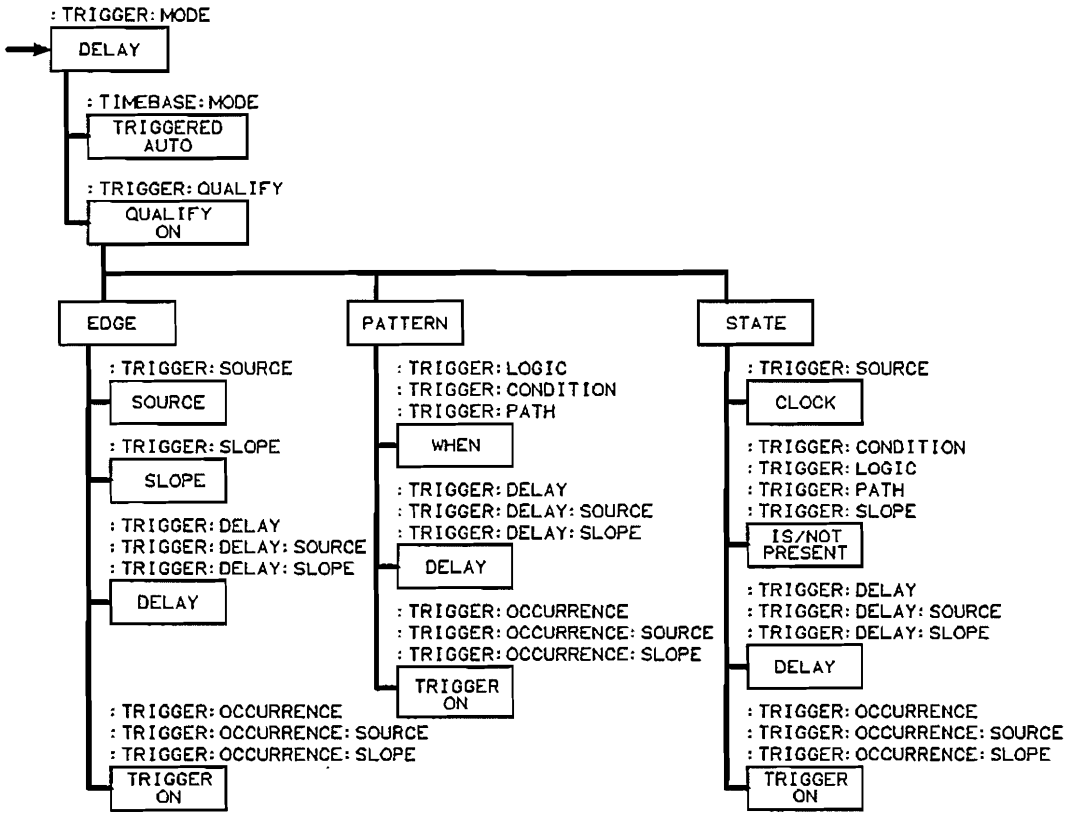


Figure 4-2. Front Panel Menus and Related HP-IB Commands (continued)

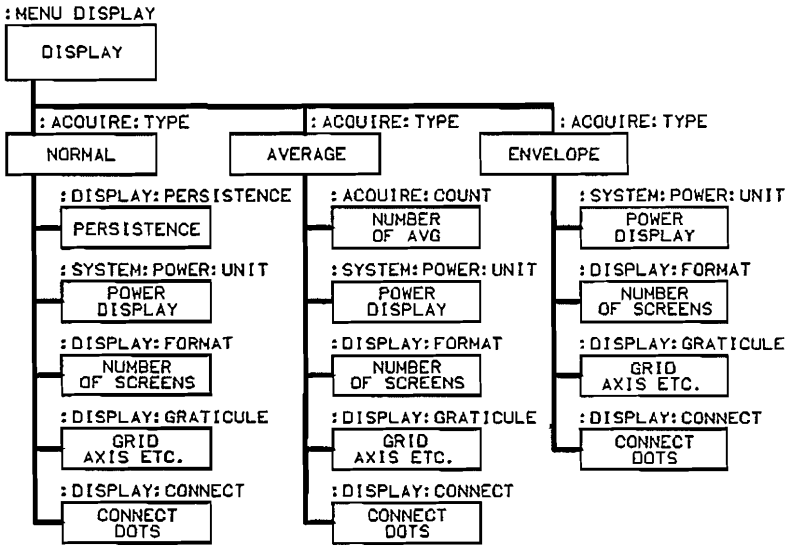


Figure 4-2. Front Panel Menus and Related HP-IB Commands (continued)

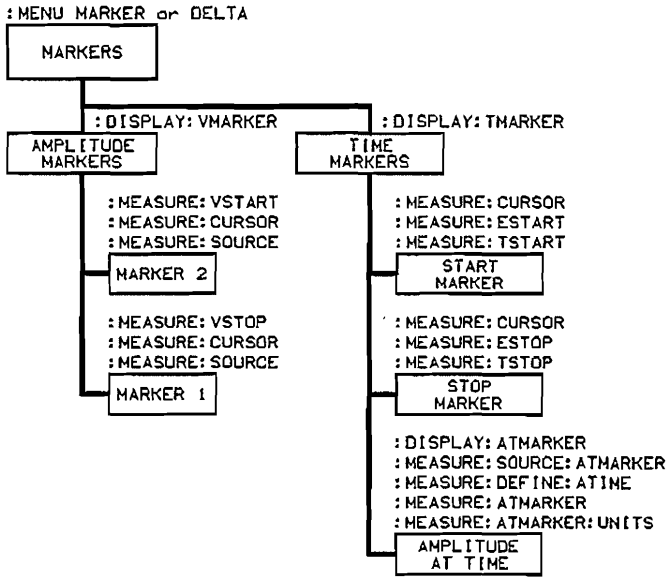


Figure 4-2. Front Panel Menus and Related HP-IB Commands (continued)

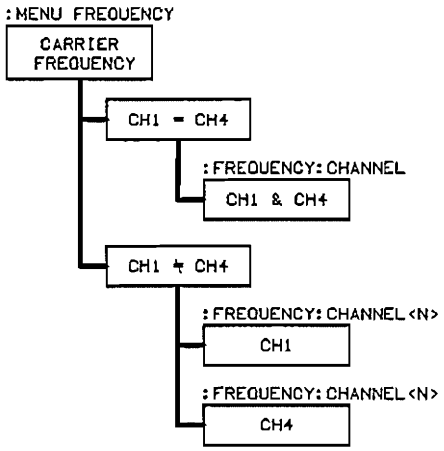


Figure 4-2. Front Panel Menus and Related HP-IB Commands (continued)

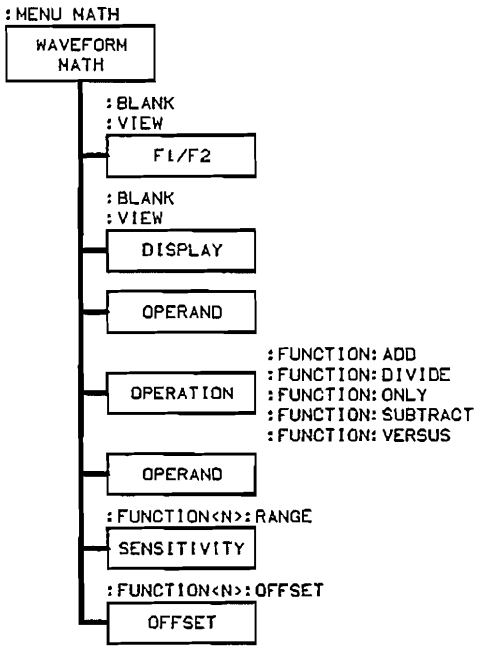


Figure 4-2. Front Panel Menus and Related HP-IB Commands (continued)

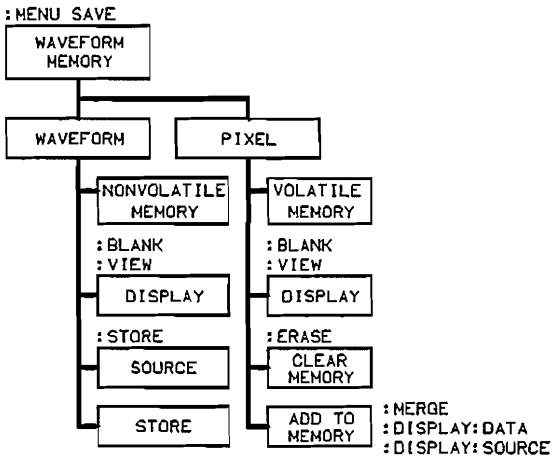


Figure 4-2. Front Panel Menus and Related HP-IB Commands (continued)

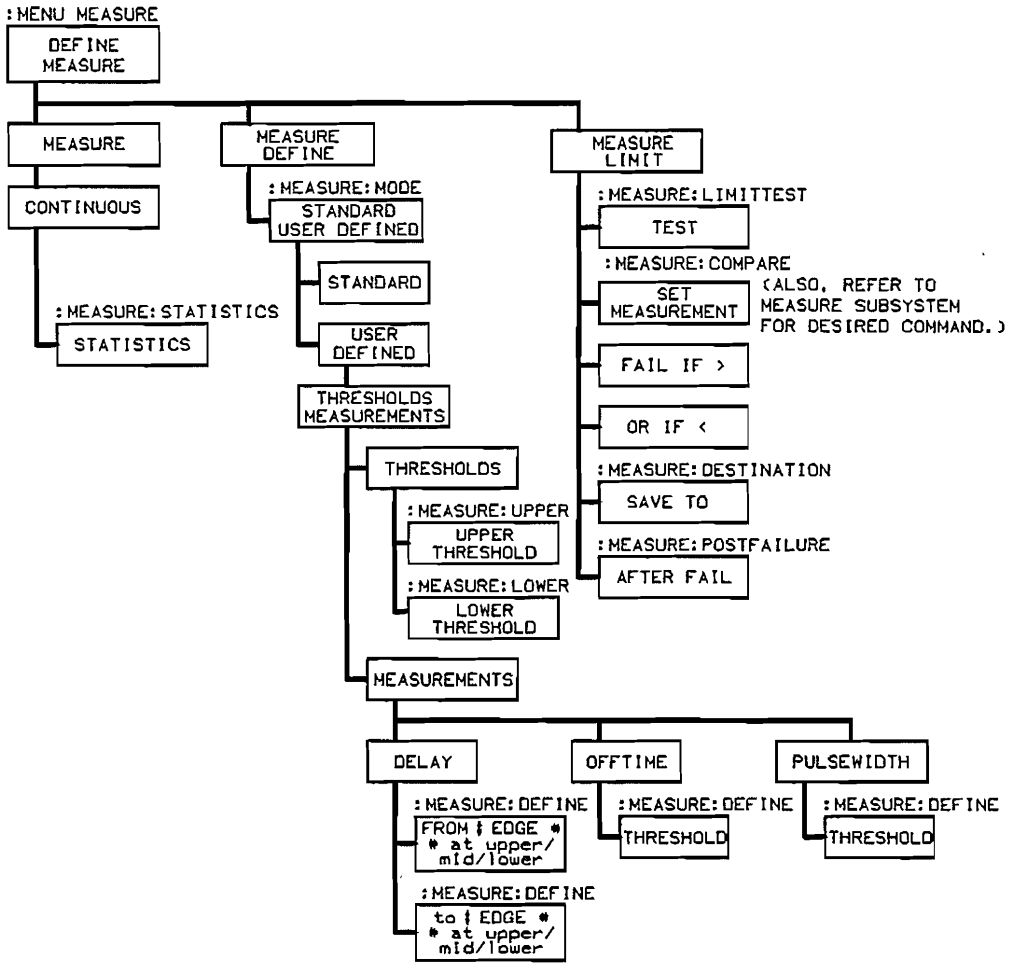


Figure 4-2. Front Panel Menus and Related HP-IB Commands (continued)

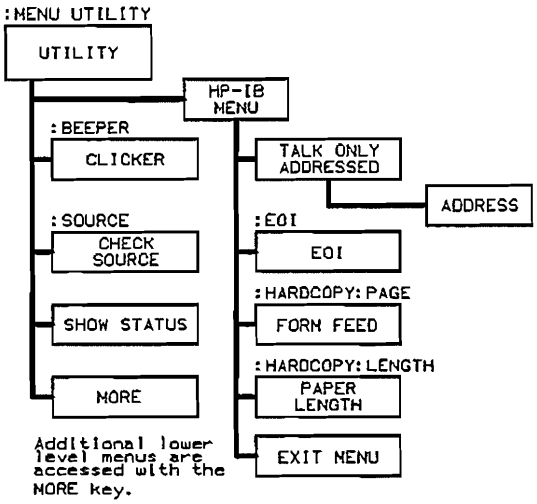


Figure 4-2. Front Panel Menus and Related HP-IB Commands (continued)

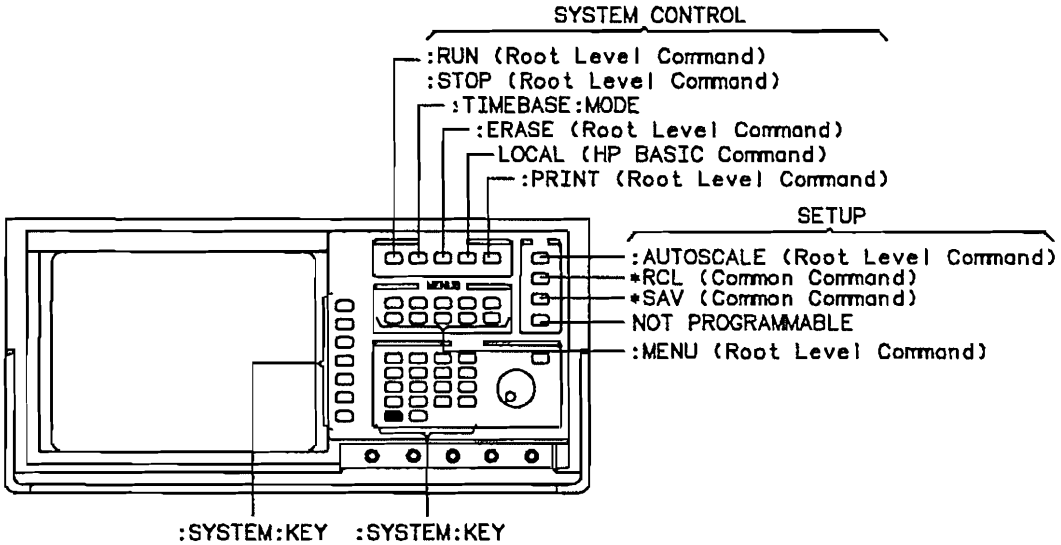


Figure 4-3. Front Panel Keys and Related HP-IB Commands

The Command Tree

The command tree (Figure 4-1) shows all commands in the Peak Power Analyzer and the relationship of the commands to each other. The IEEE 488.2 common commands are not listed as part of the command tree since they are independent of the position of the parser within the tree. After a <NL> (linefeed - ASCII decimal 10) has been sent to the Peak Power Analyzer, the parser will be reset to the "root" of the command tree.

Command Types

The commands for this instrument can be placed into three types. The three types are:

Common commands. Common commands are independent of the tree and do not change the position of the parser within the tree. These differ from root level commands in that root level commands place the parser back at the root.

Example: ***RST.**

Root Level commands. The root level commands reside at the root of the command tree. These commands are always parsable if they occur at the beginning of a program message or are preceded by a colon.

Example: **:AUTOSCALE**

Subsystem commands. Subsystem commands are grouped together under a common node of the tree, such as the TIMEBASE commands under the TIMEBASE node.

Tree Traversal Rules

Command headers are created by traversing down the command tree. A legal command header from the command tree in Figure 4-1 would be ":CHANNEL1:RANGE". This is referred to as a compound header. A compound header is a header made of two or more mnemonics separated by colons. The

mnemonic created contains no embedded spaces. The following rules apply to traversing the tree:

- A leading colon or a <program message terminator> (either a <NL> or EOI true on the last byte) places the parser at the root of the command tree. A leading colon is a colon that is the first character of a program header.
- Executing a subsystem command places you in that subsystem (until a leading colon or a <program message terminator> is found). In the Command Tree, Figure 4-1, use the last mnemonic in the compound header as a reference point (for example, RANGE). Then, find the last colon above that mnemonic (CHANNEL1:), and that is where the parser will be. Any command below that point can be sent within the current program message without sending the mnemonic(s) which appear above them (OFFSET).

Examples

The OUTPUT statements are written using HP BASIC 5.0 on a HP 9000 Series 200/300 Controller. After the HP-IB talk and listen address sequence, the quoted string is placed on the bus, followed by a carriage return and linefeed (CRLF).

Example 1:

OUTPUT 707;":CHAN1:RANGE 0 dBm;:SPAN 40 dB"

Comments:

The colon between CHANNEL1 and RANGE is necessary, CHANNEL1:RANGE is a compound command. The semicolon between the RANGE command and the SPAN command is the required <program message unit separator>. The SPAN command does not need CHANNEL1 preceding it, since

the CHANNEL1:RANGE command set the parser to the CHANNEL1 node in the tree.

Example 2:

```
OUTPUT 707;":TIM:REF CENTER;:DELAY 0.00001"
```

■ or

```
OUTPUT 707;":TIMEBASE:REFERENCE CENTER"
OUTPUT 707;":TIMEBASE:DELAY 0.00001"
```

Comments:

In the first line of example 2, the "subsystem selector" is implied for the DELAY command in the compound command.

The DELAY command must be in the same program message as the REFERENCE command, because the <program message terminator> will place the parser back at the root of the command tree.

A second way to send these commands is by placing "TIMEBASE:" before the DELAY command as shown in the second part of example 2.

Example 3:

```
OUTPUT 707;":TIM:REF CENT;:CHAN1:SPAN 40
DB"
```

Comments:

The leading colon before CHAN1 tells the parser to go back to the root of the command tree. The parser can then associate the CHAN1:SPAN command with the channel subsystem.

Infinity Representation

The representation of infinity is 9.99999E+37. This is also the value returned when a measurement cannot be made. If the result of a power measurement is too small to be represented as a dBm value, the value returned for the measurement is −9.99999E+37. When −9.99999E+37 is returned, −99 dBm is the result output to the front panel.

Sequential and Overlapped Commands.

IEEE 488.2 makes the distinction between sequential and overlapped commands. Sequential commands finish their task before the execution of the next command starts. Overlapped commands run concurrently, and, therefore, the command following an overlapped command may be started before the overlapped command is completed. All the commands of the Peak Power Analyzer are **sequential.**

Response Generation

IEEE 488.2 defines two times when query responses may be buffered. The first is when the query is parsed by the instrument, the second is when the controller addresses the instrument to talk so that it may read the response. The Peak Power Analyzer will buffer responses to a query when the query is parsed.

Notation Conventions and Definitions

The following conventions are used in this manual in descriptions of remote (HP-IB) operation:

< > (angular brackets) are used to enclose words or characters that symbolize a program code parameter or an HP-IB command.

::= means "is defined as". For example, <A> ::= <B> indicates that <A> can be replaced by <B> in any statement containing <A>.

| (vertical bar) indicates a choice of one element from a list. For example, <A> | <B> indicates <A> or <B> but not both.

... (an ellipsis) is used to indicate that the preceding element may be repeated one or more times.

[ ] (square brackets) indicate that the enclosed items are optional.

{ } (braces) indicate that one and only one of the enclosed elements must be selected.

The following definitions are used:

d ::= A single ASCII numeric character, 0-9.

n ::= A single ASCII non-zero, numeric character, 1-9.

<NL> ::= Newline or Linefeed (ASCII decimal 10).

<sp> ::= <white space>

<white space> ::= 0 through 32 (decimal) except linefeed (decimal 10). <white space> is usually 32.

Syntax Diagrams

At the beginning of each of the following chapters are syntax diagrams showing the proper syntax for each command. All characters contained in a circle or oblong are literals and must be entered exactly as shown. Words and phrases contained in rectangles are names of items used with the command and are described in the accompanying text of each command. Each line can only be entered from one direction as indicated by the arrow on the entry line. Any combination of commands and arguments that can be generated by following the lines in the proper direction is syntactically correct. An argument is optional if there is a path around it. Where there is a rectangle which contains the word "space", a white space character must be entered. White space is optional in many other places.

Command Structure

The Peak Power Analyzer programming commands are divided into three types: common commands, root level commands, and subsystem commands. A programming command tree is shown in figure 4-1.

Common Commands

The common commands are the commands defined by IEEE 488.2. These commands control some functions that are common to all IEEE 488.2 instruments. Sending the common commands does not take the Peak Power Analyzer parser out of a selected subsystem.

Root Level Commands

The root level commands control many of the basic functions of the Peak Power Analyzer.

Subsystem Commands

There are several subsystems in the Peak Power Analyzer. Only one subsystem may be programmed or queried at any given time. At power on, the command parser is set to the root of the command tree; therefore, no subsystem is active.

Note

When a program message terminator or a leading colon (:) is sent in a program unit, the command parser is returned to the root of the command tree

The 12 subsystems in the Peak Power Analyzer are listed below:

System - controls some basic functions of the Analyzer.

Acquire - sets the parameters for acquiring and storing data.

Calibrate - sets the time nulls (channel-to-channel skew).

Channel - controls all amplitude measurement Analyzer functions.

Display - controls how waveforms, amplitude and time markers, graticule, and text are displayed and written on the screen.

Function - controls the waveform math functions of the Analyzer.

Hardcopy - controls the parameters used during the plotting or printing of waveforms.

Measure - selects the automatic measurements to be made.

Timebase - controls all X-axis Analyzer functions.

Trigger - controls the trigger modes and parameters for each trigger mode.

Waveform - provides access to waveform data, including active data from channels and functions as well as static data from waveform memories.

Frequency - sets the carrier frequency that is at the sensor inputs.

Program Examples

The program examples given for each command in the following chapters and appendices were written on an HP 9000 Series 200/300 controller using HP BASIC 5.0 language. The programs always assume the Peak Power Analyzer is at address 707. If a printer is used, it is always assumed to be at address 701.

In these examples, special attention should be paid to the ways in which the command/query can be sent. The way that the Peak Power Analyzer is set up to respond to a command/query has no bearing on how you send the command/query. That is, the command/query can be sent using the longform or shortform if one exists for that command. You can send the command/query using upper case (capital) letters or lower case (small) letters; both work the same way. Also, the data can be sent using almost any form you wish. If you were sending a channel 1 range value of 100 m Watts, that value can sent using a decimal (.1 W), an exponent (1e-01 or 1.0E-01), or a suffix (100 mw or 100 MW).

As an example, set channel 1 range to 100 mWatts by sending one of the following:

- commands in longform and using the decimal format and a suffix.

  OUTPUT 707;":CHANNEL1:RANGE .1 W"

- commands in shortform and using an exponential format.

  OUTPUT 707;":CHAN1:RANG 1E-01 W"

- commands using lower case letters, shortforms, and a suffix.

  OUTPUT 707;":chan1:rang 100 mW"

Note

In these examples, the colon as the first character of the command is optional. The space between RANGE and the argument is required.

If you want to observe the headers for the queries, you must bring the returned data into a string variable. Generally, you should dimension all BASIC string variables before reading the data. If the string variable will contain sixteen or more characters, it must be dimensioned. The following is an example of a dimensioned string variable:

```
DIM Freq$[100]
OUTPUT 707;":SYSTEM:HEADER ON"
OUTPUT 707;":SYSTEM:LONGFORM ON"
OUTPUT 707;":MEASURE:FREQUENCY?"
ENTER 707;Freq$
PRINT Freq$
END
```

If you do not need to see the headers and a numeric value is returned from the Peak Power Analyzer, then you should use a numeric variable. In this case the headers should be turned off. The following is an example using a numeric variable:

```
OUTPUT 707;":SYSTEM:HEADER OFF"
OUTPUT 707;":SYSTEM:LONGFORM OFF"
OUTPUT 707;":MEASURE:FREQUENCY?"
ENTER 707;Freq
PRINT Freq
END
```

Command Set Organization

The command set for the Peak Power Analyzer is divided into 14 separate groups: Common commands, root level commands and 12 sets of subsystem commands. Each of the 14 groups of commands is described in the following chapters. Each of the chapters contain a brief description of the subsystem, a set of syntax diagrams for those commands, and finally, the commands for that subsystem in alphabetic order. The commands are shown in the longform and shortform using upper and lowercase letters. As an example, AUToscale indicates that the longform of the command is AUTOSCALE and the shortform of the command is AUT. Each of the commands contains a description of the command and its arguments, the command syntax, and a programming example.

Table 4-2. Alphabetic Command Cross-Reference

Command	Where Used	Command	Where Used
ADD	FUNCtion Subsystem	DELay	MEASure Subsystem
ALL	MEASure Subsystem	DELay	TIMebase Subsystem
ATMarker	DISPlay Subsystem	DELay	TRIGger Subsystem
ATMarker	MEASure Subsystem	DELay:SLOPe	TRIGger Subsystem
ATMarker:UNITS	MEASure Subsystem	DELay:SOURce	TRIGger Subsystem
AUTodig	MEASure Subsystem	DESTination	MEASure Subsystem
AUToscale	Root Level Command	DIGitize	Root Level Command
BANDwidth	CHANnel Subsystem	DIVide	FUNCtion Subsystem
BEEPer	Root Level Command	DSP	SYSTem Subsystem
BLANk	Root Level Command	DUTycycle	MEASure Subsystem
BWmode	CHANnel Subsystem	ECL	CHANnel Subsystem
CHANnel	FREQuency Subsystem	EOI	Root Level Command
*CLS	Common Command	ERASe	Root Level Command
COLumn	DISPlay Subsystem	ERRor	SYSTem Subsystem
COMPare	MEASure Subsystem	*ESE	Common Command
COMPlete	ACQuire Subsystem	*ESR	Common Command
CONDition	TRIGger Subsystem	ESTArt	MEASure Subsystem
CONNect	DISPlay Subsystem	ESTOp	MEASure Subsystem
COUNt	ACQuire Subsystem	FACTor	CHANnel Subsystem
COUNt	WAVeform Subsystem	FALLtime	MEASure Subsystem
COUPling	CHANnel Subsystem	FORMat	DISPlay Subsystem
CURSor	MEASure Subsystem	FORMat	WAVeform Subsystem
DATA	DISPlay Subsystem	FREQuency	MEASure Subsystem
DATA	WAVeform Subsystem	GRATicule	DISPlay Subsystem
DEFine	MEASure Subsystem	HEADer	SYSTem Subsystem

Table 4-2. Alphabetic Command Cross-Reference (continued)

Command	Where Used	Command	Where Used
HFReject	CHANnel Subsystem	OCCurrence:SLOPe	TRIGger Subsystem
HOLDoff	TRIGger Subsystem	OCCurrence:SOURce	TRIGger Subsystem
*IDN	Common Command	OFFSet	CHANnel Subsystem
INVerse	DISPlay Subsystem	OFFSet	FUNCtion Subsystem
*IST	Common Command	ONLY	FUNCtion Subsystem
KEY	SYSTem Subsystem	*OPC	Common Command
LENGth	HARDcopy Subsystem	*OPT	Common Command
LER	Root Level Command	OVERshoot	MEASure Subsystem
LEVel	TRIGger Subsystem	PAGE	HARDcopy Subsystem
LEVel:UNITs	TRIGger Subsystem	PATH	TRIGger Subsystem
LIMittest	MEASure Subsystem	PERiod	MEASure Subsystem
LINE	DISPlay Subsystem	PERSistence	DISPlay Subsystem
LOGic	TRIGger Subsystem	POINts	ACQuire Subsystem
LONGform	SYSTem Subsystem	POINts	WAVeform Subsystem
LOWer	MEASure Subsystem	POSTfailure	MEASure Subsystem
*LRN	Common Command	POWER:UNIT	SYSTem Subsystem
LTER	Root Level Command	*PRE	Common Command
MASK	DISPlay Subsystem	PREamble	WAVeform Subsystem
MENU	Root Level Command	PRECision	MEASure Subsystem
MERGe	Root Level Command	PREShoot	MEASure Subsystem
MODE	MEASure Subsystem	PRINt	Root Level Command
MODE	TIMebase Subsystem	PROBe	CHANnel Subsystem
MODE	TRIGger Subsystem	PWIDth	MEASure Subsystem
NWIDth	MEASure Subsystem	QUALify	TRIGger Subsystem
OCCurrence	TRIGger Subsystem	RANGe	CHANnel Subsystem

Table 4-2. Alphabetic Command Cross-Reference (continued)

Command	Where Used	Command	Where Used
RANGe	FUNCtion Subsystem	*STB	Common Command
RANGe	TIMebase Subsystem	STOP	Root Level Command
*RCL	Common Command	STORe	Root Level Command
REFerence	TIMebase Subsystem	STRing	DISPlay Subsystem
RESults	MEASure Subsystem	SUBTract	FUNCtion Subsystem
RISetime	MEASure Subsystem	TDELta	MEASure Subsystem
ROW	DISPlay Subsystem	TER	Root Level Command
*RST	Common Command	TEXT	DISPlay Subsystem
RUN	Root Level Command	TMARker	DISPlay Subsystem
*SAV	Common Command	TMAX	MEASure Subsystem
SCRatch	MEASure Subsystem	TMIN	MEASure Subsystem
SCReen	DISPlay Subsystem	TNULl	CALibrate Subsystem
SENSor	CHANnel Subsystem	*TRG	Common Command
SENSor:TEMPerature	CHANnel Subsystem	*TST	Common Command
SETup	SYSTem Subsystem	TSTArt	MEASure Subsystem
SLOPe	TRIGger Subsystem	TSTOp	MEASure Subsystem
SOURce	DISPlay Subsystem	TTL	CHANnel Subsystem
SOURce	MEASure Subsystem	TVERtical	MEASure Subsystem
SOURce	Root Level Command	TVOLt	MEASure Subsystem
SOURce	TRIGger Subsystem	TYPE	ACQuire Subsystem
SOURce	WAVeform Subsystem	TYPE	WAVeform Subsystem
SOURce:ATMarker	MEASure Subsystem	UNITs	FUNCtion Subsystem
SPAN	CHANnel Subsystem	UNITs	MEASure Subsystem
*SRE	Common Command	UPPer	MEASure Subsystem
STATistics	MEASure Subsystem	VAMPlitude	MEASure Subsystem

Table 4-2.
Alphabetic Command Cross-Reference (continued)

Command	Where Used	Command	Where Used
VAVerage	MEASure Subsystem	VSTOp	MEASure Subsystem
VBASe	MEASure Subsystem	VTIMe	MEASure Subsystem
VDELta	MEASure Subsystem	VTOP	MEASure Subsystem
VERSus	FUNCtion Subsystem	*WAI	Common Command
VERTical	MEASure Subsystem	WINDow	TIMebase Subsystem
VFIFty	MEASure Subsystem	WINDow:DELay	TIMebase Subsystem
VIEW	Root Level Command	WINDow:RANGe	TIMebase Subsystem
VMARker	DISPlay Subsystem	XINCrement	WAVeform Subsystem
VMAX	MEASure Subsystem	XORigin	WAVeform Subsystem
VMIN	MEASure Subsystem	XREFerence	WAVeform Subsystem
VPP	MEASure Subsystem	YINCrement	WAVeform Subsystem
VRELative	MEASure Subsystem	YORigin	WAVeform Subsystem
VRMS	MEASure Subsystem	YREFerence	WAVeform Subsystem
VSTArt	MEASure Subsystem	ZERo	CALibrate Subsystem

5

Common Commands

Introduction

The common commands are defined by the IEEE 488.2 standard. These commands will be common to all instruments that comply with this standard. They control some of the basic instrument functions:

- Instrument identification and reset
- Reading the learn (instrument setup) string
- Status reading and clearing
- Receiving and processing of commands and queries by the instrument.

Common commands can be received and processed by the Peak Power Analyzer whether they are sent over the HP-IB as separate program messages or within other program messages. If an instrument subsystem has been selected, and a common command is received by the instrument, the instrument will remain in the current subsystem for the rest of that message or until a leading colon is seen. For example, if the program message

```
OUTPUT 707;":ACQUIRE:COUNT 1024;*CLS;
:TYPE AVERAGE"
```

is received by the instrument, the instrument will set the acquire count, clear the status information, and specify the acquisition as average. This would not be the case if some other type of command were received within the program message. For example, the program message

```
OUTPUT 707;":ACQUIRE:COUNT 1024;:AUTOSCALE;
:ACQUIRE:TYPE AVERAGE"
```

would set the acquire count, complete the autoscale, and then set the acquire type. In this example :ACQUIRE is sent again in order to reenter the acquire subsystem and set the type.

Refer to figure 5-1 for the common command syntax diagram.

Note

Each of the status registers mentioned in this chapter has an enable (mask) register. By setting the bits in the enable register you can select the status information you wish to use. For a complete discussion of how to read the status registers and how to use the status information available from the Peak Power Analyzer, refer to chapter 3.

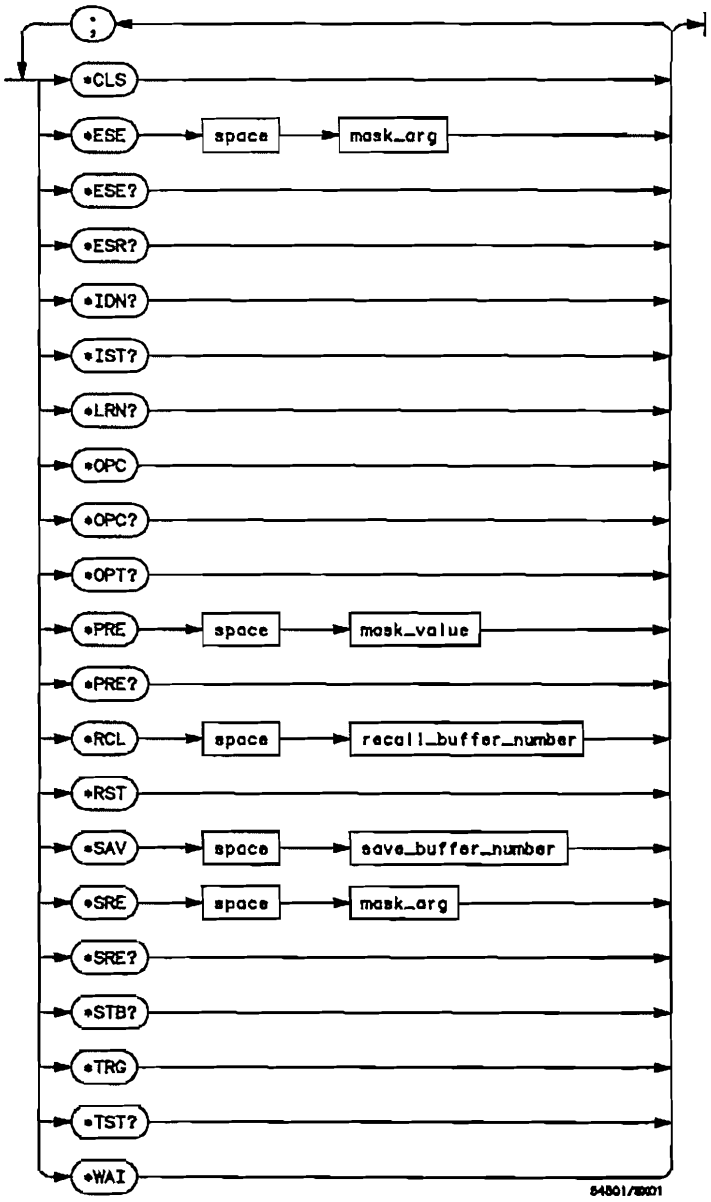


Figure 5-1. Common Commands Syntax Diagram

mask_arg = An integer, 0 through 255. This number is the sum of all the bits in the mask corresponding to conditions that are enabled. Refer to the *ESE and *SRE commands for bit definitions in the enable registers.

mask_value = An integer, 0 through 255. This number is the sum of all the bits in the mask corresponding to conditions that are enabled. Refer to the *IST? query.

recall_buffer_number = An integer, 0 through 4.

save_buffer_number = An integer, 1 through 4.

Figure 5-1. Common Commands Syntax Diagram (continued)

*CLS (Clear Status)

The *CLS (clear status) common command clears the status data structures, including the device defined error queue. This command also clears the Request-for-OPC flag.

If the *CLS command immediately follows a PROGRAM MESSAGE TERMINATOR, the output queue and the MAV (message available) bit will be cleared.

Command Syntax: *CLS

Example:

```
OUTPUT 707;"*CLS"
```

Note Refer to chapter 3 for a complete discussion of status.

*ESE (Event Status Enable)

The *ESE command sets the Standard Event Status Enable Register bits. The Standard Event Status Enable Register contains a mask value for the bits to be enabled in the Standard Event Status Register (see HP-IB command *ESR?). A "one" in the Standard Event Status Enable Register will enable the corresponding bit in the Standard Event Status Register, a "zero" will disable the bit. Refer to Table 5-1 for the information about the Standard Event Status Enable Register bits, bit weights, and what each bit masks.

The *ESE query returns the current contents of the Standard Event Status Enable Register.

Command Syntax:

*ESE <mask>

Where:

<mask> ::= 0 to 255

Example:

```
OUTPUT 707;"*ESE  64"
```

In this example, the *ESE 64 command will enable URQ (user request) bit 6 of the Standard Event Status Enable Register. Therefore, when a front panel key is pressed, the ESB (event summary bit) in the Status Byte Register will also be set.

Query Syntax: *ESE?

Returned Format:

<mask><NL>

Where:

<mask> ::= 0 to 255 (integer—NR1 format)

Example:

```
OUTPUT 707;"*ESE?"
ENTER 707;Event
PRINT Event
END
```

Table 5-1.
Standard Event Status enable Register

Event Status Enable Register (High–Enables the ESR bit)		
Bit	**Weight**	**Enables**
7	128	PON–Power On
6	64	URQ–User Request
5	32	CME–Command Error
4	16	EXE–Execution Error
3	8	DDE–Device Dependent Error
2	4	QYE–Query Error
1	2	RQC–Request Control
0	1	OPC–Operation Complete

Note Refer to chapter 3 for a complete discussion of status.

*ESR? (Event Status Register)

The *ESR query returns the contents of the Standard Event Status Register.

Note

Reading the register clears the Standard Event Status Register.

Query Syntax: *ESR?

Returned Format:

<status><NL>

Where:

<status> ::= 0 to 255

Example:

```
OUTPUT 707;"*ESR?"
ENTER 707;Event
PRINT Event
END
```

Table 5-2 shows the Event Status Register. The table shows each bit in the Event Status Register and the bit weight. When you read the Event Status Register, the value returned is the total bit weights of all bits that are high at the time you read the byte.

Table 5-2. Standard Event Status Register

Bit	Bit Weight	Bit Name	Condition
7	128	PON	1 = an OFF to ON transition has occurred
6	64	URQ	0 = no front-panel key has been pressed
			1 = front-panel key has been pressed
5	32	CME	0 = no command errors
			1 = a command error has been detected
4	16	EXE	0 = no execution error
			1 = an execution error has been detected
3	8	DDE	0 = no device dependent errors
			1 = a device dependent error has been detected
2	4	QYE	0 = no query errors
			1 = a query error has been detected
1	2	RQC	0 = request control - NOT used - always 0
0	1	OPC	0 = operation is not complete
			1 = operation is complete
0 = False = Low 1 = True = High			

*IDN? (Identification Number)

The *IDN query allows the instrument to identify itself. It returns the string:

"HEWLETT-PACKARD,8990A,
<XXXXAYYYYY>,<MMDD>"

Where:

<XXXXAYYYYY> ::= the serial number of this instrument.

<MMDD> ::= the software revision of this instrument. The first two parameters represent the month, and the second two parameters represent the day of the month.

An *IDN query must be the last query in a message. Any queries after the *IDN query in this program message will be ignored.

Query Syntax: *IDN?

Returned Format:

HEWLETT-PACKARD,8990A,
XXXXAYYYYY,MMDD<NL>

Example:

```
DIM Id$[100]
OUTPUT 707;"*IDN?"
ENTER 707;Id$
PRINT Id$
END
```

*IST? (Individual Status Query)

The *IST query returns the current state of the IEEE 488.1 defined "ist" local message in the instrument. This is the same information returned during a parallel poll of the Peak Power Analyzer. The response to this query is dependent upon the current status of the instrument

Query Syntax: *IST?

Returned Format:

<id><NL>

Where:

<id> ::= 0 or 1

Where:

0 indicates the "ist" local message is false.
1 indicates the "ist" local message is true.

Example:

```
OUTPUT 707;"*IST?"
ENTER 707;Id
PRINT Id
END
```

*LRN? (Learn)

The *LRN query returns a program message that contains the current state of the Peak Power Analyzer.

This command allows you to store an instrument setup in the controller. The stored setup can then be returned to the instrument when you want that setup at a later time.

Note The format of the learn string contents may change if the firmware version is changed. This means that the format of the stored learn string may not be compatible with the new firmware.

This command performs the same function as the :SYSTEM:SETUP? query. The data can be sent to the instrument using the :SYSTEM:SETUP command.

Note The Peak Power Analyzer always returns the header :SYSTEM:SETUP for the LRN query, whether the current :SYSTEM:HEADER setting is ON or OFF.

Query Syntax: *LRN?

Returned Format:

:SYSTem:SETup <setup><NL>

Where:

<setup> ::= #800001024<learn string><NL>

The learn string is 1024 data bytes in length.

Example:

```
DIM Lrn$[2000]
OUTPUT 707;"*LRN?"
ENTER 707 USING "-K";Lrn$
END
```

*OPC (Operation Complete)

The *OPC (operation complete) command will cause the instrument to set the operation complete bit in the Standard Event Status Register when all pending device operations have finished.

The *OPC query places an ASCII "1" in the output queue when all pending device operations have finished.

Command Syntax: *OPC

Example:

OUTPUT 707;"*OPC"

Query Syntax: *OPC?

Returned Format:

1<NL>

Example:

```
OUTPUT 707;":AUTOSCALE;*OPC?"
ENTER 707;Op
PRINT Op
END
```

*OPT?

The *OPT query will report the total number of installed channels (sensor and oscilloscope), the number of plugged in sensors, and whether or not a Sensor Check Source is installed.

Query Syntax: *OPT?

Return Syntax:

<Channels>,<Sensors>,<Sensor Check Source><NL>

Where:

<Channels> ::= 2, 3, or 4
<Sensors> ::= 0, 1, or 2
<Sensor Check Source> ::= 0 or 1

Example:

```
DIM Value$[100]]
OUTPUT 707;"*OPT?"
ENTER 707;Value$
PRINT Value$
END
```

*PRE (Parallel Poll Register Enable)

The *PRE command sets the parallel poll register enable bits.

The Parallel Poll Enable Register contains a mask value for the bits to be enabled that can produce an "ist" during a parallel poll. For additional information, refer to the *IST? command in this section.

The *PRE query returns the current value.

Command Syntax: *PRE <mask>

Where:

<mask> ::= 0 to 255

Example:

OUTPUT 707;"*PRE 16"

Note

This example will allow the Peak Power Analyzer to generate an "ist" when a message is available in the output queue. When a message is available, the MAV bit in the Status Byte Register will be set high.

Query Syntax: *PRE?

Returned Format:

<mask_value><NL>

Where:

<mask_value> ::= sum of all bits that are set–0 through 255.

Example:

```
OUTPUT 707;"*PRE?"
ENTER 707;V1
Print V1
END
```

*RCL (Recall)

The *RCL command restores the state of the Peak Power Analyzer from the specified internal save/recall register. An instrument setup must have been stored previously in the specified register. Registers 1 through 4 are general purpose and can be used with the *SAV command. Register 0 is a special recall only register, because it recalls the state that existed before the last AUTOSCALE, RECALL, ECL, or TTL operation.

Note 

An error message will appear on screen if nothing has been previously saved in the specified register.

Command Syntax: *RCL <rcl_register>

Where:

<rcl_register> ::= 0 through 4

Example:

OUTPUT 707;"*RCL 3"<NL>

*RST (Reset)

The *RST command places the Peak Power Analyzer in a known state. Refer to Table 5-3 for the reset conditions.

Command Syntax: *RST

Example:

```
OUTPUT 707;"*RST"
```

Table 5-3.
Reset Conditions for the Peak Power Analyzer

Timebase Menu	
reference	cntr
Time/Div	100 μs
delay	0.00 s
timebase window	off
Channel Menu	
Channel 1	on (With sensor connected.)
Channel 2, 3, and 4	off
Scale	1 mW/Div
external loss	0 dB
Bandwidth	Auto
Volts/Div	500 mV
offset	0.00
coupling	dc
probe attenuation	1.000:1
Trigger Menu	
Mode	edge
triggering	auto
source	Channel 1
level	0.0 W
slope	positive
holdoff	40 ns

Table 5-3.
Reset Conditions for the Peak Power Analyzer
(continued)

Display Menu	
Mode	norm
persistence	minimum
Power Display	linear
off/frame/axes/grid	frame
connect dots	off
# of screens	1
Markers	
Time markers	off
Amplitude markers	off
Carrier Frequency	
ch1=ch4	1 GHz
Waveform Math Menu	
f1	off
f2	off
display	off
chan/mem	chan 1
operator	+
chan/mem	chan 1
function sensitivity	64 mW/Div
function offset	0.0 W
Waveform Memory Menu	
waveform/pixel	waveform
nonvolatile	m1
display	off
source	chan 1

Table 5-3.
Reset Conditions for the Peak Power Analyzer
(continued)

Define Meas Menu	
meas/def/limit	meas
meas	
continuous	on
statistics	off
meas def	
definition	standard
meas limit	
test	off
set	risetime
fail if >	50.0000 s
or if <	1.000 ns
save to	off
after fail	stop
Utility Menu	
HP-IB Menu	
form feed	off
paper length	11 in.
clicker	on
check source	pulse

*SAV (SAVE)

The *SAV command stores the current state of the Peak Power Analyzer in an internal save register. The data parameter is the number of the save register where the data will be saved. Internal registers 1 through 4 are valid for this command.

Command Syntax: *SAV <save_register>

Where:

<save_register> ::= 1 through 4

Example:

```
OUTPUT 707;"*SAV 3"
```

*SRE (Service Request Enable)

The *SRE command sets the Service Request Enable Register bits. The Service Request Enable Register contains a mask value for the bits to be enabled in the Status Byte Register. A one in the Service Request Enable Register will enable the corresponding bit in the Status Byte Register, a zero will disable the bit. Refer to Table 5-5 for the bits in the Service Request Enable Register and what they mask.

The *SRE query returns the current value.

Command Syntax: *SRE <mask>

Where:

<mask> ::= 0 to 255

Example:

```
OUTPUT 707;"*SRE 16"
```

Note

This example enables a service request to be generated when a message is available in the output queue. When a message is available the MAV bit will be high.

Query Syntax: *SRE?

Returned Format:

<mask><NL>

Where:

<mask> ::= sum of all bits that are set—0 through 255

Example:

```
OUTPUT 707;"*SRE?"
ENTER 707;Value
PRINT Value
END
```

Table 5-4. Service Request Enable Register

Service Request Enable Register (High –Enables the SRE bit)		
Bit	**Weight**	**Enables**
7	128	not used
6	64	RQS–Request Service
5	32	ESB–Event Status Bit
4	16	MAV–Message Available
3	8	LTF–Limit Test Fail
2	4	MSG–Message
1	2	LCL–Local
0	1	TRG–Trigger

*STB (Status Byte)

The *STB query returns the current value of the instrument's status byte. The MSS (Master Summary Status) bit is reported on bit 6 instead of the RQS (request service) bit. The MSS indicates whether or not the device has at least one reason for requesting service. Refer to Table 5-6 for the meaning of the bits in the status byte.

Note

To read the instrument's status byte with RQS reported on bit 6, use the HP-IB Serial Poll.

Query Syntax: *STB?

Returned Format:

<value><NL>

Where:

<value> ::= 0 through 255 (integer–NR1)

Example:

```
OUTPUT 707;"*STB?"
ENTER 707;Value
PRINT  Value
END
```

Table 5-5. The Status Byte Register

Bit	Bit Weight	Bit Name	Condition
7	128	–	0 = not used
6	64	RQS/MSS	0 = instrument has no reason for service
			1 = instrument is requesting service
5	32	ESB	0 = no event status conditions have occurred
			1 = an enabled event status condition has occurred
4	16	MAV	0 = no output messages are ready
			1 = an output message is ready
3	8	LTF	0 = no limit test has failed
			1 = limit test has failed
2	4	MSG	0 = no message has been displayed
			1 = message has been displayed
1	2	LCL	0 = a remote-to-local transition has not occurred
			1 = a remote-to-local transition has occurred
0	1	TRG	0 = no trigger has occurred
			1 = a trigger has occurred
0 = False = Low 1 = True = High			

*TRG (Trigger)

The *TRG command has the same effect as the Group Execute Trigger (GET). That effect is as if the Root Level Command RUN had been sent.

The *TRG command performs the same function as the front panel (SINGLE) key when :TIMEBASE:MODE is set to SINGLE, and it performs the same function as the front panel (RUN) key when :TIMEBASE:MODE is set to AUTO or TRIGGERED.

Command Syntax: *TRG

Example:

```
OUTPUT 707;"*TRG"
```

*TST? (Test)

The *TST query causes the Peak Power Analyzer to perform a self-test. The result of the test will be placed in the output queue.

Note

Prior to sending this command all front panel inputs must be disconnected.

If either vertical cal or delay cal have failed, the *TST? query will fail. For more information about vertical cal and delay cal, refer to the "instr cal menu" portion of the Utility Menu in the Operating Manual.

A 0 indicates the test passed and a non-zero value indicates the test failed.

If a test fails refer to the troubleshooting section of the Peak Power Analyzer Service Manual.

Query Syntax: *TST?

Returned Format:

<result><NL>

Where:

<result> ::= 0 or non-zero value

Where:

0 indicates the test passed.
Non-zero indicates the test failed.

Example:

```
OUTPUT 707;"*TST?"
ENTER 707;Result
PRINT Result
END
```

*WAI (Wait)

The *WAI command has **no function** in the Peak Power Analyzer, but is parsed for compatibility with other instruments.

Command Syntax: *WAI

Example:

```
OUTPUT 707;"*WAI"
```

5-32 This Page Intentionally Left Blank

6

Root Level Commands

Introduction

Root Level commands control many of the basic operations of the Peak Power Analyzer. These commands are always recognized by the parser if they are prefixed with a colon, regardless of current command tree position. They are also recognized at the beginning of each new program message, which is more common. After executing a root level command, the parser is positioned at the root of the command tree. Figure 6-1 lists the root level command syntax diagrams.

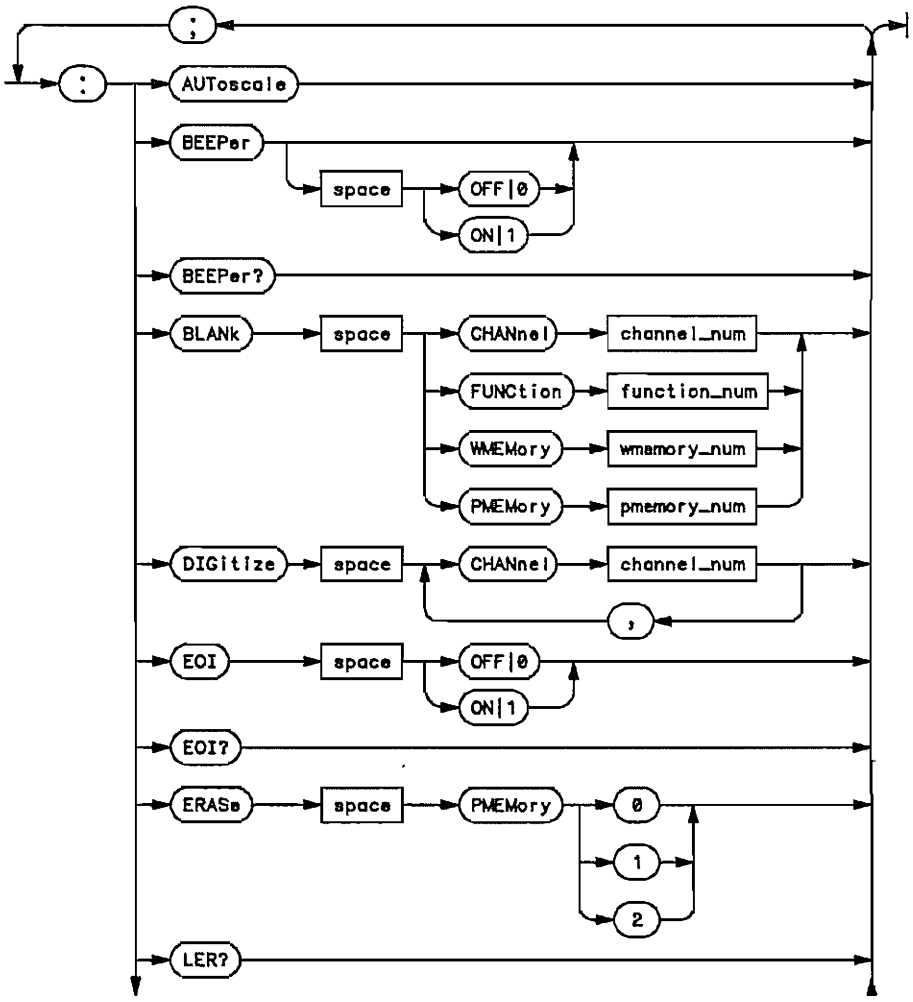


Figure 6-1. Root Level Commands Syntax Diagrams

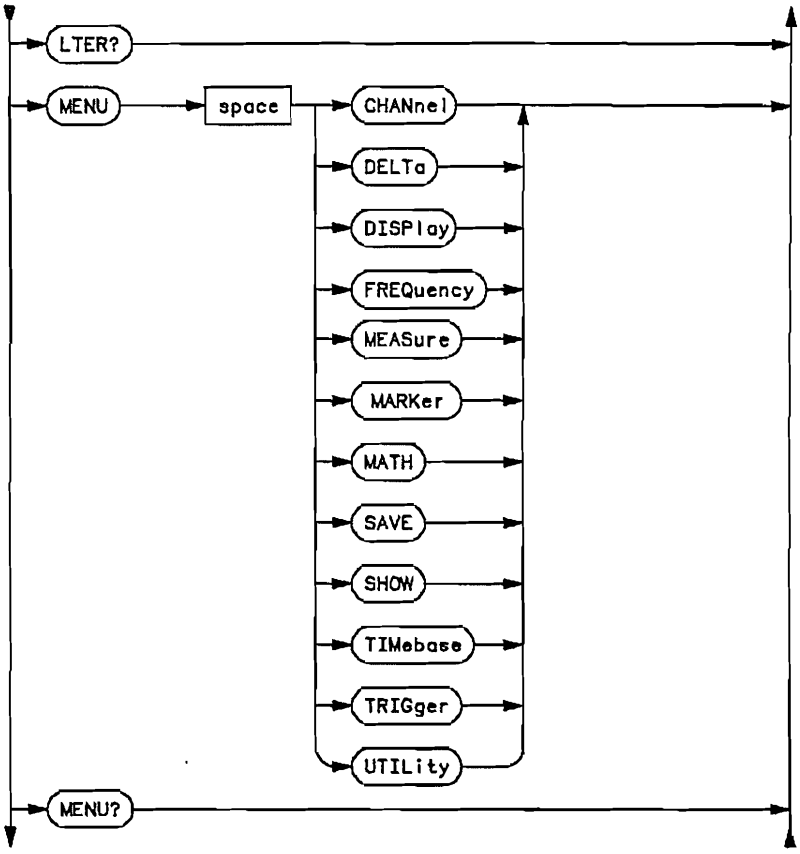
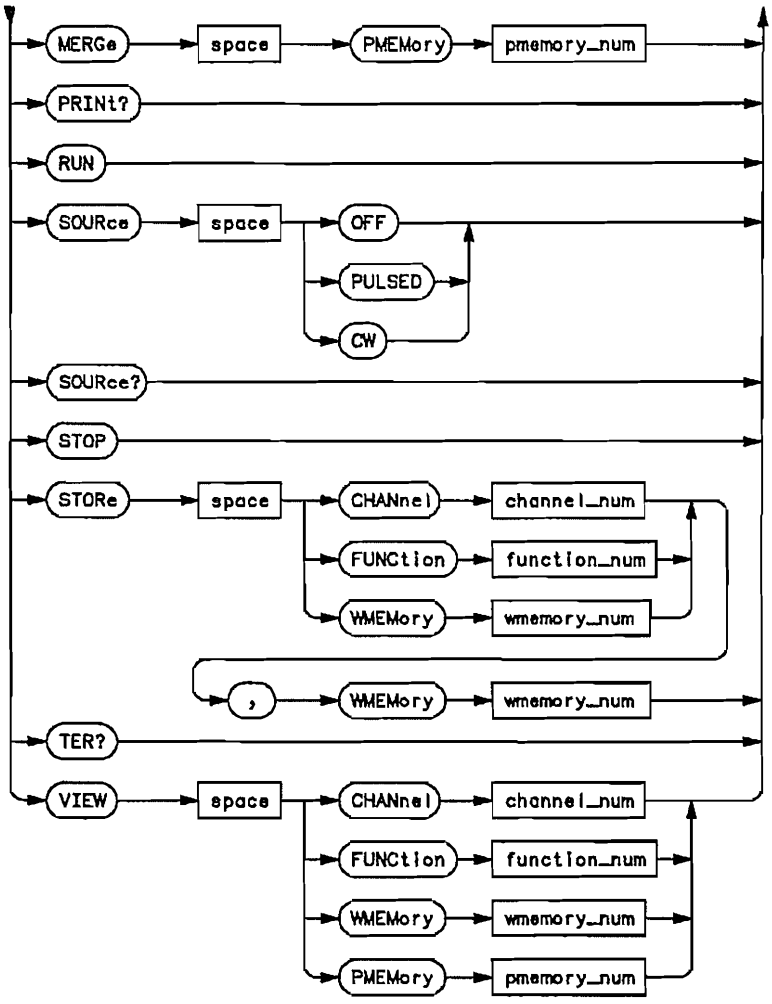


Figure 6-1. Root Level Commands Syntax Diagrams (continued)



channel_num = an integer 1 through 4
function_num = an integer 1 or 2
wmemory_num = an integer 1 through 4
pmemory_num = an integer 1 or 2

Figure 6-1. Root Level Commands Syntax Diagrams (continued)

AUToscale

The AUTOSCALE command causes the Peak Power Analyzer to evaluate all input signals and set the optimum conditions to display the signals. Autoscale is aborted by returning the Peak Power Analyzer to local and pressing any key or turning the knob. When the AUTOSCALE command is sent, the following conditions are set:

vertical sensitivity;

vertical offset; (Channels 2 and 3)

trigger, to edge mode with minimum persistence;

trigger level, holdoff, and slope;

sweep speed of the displayed channel; and

display to linear mode

In addition, the AUTOSCALE command turns off:

markers;

all measurements;

functions;

windows; and

memories

If signals are present on more than one vertical input, the sweep triggers on channel 2 if a signal is present on that channel. If a signal is not present on channel 2, the Peak Power Analyzer triggers on channel 3 if a signal is present on that channel. If a signal is not present on channel 3, the Peak Power Analyzer triggers on channel 1 if a signal is present on that channel, or on channel 4 if no other signals are found. If no signals are found on any

vertical input, the Peak Power Analyzer is returned to its former state.

Note

When using AUTOSCALE, it is not necessary to use the Common Command *OPC (Operation Complete). The Peak Power Analyzer does not allow another operation to begin until autoscale is complete.

A measurement command (Measure Subsystem) immediately following an autoscale may fail. The acquisition which is automatically initiated by autoscale may not be completed by the time the measurement command is executed. Refer to the Measure Subsystem for the correct procedure to follow when making a measurement after autoscale or any other command.

Under certain conditions, executing autoscale repeatedly on the same signal may produce a different sweep speed each time.

Command Syntax: :AUToscale

Example:

```
OUTPUT 707;":AUTOSCALE"
```

BEEPer

The BEEPER command sets the beeper mode, which controls the sound of the Peak Power Analyzer during most front panel key inputs. The beeper can be set to on or off. If the BEEPER command is sent without an argument, the beeper will be sounded without affecting the current beeper mode.

The BEEPER query returns the current state of the beeper mode.

Command Syntax: :BEEPer {{ON | 1} | {OFF | 0}}

Example:

```
OUTPUT 707;":BEEPER 1"
```

Query Syntax: :BEEPer?

Returned Format:

[:BEEPer] {1 | 0}<NL>

Example:

```
OUTPUT 707;":BEEP?"
ENTER 707;Click
PRINT Click
END
```

BLANk

The BLANK command causes the Peak Power Analyzer to turn off (stop displaying) the specified channel, function, pixel memory, or waveform memory. To blank a channel display, use the command :BLANK CHANNEL{1|2|3|4}. To blank a waveform memory display, use the command :BLANK WMEMORY{1|2|3|4}. To blank a pixel memory display, use the command :BLANK PMEMORY{1|2}, and to blank a function, use the command :BLANK FUNCTION{1|2}.

To turn on (start displaying) a specified channel, function, pixel memory, or waveform memory, refer to the Root Level Command :VIEW.

Command Syntax: :BLANk <display>

Where:

<display> ::= {CHANnel{1 | 2 | 3 | 4} | FUNCtion{1 | 2} | WMEMory{1 | 2 | 3 | 4} | PMEMory{1 | 2 }}

Example:

```
OUTPUT 707;":BLANK CHANNEL1"
```

DIGitize

The DIGITIZE command is used to acquire waveform data for transfer over the bus. It causes an acquisition to take place on the specified channel(s) with the resulting data being placed in the channel buffer.

The ACQUIRE subsystem commands are used to set up conditions such as TYPE, number of POINTS, and COUNT for the next DIGITIZE command. See the ACQUIRE subsystem for a description of these commands.

Note The ACQUIRE subsystem commands will affect how long a digitize will take; some combinations of commands can cause a digitize to take an unreasonably long time to complete. For additional information, refer to the COMPLETE command in the ACQUIRE subsystem.

To determine the actual number of points that are acquired and how the data will be transferred, refer to the WAVEFORM Subsystem commands.

Note Only the channel(s) which are being digitized will be displayed; the other channels are turned off. Digitizing a channel that was previously off results in that channel being turned on and left on.

When the Digitize operation is complete, the Peak Power Analyzer is placed in the stopped mode. When the instrument is restarted with a RUN command or the front panel RUN key, the digitized data stored in the channel buffers is overwritten. Therefore, ensure all operations (for example, measurements) that require the digitized data are completed before restarting the Peak Power Analyzer .

Additional DIGITIZE commands after the first digitize will run quicker if no instrument state parameters are changed. The digitize process is further improved if all channels that are to be digitized are done so with a single command line.

The sources for the :DIGITIZE command are channels 1 through 4.

Note

When using DIGITIZE, it is not necessary to use the Common Command *OPC (Operation Complete). The Peak Power Analyzer does not allow another operation to begin until a digitize is complete.

When Option 001, Single Sensor Input, is installed, channel 4 can't be specified. Error −360, "Missing a Baseband Board" is placed in the error queue when an attempt is made to specify channel 4.

Command Syntax: :DIGitize CHANnel<N>[,CHANnel<N>, CHANnel<N>,CHANnel<N>]

Where:

<N> ::= 1, 2, 3, or 4.

Example:

```
OUTPUT 707;":DIGITIZE CHANNEL1,CHANNEL2"
```

EOI
(End or Identify)

The EOI command specifies whether or not the last byte of a reply from the Peak Power Analyzer is to be sent with the EOI HP-IB control line set true or not true. The last byte of a response is usually a "new line" character, ASCII decimal 10 (LF). EOI is set to ON after a key-down-power-up reset.

Note

EOI must be turned on for response messages to be in compliance with IEEE 488.2.

The EOI query returns the current state of EOI.

Command Syntax: :EOI {{ON | 1} | {OFF | 0}}

Example:

```
OUTPUT 707;":EOI OFF"
```

Query Syntax: :EOI?

Returned Format:

[:EOI] {1 | 0}<NL>

Example:

```
OUTPUT 707;":EOI?"
ENTER 707;End
PRINT End
END
```

ERASe

The ERASE command erases a specified pixel memory.

Erasing pixel memory 0 is a special case which is the same as pressing the CLEAR DISPLAY front-panel key. If the Peak Power Analyzer is running and being triggered and ERASE PMEMORY0 is executed, the instrument will momentarily stop acquiring data, clear the CRT, and then continue with data acquisition.

Erasing pixel memory 1 or 2 clears the specified pixel memory and anything on the display from that pixel memory.

Note

Once you erase pixel memory 1 or 2, there is no way to retrieve the original information.

Command Syntax: :ERASe {PMEMory0 | PMEMory1 | PMEMory2}

Example:

```
OUTPUT 707;":ERASE PMEMORY1"
```

LER? (Local Event Register)

The LER query allows the LCL (Local) Event Register to be read. After the LCL Event Register is read, it is cleared. A "one" indicates a remote to local transition has taken place due to the front-panel [LOCAL] key being pressed. A "zero" indicates a remote to local transition has not taken place.

Once this bit is set, it can only be cleared by reading the Event Register or sending a *CLS command.

A Service Request (SRQ) can only be generated when the bit transitions from 0 to 1. Therefore, the bit must be cleared each time you would like a new Service Request to be generated.

Query Syntax: :LER?

Returned Format:

[:LER] {1 | 0}<NL>

Example:

```
OUTPUT 707;":LER?"
ENTER 707;Event
PRINT Event
END
```

LTER? (Limit Test Event Register)

The LTER query allows the Limit Test Event Register to be read. The Limit Test Event Register contains the Limit Test Fail bit. This bit is set when the limit test is active, and a limit test has failed. After the Limit Test Event Register is read, it is cleared.

A Service Request (SRQ) can only be generated when the bit transitions from 0 to 1. Therefore, the bit must be cleared each time you would like a new Service Request to be generated.

Query Syntax: :LTER?

Returned Format:

[:LTER] {1 | 0}<NL>

Example:

```
OUTPUT 707;":LTER?"
ENTER 707;Lmt
PRINT Lmt
END
```

MENU

The MENU command selects one of the 11 menus on the front panel.

The MENU query returns the current menu.

Command Syntax: :MENU <name><NL>

Where:

<name> ::= {TIMebase | CHANnel | TRIGger | DISPlay | {MARKer | DELTa} | FREQuency | MATH | SAVE | MEASure | UTILity | SHOW}

Example:

```
OUTPUT 707;":MENU DISPLAY"
```

Query Syntax: :MENU?

Returned Format:

[:MENU] <name><NL>

Where:

<name> ::= {TIMebase | CHANnel | TRIGger | DISPlay | DELTa[1] | FREQuency| MATH | SAVE[2] | MEASure | UTILity | SHOW}

[1]DELTA is returned for the MARKER menu. DELTA is provided for compatibility with other instruments.

[2]SAVE is returned for the WAVEFORM MEMORY menu. SAVE is provided for compatibility with other instruments..

Example:

```
DIM Menu$[100]
OUTPUT 707;":MENU?"
ENTER 707;Menu$
PRINT Menu$
END
```

MERGe

The MERGE command stores the contents of the active display into the specified pixel memory. The pixel memories are PMEMORY 1 or 2.

This command has an identical function as the "add to memory" key in the pixel menu of the Waveform Memory menu.

Command Syntax: :MERGe {PMEMory1 | PMEMory2}

Example:

```
OUTPUT 707;":MERGE PMEMory2"
```

PRINt?

The PRINT query outputs a copy of the display over HP-IB as soon as the Peak Power Analyzer is addressed to talk.

The output includes the displayed waveforms, the graticule, time and amplitude markers, trigger setup, and measurement results.

Query Syntax: :PRINt?

The following examples describe two ways to output the display data over HP-IB. In the first example, the controller directs the flow of data from the Peak Power Analyzer to the printer. In the second example, the controller takes a more active role. The display data is first sent to the controller, and then the controller outputs the data to the printer.

First Example:

```
10 OUTPUT 707;":HARDCOPY:PAGE AUTOMATIC"
20 OUTPUT 707;":PRINT?"
30 SEND 7;UNT UNL
40 ! Devices on interface stop talking and listening.
50 SEND 7;LISTEN 1
60 ! Printer is set to listen.
70 SEND 7;TALK 7
80 ! Peak Power Analyzer is set to talk.
90 SEND 7;DATA
100 ! Switch to data mode.
110 END
```

Second Example:

```
10  DIM Dump$[32765]
20  !Load raster dump commands into this array.
30  ASSIGN @Buffer TO BUFFER  [35000]
40  !Raw bits are read into this buffer
50  ASSIGN @Spike TO 707
60  !Access to instrument through I/O path
70  ASSIGN @Printer TO 701
80  !Access to printer through I/O path
90  !
100 CLEAR SC(@Spike)
110 !Interface clear out any previous attempts
120 OUTPUT @Spike;"EOI ON"
130 !Instrument send last byte of message with
140 !EOI set.
150 OUTPUT @Spike;"PRINT?"
160 !Tell instrument to dump display image
170 !
180 DISP "Reading image from instrument.."
190 !Tell operator we are waiting for data.
200 TRANSFER @Spike TO @Buffer;END,WAIT
210 !Read and wait until an EOI comes in.
220 DISP
230 !Clear message, we now have the data in
240 !buffer
250 !
260 DISP "Dumping to printer..."
270 TRANSFER @Buffer TO @Printer;WAIT
280 !Copy buffer contents over HP-IB to printer
290 DISP
300 !Clear display message, transfer now finished.
310 !
320 END
```

RUN

The RUN command acquires data for the active waveform display. The data is acquired in the manner defined by the timebase subsystem.

If the timebase subsystem MODE is set to SINGLE, the RUN command enables the trigger once and displays the acquired data on the CRT. This also occurs when the front panel [SINGLE] key is pressed, and the instrument is STOPPED.

If the timebase subsystem MODE is set to AUTO or TRIGGERED, the RUN command enables the trigger repeatedly and displays the data it acquires after each trigger continuously on the CRT. This is the same thing that happens when the front panel [RUN] key is pressed. See the :TIMEBASE:MODE command for a description of the various modes.

Command Syntax: :RUN

Example:

```
OUTPUT 707;":RUN"
```

SOURce

The SOURCE command controls the Sensor Check Source. The Sensor Check Source is either off, CW, or pulsed. At power-up or preset, the Sensor Check Source defaults to pulse.

The SOURCE query returns the current state of the Sensor Check Source.

Note Settling time for the Sensor Check Source is two to three seconds.

Command Syntax

:SOURCE {OFF | PULSed | CW}

Example

```
OUTPUT 707;":SOURce PULSED"
```

Query Syntax

:SOURCE?

Returned Format

[:SOURCE] {OFF | PULSed | CW}<NL>

Example

```
DIM Source$[100]
OUTPUT 707;":SOURCE?"
ENTER 707;Source$
PRINT Source$
END
```

STOP

The STOP command causes the Peak Power Analyzer to stop acquiring data for the active display. The command does not have any effect if a DIGITIZE is being performed.

The RUN command must be executed to restart the acquisition.

Command Syntax: :STOP

Example:

```
OUTPUT 707;":STOP"
```

STORe

The STORE command moves a channel, function, or stored waveform to an internal waveform memory. This command has two parameters. The first is the source of the waveform. The source can be specified as any channel, function, or waveform memory. The second parameter is the destination of the waveform which can only be waveform memories 1 through 4.

Note

When Option 001, Single Sensor Input, is installed, the advisory "no valid data ... nothing stored" is displayed on the front panel when an attempt is made to store channel 4.

Command Syntax: :STORe <source>,<destination>

where:

<source> ::= {CHANnel{1 | 2 | 3 | 4} | FUNCtion{1 | 2} | WMEMory{1 | 2 | 3 | 4}}
<destination> ::= WMEMory {1 | 2 | 3 | 4}

Example:

```
OUTPUT 707;":STORE CHANNEL2,WMEMORY4"
```

TER? (Trigger Event Register)

The TER query allows the Trigger Event Register to be read. When the Trigger Event Register is read, it is cleared. A "one" indicates a trigger has occurred. A "zero" indicates a trigger has not occurred.

Note

If a trigger event is not found and the sweep is auto-triggering, this bit will not be set.

A Service Request (SRQ) can only be generated when the bit transitions from 0 to 1; therefore, the bit must be cleared each time you would like an new Service Request to be generated.

Query Syntax: :TER?

Returned Format:

[:TER] {1 | 0}<NL>

Example:

```
OUTPUT 707;":TER?"
ENTER 707;Trg_event
PRINT Trg_event
END
```

VIEW

The VIEW command causes the Peak Power Analyzer to turn on (start displaying) an active channel, function, pixel memory, or waveform memory.

If you want to display a channel, use the command :VIEW CHANnel{1 | 2 | 3 | 4}. If you want to display a pixel memory, use the parameter :VIEW PMEMory{1 | 2}. To display a function, send the command :VIEW FUNCtion{1 | 2}.

The BLANK command causes the Peak Power Analyzer to turn off (stop displaying) a specified channel, function, pixel memory, or waveform memory.

Note

When Option 001, Single Sensor Input, is installed, channel 4 can't be specified. Error −360, "Missing a Baseband Board" is placed in the error queue when an attempt is made to specify channel 4.

Command Syntax: :VIEW {CHANnel{1 | 2 | 3 | 4} | FUNCtion{1 | 2} | PMEMory{1 | 2} | WMEMory{1 | 2 | 3 | 4}}

Example:

```
OUTPUT 707;":VIEW CHANNEL1"
```

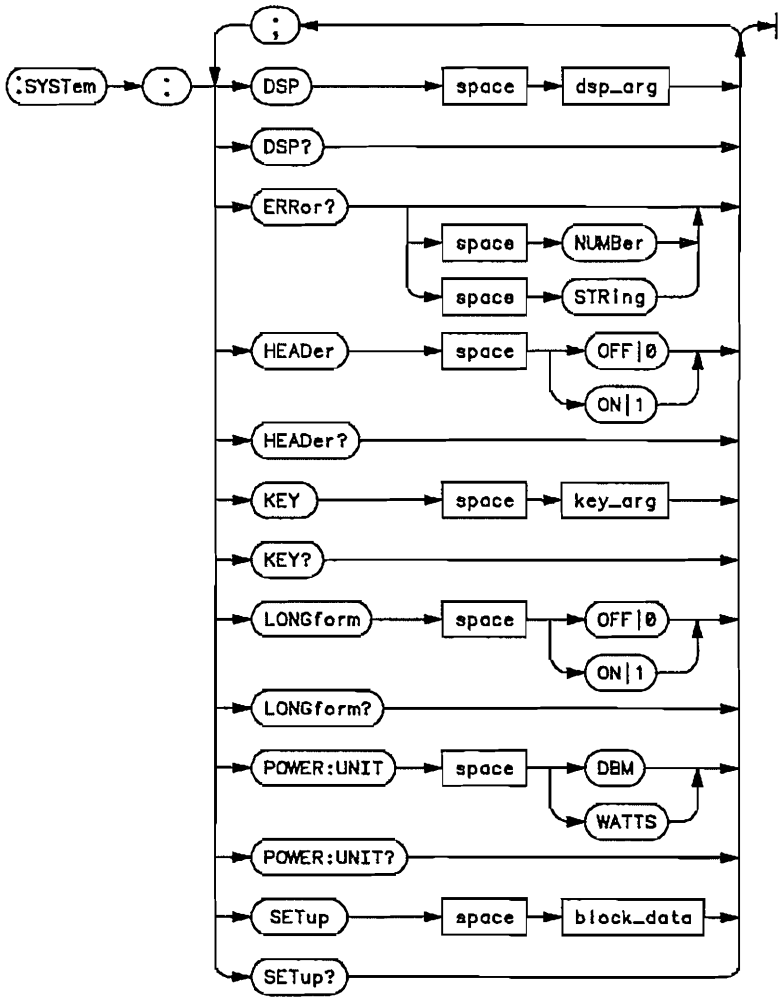
7

System Subsystem

Introduction

System subsystem commands control the way in which query responses are formatted, simulate front panel key presses, and enable reading and writing to the advisory line of the instrument.

Refer to Figure 7-1 for System Command syntax diagrams.



dsp_arg = any quoted string
key_arg = keycode 1 through 45
block_data = data in IEEE 488.2 # format

Figure 7-1. System Subsystem Syntax Diagrams

DSP

The :SYSTEM:DSP command writes a quoted string, excluding delimiting quotes, to the advisory line (line 1) of the CRT.

The DSP query returns the last string written to the advisory line. This may be a string written with a DSP command, an internally generated advisory, or an error message.

The string is actually read from the message queue which is one message deep. The message queue is cleared when it is read. Therefore, the displayed message can only be read once over the bus.

Command Syntax: :SYSTem:DSP <quoted ASCII string>

Example:

```
OUTPUT 707;":SYSTEM:DSP ""This is a message"""
```

Query Syntax: :SYSTem:DSP?

Returned Format:

[SYSTem:DSP] <string><NL>

Where:

<string> ::= string response data containing the last information written on the advisory line

Example:

```
DIM Display$[100]
OUTPUT 707;"SYSTEM:DSP?"
ENTER 707;Display$
PRINT Display$
END
```

ERRor?

The :SYSTEM:ERROR query outputs the next error number in the error queue over HP-IB. The Peak Power Analyzer has an error queue that is 30 errors deep and operates on a first-in, first-out basis. Successively sending the query, :SYSTEM:ERROR?, returns the error numbers in the order that they occurred until the queue is empty. If the error queue overflows, error −350 is returned. When the error queue is empty, any further queries return zeros until another error occurs.

When the NUMBER parameter is used in the query, only the numeric error code is output. When the STRING parameter is used, the error number is output followed by a comma and a quoted string. If no parameter is specified, the numeric error code is output. No parameter specified is the same as specifying NUMBER.

See Table 7-1 for the ERROR numbers.

Note

For an explanation of error codes +9.99999E+37 and −9.99999E+37, refer to "Measurement Error" at the beginning of Chapter 15, Measure Subsystem.

Query Syntax: :SYSTem:ERRor? {NUMBer | STRing | (no_param)}

Returned Format:

[:SYSTem:ERRor] <error>[,<quoted string>]<NL>

Where:

<error> ::= an integer error code

<quoted string> ::= an alpha string specifying the error condition

Example:

```
DIM Emsg$[100]
OUTPUT 707;":SYSTEM:ERROR?"
ENTER 707;Emsg$
PRINT Emsg$
END
```

Table 7-1. Error Messages

Error Number	Description
10	Data invalidated
11	Questionable horizontal scaling
12	Edges required not found
20	Invalid on/off parameter
70	Ram write protected
−100	Command error (unknown command)
−101	Invalid character received
−110	Command header error
−111	Header delimiter error
−120	Numeric argument error
−121	Wrong data type (numeric expected)
−123	Numeric overflow
−129	Missing numeric argument
−130	Non-numeric argument error
−131	Wrong data type (char expected)
−132	Wrong data type (string expected)
−133	Wrong data type (block expected)
−134	Data Overflow: string or block to long
−139	Missing non-numeric argument
−142	Too many arguments
−143	Argument delimiter error
−144	Invalid message unit delimiter
−145	Invalid suffix for power
−146	Invalid suffix for volts
−147	Valid only for voltages
−149	Valid only for power channel

Table 7-1. Error Messages (continued)

Error Number	Description
−150	Valid only for voltage channel
−200	No Can Do (generic execution error)
−201	Not executable in local mode
−202	Settings lost due to rtl or power on*
−203	Trigger ignored
−211	Legal command, but settings conflict
−212	Argument out of range
−221	Busy doing something else
−222	Insufficient capability or configuration
−232	Output buffer full or overflow
−233	Power Sensor is missing or unusable
−300	Device failure
−301	Interrupt fault
−302	System error
−303	Time out
−310	RAM error
−311	RAM failure (hard error)
−312	RAM data loss (soft error)
−313	Calibration data loss
−320	ROM error
−321	ROM checksum
−322	Hardware and firmware incompatible
−330	Power on test failed
−340	Self test failed

Table 7-1. Error Messages (continued)

Error Number	Description
−350	Too Many Errors (error queue overflow)
−360	Missing a baseband board
−370	Bad sensor data checksum
−371	Unknown sensor data format
−372	Sensor data read failure
−373	Sensor data write failure
−400	Query Error (generic)
−410	Query INTERRUPTED
−420	Query UNTERMINATED
−421	Query received, Indefinite block response in progress
−422	Addressed to Talk, Nothing to Say
−430	Query DEADLOCKED

*rtl= remote to local

HEADer

The :SYSTEM:HEADER command tells the Peak Power Analyzer whether or not to output a header for query responses. When HEADER is set to ON, query responses will include the command header.

Note

The HEADER command does not affect the header returned by the *LRN? query. The header for this query is always returned whether the HEADER command is set to on or off.

The HEADER query returns the state of the HEADER command.

Command Syntax: :SYSTem:HEADer {{ ON | 1 } | { OFF | 0 }}

Example:

```
OUTPUT 707;":SYSTEM:HEADER ON"
```

Query Syntax: :SYSTem:HEADer?

Returned Format:

[:SYSTem:HEADer] {1 | 0 }<NL>

Where:

1 ::= ON
0 ::= OFF

Example:

```
DIM Hdr$[100]
OUTPUT 707;":SYSTEM:HEADER?"
ENTER 707;Hdr$
PRINT Hdr$
END
```

The following example shows the response to the query :CHANNEL1:RANGE? with the headers on and off.

```
Headers ON; longform ON:    :CHANNEL1:RANGE +4.00000E-07
Headers ON; longform OFF:   :CHAN1:RANG +4.00000E-07
Headers OFF:                +4.00000E-07
```

Note

Headers should be turned off when returning values to numeric variables. For more information, refer to Chapter 1, Introduction to Programming the HP 8990A Peak Power Analyzer.

KEY

The :SYSTEM:KEY command simulates the pressing of a specified front panel key. Key commands may be sent over the HP-IB in any order that are legal key presses from the front panel. Make sure the Peak Power Analyzer is in the desired state before executing the KEY command.

The KEY query returns the key code for the last key pressed from the front panel or the last simulated key press over the HP-IB. Key codes range from 1 to 45; zero represents no key and is returned after power up.

Note

Some Utility Menu functions may act unpredictably if accessed with the KEY command. The HP-IB may lock up until the function is complete.

Concerning portability, softkey positions may change with a firmware revision.

Refer to Table 7-2 for key codes.

Command Syntax: :SYSTem:KEY <keycode>

Where:

<keycode> ::= 1 to 45

Example:

```
OUTPUT 707;":SYSTEM:KEY 2"
```

Query Syntax: :SYSTem:KEY?

Returned Format:

[:SYSTem:KEY] <keycode><NL>

Where:

<keycode> ::= 0 through 45 (integer–NR1 format)

Example:

```
DIM Input$[100]
OUTPUT 707;":SYSTEM:KEY?"
ENTER 707;Input$
PRINT Input$
END
```

Table 7-2.
Peak Power Analyzer Front-Panel Key Codes

Key	Key Code	Key	Key Code
Menus–TIMEBASE	1	FINE	17
Menus–CHAN/VERT	2	s W	18
Menus–TRIG	3	ms mW	19
Menus–DISPLAY	4	μs μW	20
Menus–MARKERS	5	ns nW	21
Menus–WFORM MATH	6	CLEAR	22
Menus–WFORM MEMORY	7	Shift (blue key)	23
Menus–DEFINE MEAS	8	"–"(minus)	24
Menus–UTIL	9	"." (decimal pt.)	25
Function Select 1	10	0	26
Function Select 2	11	1	27
Function Select 3	12	2	28
Function Select 4	13	3	29
Function Select 5	14	4	30
Function Select 6	15	5	31
Function Select 7	16	6	32

Table 7-2.
Peak Power Analyzer Front-Panel Key Codes
(continued)

Key	Key Code	Key	Key Code
7	33	HARDCOPY	40
8	34	AUTOSCALE	41
9	35	RECALL	42
RUN/STOP	36	SAVE	43
SINGLE	37	SHOW	44
CLEAR DISPLAY	38	Menus–CARRIER FREQ	45
LOCAL	39	no key	0

The function select keys are at the right of the CRT and are numbered from the top (10) to the bottom (16).

LONGform

The :SYSTEM:LONGFORM command sets the longform variable which tells the Peak Power Analyzer how to format query responses. If the LONGFORM command is set to OFF, command headers and alpha arguments are sent from the Peak Power Analyzer in the short form. If the LONGFORM command is set to ON, the whole word will be output. This command does not affect the input data messages to the Peak Power Analyzer. Headers and arguments may be sent to the Peak Power Analyzer in either the longform or shortform regardless of how the LONGFORM command is set. For more information, refer to the HEADER command in this chapter.

The LONGFORM query returns the state of the LONGFORM command.

Note

Even though the command can be sent using an alpha or numeric argument, the response is always a 1 or 0 (1 for ON, 0 for OFF).

Command Syntax: :SYSTem:LONGform {{ ON | 1 } | { OFF | 0 }}

Example:

```
OUTPUT 707;":SYST:LONG ON"
```

Query Syntax: :SYSTem:LONGform?

Returned Format:

[:SYSTem:LONGform]{1 | 0}<NL>

Where:

1 ::= ON
0 ::= OFF

Example:

```
DIM Long$[100]
OUTPUT 707;":SYSTEM:LONGFORM?"
ENTER 707;Long$
PRINT Long$
END
```

POWER:UNIT

Note The POWER:UNIT command only has an affect on channels 1 and 4

The :SYSTem:POWER:UNIT command sets the measurement units to either watts (linear) or dBm (log).

The POWER:UNIT query returns the current measurement units.

Note The Peak Power Analyzer performs all calculations on linear values. If log mode (dBm) has been selected, the results of the calculations are converted prior to displaying or outputting the data. The ability to select linear or log units was added as a convenience for the user.

The POWER:UNIT command does not affect the units for waveform data. Waveform data sent over the bus is always linear.

Command Syntax: :SYSTem:POWER:UNIT {DBM | WATTs}

Example:

```
OUTPUT 707;":SYST:POWER:UNIT WATTS"
```

Query Syntax: :SYSTem:POWER:UNIT?

Returned Format:

[:SYSTem:POWER:UNIT]{DBM | WATTS}<NL>

Note

If the :SYSTem:LONGform command is off, a query will return WATT if watts is the current unit.

Example:

```
DIM Unit$[100]
OUTPUT 707;":SYSTEM:POWER:UNIT?"
ENTER 707;Unit$
PRINT Unit$
END
```

SETup

The :SYSTEM:SETUP command sets the Peak Power Analyzer as defined by the data in a previously obtained learn string sent from the Peak Power Analyzer to the controller. The setup string contains 1024 bytes of setup data. The 1024 bytes do not include the header or "#800001024".

The SETUP query outputs the current Peak Power Analyzer setup in the form of a learn string to the controller. The SETUP query operates in the same manner as the *LRN? query. Unlike the *LRN? query, the header for the SETUP query can be turned off.

As discussed in Chapter 1, Introduction to Programming the HP 8990A Peak Power Analyzer, the learn string is sent and received as a binary block of data. The format for the data transmission is the # format defined in the IEEE 488.2 specification.

Note

The format of the learn string contents may change if the firmware version is changed. This means that the format of the stored learn string may not be compatible with the new firmware.

Command Syntax: :SYSTem:SETup <setup>

Example:

```
OUTPUT 707;":SYSTEM:SETUP <setup>"
```

Where:

<setup> ::= #800001024<setup data string>

Query Syntax: :SYSTem:SETup?

Returned Format:

[:SYSTem:SETup] <setup><NL>

Where:

<setup> ::= #800001024<setup data string>

Example:

```
10  DIM Set$[2000]
20  !Setup the instrument as desired
30  OUTPUT 707;":SYST:HEAD OFF"
40  OUTPUT 707;":SYSTEM:SETUP?"
50  !Transfer the instrument setup to controller
60  ENTER 707 USING "-K";Set$  !Store the setup
70  PAUSE  !Change the setup as desired
80  OUTPUT 707 USING "#,K";":SYST:SETUP ";Set$
90  !Returns the instrument to the first setup
100 END
```

Note

The logical order for this instruction entails sending the query first and following it with the command at a time of your choosing. The query causes the learn string to be sent to the controller, and the command causes the learn string to be returned to the Peak Power Analyzer.

The query saves the instrument state in the controller as opposed to an internal instrument state register.

8

Acquire Subsystem

Introduction

The ACQUIRE subsystem commands set up conditions for executing a DIGITIZE root level command to acquire waveform data. This subsystem selects the type of data, the number of averages, and the number of data points. See Figure 8-1 for the ACQUIRE subsystem syntax diagrams.

Note

The on-screen time is divided into one of a few possible number of horizontal time points as defined by the :ACQUIRE:POINTS command. Each of these increments in time is referred to as a time bucket with each time bucket having a fixed time interval associated with it.

The ACQUIRE subsystem is also the HP-IB control for two display parameters: Display Mode and Number of Averages. There is a coupling between the front panel and the ACQUIRE subsystem parameters. This means that when the HP-IB parameters for ACQUIRE TYPE or COUNT are changed, the Display Menu changes. Also, when the Display Menu parameters change, the HP-IB parameters change.

(Normal) Persistence mode

The :ACQUIRE:TYPE NORMAL command sets the Peak Power Analyzer to the variable persistence mode. The front panel user activates the same mode by selecting the Display menu and setting the display mode to Normal. The persistence time is set via the bus in the DISPLAY subsystem using the :DISPLAY:PERSISTENCE command.

The :ACQUIRE:COUNT command can be set in this mode, but has no impact on the current display mode or digitized waveform. The :ACQUIRE:COUNT query always returns a 1 in normal mode.

Averaging Mode

The :ACQUIRE:TYPE AVERAGE command sets the Peak Power Analyzer to the Averaging mode.

COUNT can be set in AVERAGE mode by sending the :ACQUIRE:COUNT command followed by the number of averages. In this mode, the value is rounded to the nearest power of 2, which determines the number of averages that must be acquired.

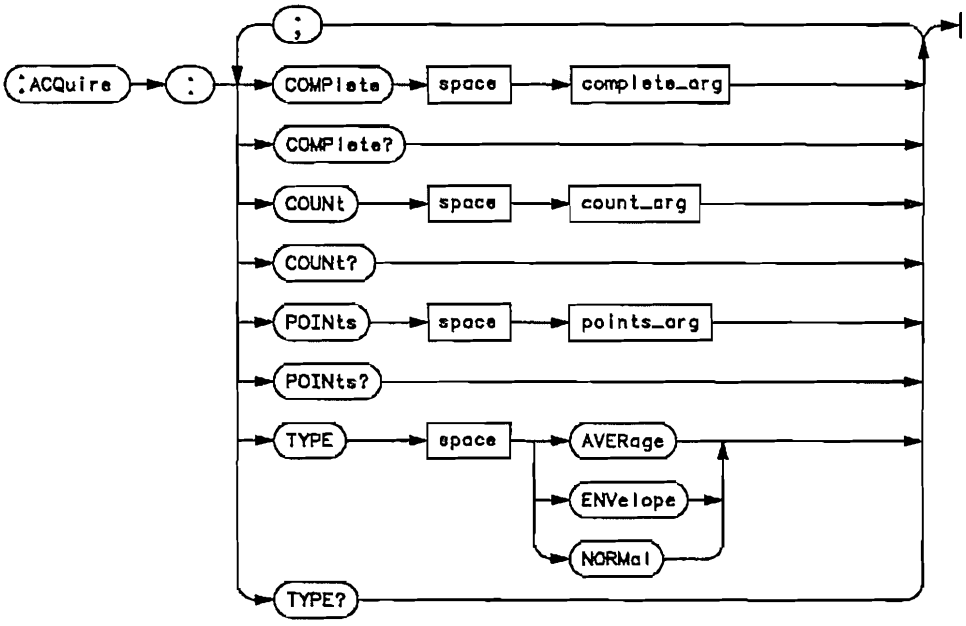
To activate the averaging mode from the front panel, select the Display menu, and then, select Average. Changing the number of averages changes the COUNT value.

Envelope Mode

The :ACQUIRE:TYPE ENVELOPE command sets the Peak Power Analyzer to the Envelope mode. In envelope mode, two waveforms are kept, one for the top part of the waveform and the other for the bottom of the waveform.

A COUNT value can be set in the envelope mode. This value determines the number of values to be used at each time point when constructing the envelope. The COUNT value cannot be set from the front panel in envelope mode.

To activate the Envelope mode from the front panel, select the Display menu, and then, select the Envelope mode.



complete_arg ::= An integer, 0 through 100.

count_arg ::= An integer, 1 to 2048. It specifies the number of values to average for each time point when in averaged mode, or the number of hits per each time point to form the envelope in Envelope Acquisition mode.

points_arg ::= 32, 64, 128, 256, 500, 512, or 1024.

Figure 8-1. Acquire Subsystem Syntax Diagrams

COMPlete

The :ACQUIRE:COMPLETE command specifies the completion criteria for an acquisition. The parameter determines what percentage of the time buckets need to be "full" before an acquisition is considered complete. If you are in the NORMAL mode, the Peak Power Analyzer only needs one data bit per time bucket for that time bucket to be considered full. In order for the time bucket to be considered full in the AVERAGE or ENVELOPE modes, a specified number of data points (COUNT) must be acquired.

The range for the COMPLETE command is an integer 0 to 100 inclusive and indicates the percentage of time buckets that must be "full" before the acquisition is considered complete. If the complete value is set to 100%, all time buckets must contain at least one sample for the acquisition to be considered complete. If the complete value is set to 0, then one acquisition cycle takes place.

Note

With an extremely fast or slow timebase setting, a DIGITIZE may take an unreasonably long time to complete. To reduce completion times, decrease the percent COMPLETE, or reduce the resolution by changing the timebase, the number of POINTS requested, and the COUNT.

The COMPLETE query returns the completion criteria for the currently selected mode.

Command Syntax:

:ACQuire:COMPlete <comp>

Where:

<comp> ::= 0 to 100 percent

Example:

```
OUTPUT 707;":ACQUIRE:COMPLETE 85"
```

Query Syntax:

:ACQuire:COMPlete?

Returned Format:

[:ACQuire:COMPlete] <comp><NL>

Where:

<comp> ::= 0 to 100 (integer–NR1 format)

Example:

```
DIM Cmp$[100]
OUTPUT 707;":ACQUIRE:COMPLETE?"
ENTER 707;Cmp$
PRINT Cmp$
END
```

COUNt

In average mode, the :ACQUIRE:COUNT command specifies the number of values to be averaged for each time bucket before the acquisition is considered complete for that time bucket.

When acquisition type is set to NORMAL, the count is 1.

When the acquisition type is set to AVERAGE, the count can range from 1 to 2048. Any value can be sent; however, the parameter is changed to the nearest power of 2.

When the acquisition type is set to ENVELOPE, the count can be any value between 1 and 2048.

The COUNT query returns the currently selected count value.

Command Syntax: :ACQuire:COUNt <count>

Where:

<count> ::= 1 to 2048

Example:

```
OUTPUT 707;":ACQUIRE:TYPE AVERAGE;COUNT 1024"
```

Query Syntax: :ACQuire:COUNt?

Returned Format:

[:ACQuire:COUNt] <count><NL>

Where:

<count> ::= 1 through 2048 (integer–NR1 format)

Example:

```
DIM Cnt$[100]
OUTPUT 707;":ACQ:COUNT?"
ENTER 707;Cnt$
PRINT Cnt$
END
```

POINts

The :ACQUIRE:POINTS command specifies the number of time buckets for each acquisition record. The legal settings are 32, 64, 128, 256, 500, 512, or 1024.

Any value between 32 and 1024 can be sent to the Peak Power Analyzer. If a value is sent that is not one of the legal values, it is changed to the nearest power of two. If a number smaller than 31 or greater than 1024 is sent, the instrument will generate an error response, and the points are set to the upper or lower limit.

There are some sweep speeds where the actual number of points is less than requested. These are shown in the following table:

Note 

Although the sweep speeds in the following table have points limitations, the POINTS value can be set to 1024. The Peak Power Analyzer sets the points to the limit for the desired sweep speed.

Sweep Speed (Per Division)				
	2 ns	**5 ns**	**10 ns**	**>10 ns**
Points Allowed	32, 64, 128, 200	32, 64, 128, 256, 500	32, 64, 128, 256, 500, 512, 1000	32, 64, 128, 256, 500, 512, 1024

The POINTS query returns the number of time buckets to be acquired.

Note 

Use the WAVEFORM:POINTS? query to determine the actual number of time buckets acquired.

Command Syntax:

:ACQuire:POINts <points_arg>

Where:

<points_arg>::= 32 to 1024 (see above for legal values)

Example:

```
OUTPUT 707;":ACQ:POINTS 512"
```

Query Syntax:

:ACQuire:POINts?

Returned Format:

[:ACQuire:POINts] <points_arg><NL>

Where:

<points_arg> ::= 32–1024 (see above for legal values)

Example:

```
DIM Pnts$[100]
OUTPUT 707;":ACQUIRE:POINTS?"
ENTER 707;Pnts$
PRINT Pnts$
END
```

TYPE

The :ACQUIRE:TYPE command selects the type of acquisition that takes place when a DIGITIZE root level command is executed. There are three acquisition types: NORMAL, AVERAGE, and ENVELOPE. Refer to the discussion at the beginning of this chapter for more about NORMAL, AVERAGE, and ENVELOPE modes.

The :ACQUIRE:TYPE query returns the current acquisition type.

Command Syntax:

:ACQuire:TYPE {NORMal | AVERage | ENVelope}

Example:

```
OUTPUT 707;":ACQUIRE:TYPE ENVELOPE"
```

Query Syntax:

:ACQuire:TYPE?

Returned Format:

[:ACQuire:TYPE] <type><NL>

Where:

<type> ::= {NORMal | AVERage | ENVelope}

Example:

```
DIM Tpe$[100]
OUTPUT 707;":ACQUIRE:TYPE?"
ENTER 707;Tpe$
PRINT Tpe$
END
```

9

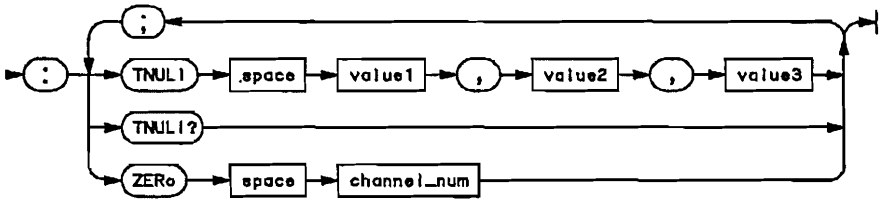
Calibrate Subsystem

Introduction

The Calibrate subsystem contains two commands. The TNULL command calibrates the Peak Power Analyzer for different probes, cables, or setups. The ZERO command zeros the peak power sensor at low power levels.

Note

The time null is set in the Probe Cal menu of the Utility menu. For more information, refer to the Peak Power Analyzer Operating Manual.



value1 = channel 1 to channel 2 skew
value2 = channel 1 to channel 3 skew
value3 = channel 1 to channel 4 skew
channel = 1 or 4

Figure 9-1. Calibrate Subsystem Syntax Diagrams

TNULl

The :CALIBRATE:TNULL command sends the time null (channel-to-channel skew) values to the Peak Power Analyzer. The time null values should have been obtained from the Peak Power Analyzer during a previous setup.

The TNULL query tells the Peak Power Analyzer to output the time null values to the controller.

Command Syntax: :CALibrate:TNULl
<null_value1>,<null_value2>,<null_value3>

Where:

<null_value1> ::= channel 1 to channel 2 skew
<null_value2> ::= channel 1 to channel 3 skew
<null_value3> ::= channel 1 to channel 4 skew

Example:

```
OUTPUT 707;":CAL:TNUL <null_value1>,<null_value2>,
<null_value3>
```

Query Syntax: :CALibrate:TNULl?

Returned Format:

[:CALibrate:TNULl] <null_value1>,<null_value2>,
<null_value3><NL>

Where:

<null_value1> ::= channel 1 to channel 2 skew
(exponential –NR3 format)
<null_value2> ::= channel 1 to channel 3 skew
(exponential –NR3 format)
<null_value3> ::= channel 1 to channel 4 skew
(exponential –NR3 format)

Example:

```
DIM N11$[100]
OUTPUT 707;":CALIBRATE:TNULL?"
ENTER 707;N11$
PRINT N11$
END
```

ZERo

At power levels less than 10 μwatts, the peak power sensor may contribute an offset that will affect the Peak Power Analyzer's accuracy specification. The command :CALIBRATE:ZERO eliminates the offset. If the power level is greater than 10 μwatts and the vertical sensitivity is set to display the signal, the Peak Power Analyzer meets its accuracy specification without the use of the :ZERO command.

Using the command is very easy:

- The peak power sensor may remain connected to the RF source. If the peak power sensor is disconnected from the RF source, any ground loop currents are not zeroed out.
- If possible, turn the RF source off. Otherwise, disconnect the peak power sensor from the RF source.
- Send the command :CALIBRATE:ZERO. The zeroing takes less than one minute.
- The advisory "Sensor zero completed" indicates when zeroing is complete.

The procedure needs to be repeated when accuracy is a concern for signals less than 10 μwatts and the peak power sensor has been disconnected from the Peak Power Analyzer since the last zeroing.

Note

When Option 001, Single Sensor Input, is installed, channel 4 can't be specified. Error −360, "Missing a Baseband Board" is placed in the error queue when an attempt is made to specify channel 4.

Command Syntax: :CALibrate:ZERo <Channel>

Where:

<Channel> ::= {CHANnel1 | CHANnel4}

Example:

```
OUTPUT 707;":CAL:ZER Chan1"
```

10

Channel Subsystem

Introduction

The CHANNEL subsystem commands control the channel display and vertical or Y axis parameters of the Peak Power Analyzer. Channels 1 through 4 are independently programmable.

The channel commands can be sent with a channel number specified or not specified. If a channel number is specified in the command, then the specified channel is affected; however, if the channel number is **not** specified, then channel 1 is assumed.

The channel displays are toggled on and off with the root level commands VIEW and BLANK.

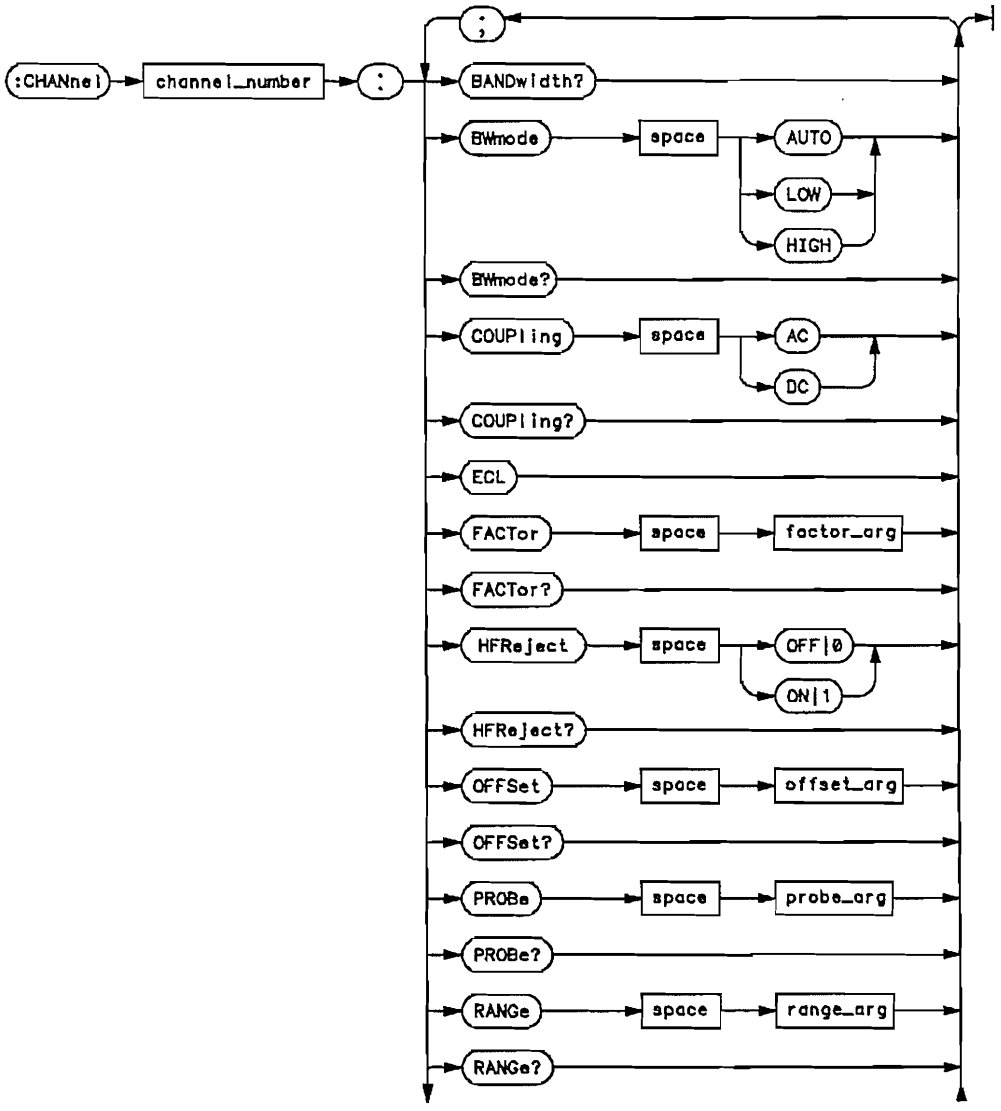
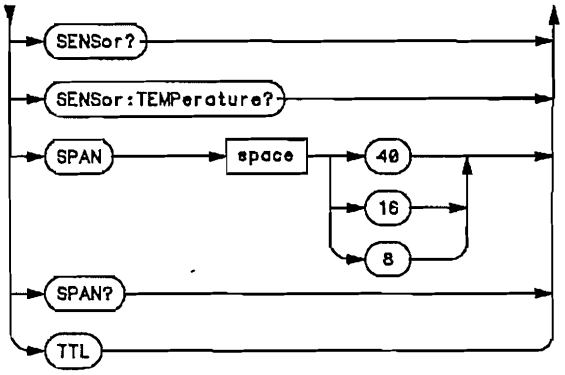


Figure 10-1. Channel Subsystem Syntax Diagrams



channel_number = 1, 2, 3, or 4

factor_arg = A real number specifying the signal loss external to the Analyzer.

offset_arg = A real number defining the voltage at the center of the display range.

probe_arg = A real number from 0.9 to 1000.0 specifying the probe attenuation with respect to 1.

range_arg = A real number specifying the full scale vertical axis.

Figure 10-1. Channel Subsystem Syntax Diagrams (continued)

BANDwidth?

Note

The BANDWIDTH query can only be used with channels 1 and 4.

The :CHANNEL<N>:BANDWIDTH? query returns the nominal measurement bandwidth for the specified channel. The bandwidth is a function of :CHANNEL:BWMODE and :CHANNEL:RANGE.

Note

When the front panel Chan/Vert Menu vertical sensitivity or bandwidth is changed for channel 1 or 4, a :BANDWIDTH query will not reflect the current bandwidth until an acquisition has been completed.

Changing vertical sensitivity (:CHANNEL:RANGE) or bandwidth (:CHANNEL:BWMODE) over the bus does not require an acquisition to be run before performing a :BANDWIDTH query.

Query Syntax

:CHANnel<N>:BANDwidth?

Where:

<N> ::= 1 or 4

Note

Under certain conditions, the queried bandwidth will not be returned:

- Error code +9.99999E+37 is returned, for example, when a peak power sensor is not connected to the specified channel, or a baseband board is not installed for the specified channel.
- Error code −9.99999E+37 is returned when the bandwidth hasn't been determined. This could be because the vertical sensitivity was changed, and the Peak Power Analyzer is in stopped mode.
- Instruments having Option 001, Single Sensor Input, installed will not have a baseband board installed for channel 4. In addition to error code +9.99999E+37 being returned, error code −360,"Missing a Baseband Board" is placed in the error queue if channel 4 is specified.

Returned Format:

[:CHANnel<N>:BANDwidth] <bandwidth><NL>

Where:

<bandwidth> ::= The bandwidth of the specified channel.

Example:

```
DIM Bw$ [100]
OUTPUT 707;":CHAN1:BANDWIDTH?"
ENTER 707;Bw$
PRINT Bw$
END
```

BWMode

Note 

The BWMODE command can only be used with channels 1 and 4.

Noise may be visible on the detected signal from the peak power sensor. The :CHANNEL<N>:BWMODE command is used to specify filtering to minimize the effect of the noise. The command selects the available bandwidth of an internal low pass filter. The maximum bandwidth available is 150 MHz. There are two variable parameters which determine the bandwidth:

- Vertical sensitivity
- Under rare conditions, temperature at the peak power sensor affects internal amplifier selection.

The :BWMODE command has three possible settings:

- **Auto** When auto is selected, the Peak Power Analyzer selects the optimum level between available bandwidth and noise.
- **Low** When low is selected, the bandwidth is the lowest possible with the selected vertical sensitivity.
- **High** When high is selected, the bandwidth is the highest possible with the selected vertical sensitivity.

The following table lists representative performance for the three possible settings. Bandwidth switching points are dependant upon circuit gain as well as ambient temperature. Bandwidth may differ somewhat from unit to unit and over wide temperature swings.

Bandwidth Settings

Vertical Sensitivity (per division)	Auto	Low	High
20 mW to 20 μW	150 MHz	12 MHz	150 MHz
10 μW	12 MHz	25 kHz	150 MHz
5 μW	500 kHz	25 kHz	500 kHz
2 μW	8 kHz	600 Hz	500 kHz
1 μW	8 kHz	600 Hz	500 kHz
500 nW	8 kHz	600 Hz	8 kHz
200 nW	2.5 kHz	600 Hz	8 kHz
100 nW	600 Hz	600 Hz	2.5 kHz
50 nW	600 Hz	600 Hz	2.5 kHz

Note

The actual bandwidth is read by using the :BANDWIDTH query. The :BANDWIDTH query is described in another part of this section.

Command Syntax

:CHANnel<N>:BWMode {AUTO | LOW | HIGH}

Where:

<N> ::= 1 or 4

Query Syntax

:CHANnel<N>:BWMode?

Where:

<N> ::= 1 or 4

Returned Format:

[:CHANnel<N>:BWMode] {AUTO | LOW | HIGH}
<NL>

Example:

```
DIM Bwm$ [100]
OUTPUT 707;":CHAN1:BWMODE?"
ENTER 707;Bwm$
PRINT Bwm$
END
```

COUPling

Note

The COUPLING command can only be used with channels 2 and 3.

The :CHANNEL<N>:COUPLING command selects the input coupling for channels 2 and 3. The coupling for each channel can be set to AC or DC.

The COUPLING query returns the current coupling for the specified channel.

Command Syntax: :CHANnel<N>:COUPling {AC | DC}

Where:

<N> ::= 2 or 3

Example:

```
OUTPUT 707;":CHAN2:COUP DC"
```

Query Syntax

:CHANnel<N>:COUPling?

Where:

<N> ::= 2 or 3

Returned Format:

[:CHANnel<N>:COUPling] {AC | DC}<NL>

Example:

```
DIM Ch$ [100]
OUTPUT 707;":CHAN2:COUPLING?"
ENTER 707;Ch$
PRINT Ch$
END
```

ECL

Note 

The ECL command can only be used with channels 2 and 3.

The :CHANNEL<N>:ECL command sets the vertical range, offset, channel coupling, and trigger level of channels 2 and 3 for optimum viewing of ECL signals. The offset and trigger level are set to −1.3 volts, and the range is set to 1.6 volts full scale. Channel coupling is set to DC.

There is no query form of this command.

Command Syntax: :CHANnel<N>:ECL

Where:

<N> ::= 2 or 3

Example:

```
OUTPUT 707;":CHAN2:ECL"
```

FACTor

Note

The FACTOR command can only be used with channels 1 and 4.

The :CHANNEL<N>:FACTOR command is used to enter the external loss from a device such as coupler, amplifier, or cable. The loss must be entered in dB. Valid entries are −30 to +99 dB. The command does not change the actual input sensitivity of the Peak Power Analyzer. It changes the reference constants for scaling the display factors and for automatic measurements, trigger levels, etc.

The FACTOR query returns the current gain or loss factor for the selected channel.

Command Syntax: :CHANnel<N>:FACTor <value> [DB]

Where:

<N> ::= 1 or 4
<value> ::= loss in dB.

Example:

```
OUTPUT 707;":CHANNEL1:FACTOR 10 DB"
```

Query Syntax: :CHANnel<N>:FACTor?

Where:

<N> ::= 1 or 4

Returned Format:

[:CHANnel<N>:FACTor] <value><NL>

Where:

<N> ::= 1 or 4
<value> ::=loss in dB (exponential–NR3 format)

Example:

```
DIM Fact$[100]
OUTPUT 707;":CHAN1:FACT?"
ENTER 707;Fact$
PRINT Fact$
END
```

HFReject

Note

The HFREJECT command can only be used with channels 2 and 3.

The :CHANNEL<N>:HFREJECT command controls an internal lowpass filter. When ON, the bandwidth of the specified channel is limited to approximately 20 MHz. The bandwidth limit filter may be used with either AC or DC coupling.

The HFREJECT query returns the current setting.

Command Syntax: :CHANnel<N>:HFReject {{ON | 1} | {OFF | 0}}

Where:

<N> ::= 2 or 3

Example:

```
OUTPUT 707;":CHANNEL2:HFR ON"
```

Query Syntax: :CHANnel<N>:HFReject?

Where:

<N> ::= 2 or 3

Returned Format:

[:CHANnel<N>:HFReject] {1 | 0}<NL>

Example:

```
DIM Hf$[100]
OUTPUT 707;":CHAN2:HFR?"
ENTER 707;Hf$
PRINT Hf$
END
```

OFFSet

Note

The OFFSET command can only be used with channels 2 and 3.

The :CHANNEL<N>:OFFSET command sets the voltage that is represented at center screen for the selected channel. The range of possible values varies with the value set with the RANGE command. If you set the offset to a value outside the valid range, it will automatically be set to the nearest valid value.

The OFFSET query returns the current offset value for the selected channel.

Command Syntax: :CHANnel<N>:OFFSet <value>[V]

Where:

<N> ::= 2 or 3
<value> ::= offset value

Example:

```
OUTPUT 707;":CHAN2:OFFS 200M"
```

Query Syntax: :CHANnel<N>:OFFSet?

Returned Format:

[:CHANnel<N>:OFFSet] <value><NL>

Where:

<N> ::= 2 or 3
<value> ::= offset value in volts (exponential–NR3 format)

Example:

```
DIM Offset$[100]
OUTPUT 707;":CHANNEL2:OFFSET?"
ENTER 707;Offset$
PRINT Offset$
END
```

PROBe

Note The PROBE command can only be used with channels 2 and 3.

The :CHANNEL<N>:PROBE command specifies the probe attenuation factor for the selected channel. The range of the probe attenuation factor is from 0.9 to 1000.0. This command does not change the actual input sensitivity of the Peak Power Analyzer. It changes the reference constants for scaling the display factors and for automatic measurements, trigger levels, etc.

The PROBE query returns the current probe attenuation factor for the selected channel.

Command Syntax: :CHANnel<N>:PROBe <atten>

Where:

<N> ::= 2 or 3
<atten> ::= 0.9 to 1000

Example:

```
OUTPUT 707;":CHANNEL2:PROBE 10"
```

Query Syntax: :CHANnel<N>:PROBe?

Where:

<N> ::= 2 or 3

Returned Format:

[:CHANnel<N>:PROBe] <atten><NL>

Where:

<N> ::= 2 or 3
<atten> ::= 0.9 to 1000 (exponential–NR3 format)

Example:

```
DIM Prb$[100]
OUTPUT 707;":CHANNEL2:PROBE?"
ENTER 707;Prb$
PRINT Prb$
END
```

RANGe

The :CHANNEL<N>:RANGE command defines the full scale vertical axis of the selected channel.

In linear mode, the RANGE for channels 1 and 4 can be set to any value from 400 nW to 160 mW. In log mode, the RANGE for channels 1 and 4 can be set to any value from −33.98 dBm to 22.04 dBm. After a reset or key-down-power-up, the Analyzer defaults to linear mode (watts). If no units are specified, the Analyzer assumes the range is in watts. The :CHANNEL:FACTOR value will scale the RANGE setting by the FACTOR value.

The RANGE for channels 2 and 3 can be set to any value from 800 mV to 4.0 V, when using 1:1 probe attenuation. If the probe attenuation is changed, the range value is multiplied by the probe attenuation factor.

Note

When setting vertical sensitivity in log mode for both local and remote operation, the range value is the full scale value. However, when in linear mode, the method for setting vertical sensitivity via HP-IB differs from the method used via the front panel. When setting sensitivity over HP-IB, the full scale value is sent. When using the front panel, the per division value is entered.

The RANGE query returns the current range setting for the specified channel.

Note

If the argument is in watts, no units suffix is required. However, if the argument is in dBm, the suffix dBm is required. The Peak Power Analyzer will display the argument in the current units.

The linear and log vertical sensitivity settings for channels 1 and 4 are coupled. When the vertical setting in one mode is changed, the setting in the other mode is affected.

Command Syntax: (Channels 1 and 4)

:CHANnel<N>:RANGe <range>[W | DBM]

Where:

<N> ::= 1 or 4
<range> ::= range value

Example:

```
:OUTPUT 707;"CHANNEL1:RANGE 400N"
```

Command Syntax: (Channels 2 and 3)

:CHANnel<N>:RANGe <range>[V]

Where:

<N> ::= 2 or 3
<range> ::= range value

Example:

```
:OUTPUT 707;"CHANNEL2:RANGE 1.2"
```

Query Syntax: :CHANnel<N>:RANGe?

Returned Format:

[:CHANnel<N>:RANGe] <range><NL>

Where:

<N> ::= 1, 2, 3, or 4
<range> ::= range value in Volts, Watts, or dBm (exponential–NR3 format)

Example:

```
DIM Rng$[100]
OUTPUT 707;":CHAN1:RANGE?"
ENTER 707;Rng$
PRINT Rng$
END
```

SENSor?

The :CHANNEL<N>:SENSOR query outputs the model number, serial number, and calibration date of the peak power sensor connected to the specified channel.

Note

When Option 001, Single Sensor Input, is installed, channel 4 can't be specified. Error −360, "Missing a Baseband Board" is placed in the error queue when an attempt is made to specify channel 4.

Query Syntax: :CHANnel<N>:SENSor?

Where:

<N> ::= 1 or 4

Returned Format:

[:CHANnel<N>:SENSor]
<Model_num>,<serial_num>,<cal_date>

Where:

<N> ::= 1 or 4
<Model_num> ::= HP84812A, 84813A, or 84814A
<serial_num> ::= In the form 1234A56789
<cal_date> ::= YYMMDD (YY is the year. MM is the month. DD is the date.)

Example:

```
DIM Sensor$[100]
OUTPUT 707;":CHAN1:SENS?"
ENTER 707;Sensor$
PRINT Sensor$
END
```

SENSor:TEMPerature?

Note

The SENSOR:TEMPERATURE query can only be used with channels 1 and 4.

The :CHANNEL<N>:SENSOR:TEMPERATURE query outputs the temperature of the thermistor in the peak power sensor that is connected to the specified channel.

Query Syntax

:CHANnel<N>:SENSor:TEMPerature?

Where:

<N> ::= 1 or 4

Note

If a peak power sensor is not connected to the specified channel, or channel 4 is specified when Option 001, Single Sensor Input, is installed, +9.99999E+37 is returned in response to the query, and error −360,"Missing a Baseband Board", is placed in the error queue.

Returned Format:

[:CHANnel<N>:SENSor:TEMPerature] <Value><NL>

Where:

<Value> ::= The temperature of the thermistor in degrees Celsius. (exponential—NR3 format)

Example:

```
DIM Stemp$ [100]
OUTPUT 707;":CHAN1:SENS:TEMP?"
ENTER 707;Stemp$
PRINT Stemp$
END
```

SPAN

Note

The SPAN command is used with channels 1 and 4 when the display units are set to log.

The :CHANNEL<N>:SPAN command sets the number of dB which the vertical axis covers. The number of dB per division is set by the number of screens being displayed. The number of screens displayed is affected by the :DISPLAY:FORMAT command, :TIMEBASE:WINDOW command, and :FUNCTION subsystem.

The SPAN query returns the current number of dB which the vertical axis covers.

Command Syntax: :CHANnel<N>:SPAN {40 | 16 | 8} [DB]

Where:

<N> ::= 1 or 4

Example:

```
OUTPUT 707;":CHANNEL1:SPAN 40"
```

Query Syntax: :CHANnel<N>:SPAN?

Where:

<N> ::= 1 or 4

Returned Format:

[:CHANnel<N>:SPAN] {40 | 16 | 8}<NL>

Note

Whether in linear mode or log mode, the returned value is always in dB.

Where:

<N> ::= 1 or 4

Example:

```
DIM Span$[100]
OUTPUT 707;":CHAN1:SPAN?"
ENTER 707;Span$
PRINT Span$
END
```

TTL

Note The TTL command is only used with channels 2 and 3.

The :CHANNEL<N>:TTL command sets the vertical range, offset, channel coupling, and trigger level of the selected channel for optimum viewing of TTL signals. The offset is set to 2.5 volts. The trigger level is set to 1.4 volts and the range is set to 4.0 volts full scale. Channel coupling is set to DC.

There is no query form of this command.

Command Syntax: :CHANnel<N>:TTL

Example:

```
OUTPUT 707;":CHAN2:TTL"
```

Where:

<N> ::= 2 or 3

11

Display Subsystem

Introduction

The DISPLAY subsystem is used to control the display of data, amplitude and time markers, text, and graticules.

Note

The command that changes the Display mode is :ACQUIRE:TYPE. The command that controls the number of averages is ACQUIRE:COUNT.

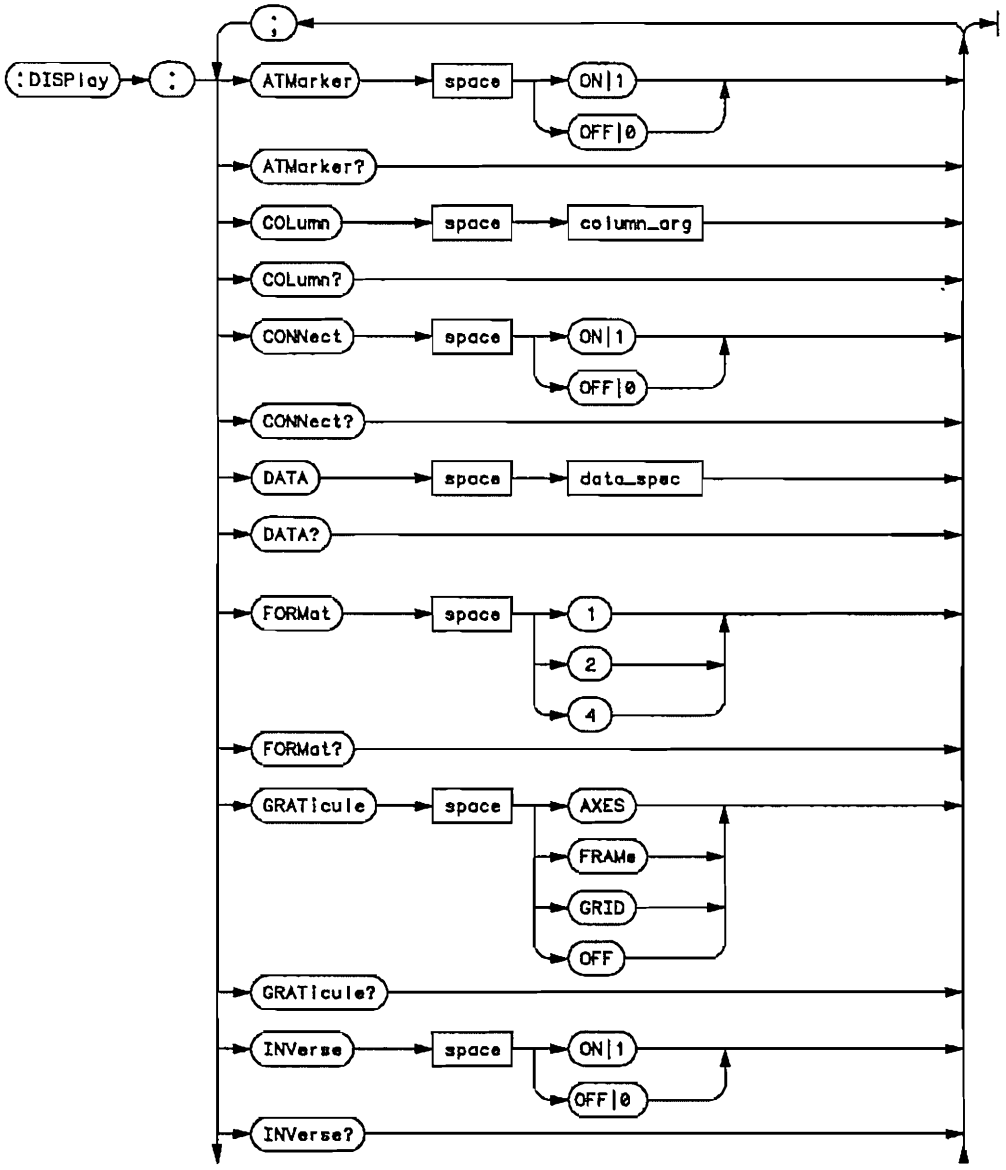


Figure 11-1. Display Subsystem Syntax Diagrams

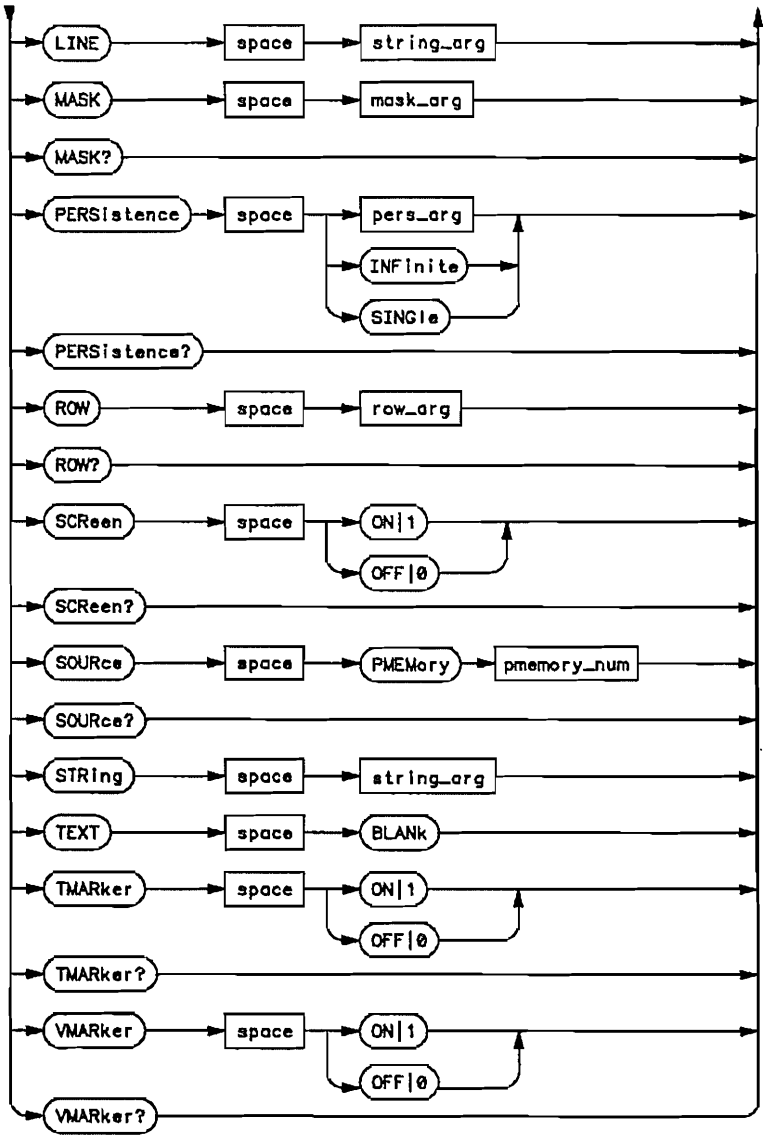


Figure 11-1. Display Subsystem Syntax Diagrams (continued)

column_arg = An integer from 0 through 72.

data_spec = Block data in IEEE 488.2 # format.

mask_arg = An integer, 0 through 255.

pers_arg = An real number 0.1 through 11.

pmemory_num = An integer, 0 through 3.

row_arg = An integer, 0 through 24.

string_arg = Any quoted string.

Figure 11-1. Display Subsystem Syntax Diagrams (continued)

ATMarker

The :DISPLAY:ATMARKER command turns the amplitude at time marker on and off.

The ATMARKER query returns the current state of the marker. The returned status is indicated by using a 1 for on and a 0 for off.

Command Syntax: :DISPlay:ATMarker {ON | 1} | {OFF | 0}

Example:

```
OUTPUT 707;":DISPLAY:ATMARKER ON"
```

Query Syntax: :DISPlay:ATMarker?

Returned Format:

[:DISPlay:ATMarker] {1 | 0} <NL>

Note

For the amplitude-at-time markers to be operational, the time markers (:DISPlay:TMARker) must be on. When the amplitude-at-time markers are operational, a positive response of the time markers (:DISPlay:TMARker?) and the amplitude-at-time markers (:DISPlay:ATMarker?) is sent from the Peak Power Analyzer to the controller.

Example:

```
DIM Atm$[100]
OUTPUT 707;":DISPLAY:ATMARKER?"
ENTER 707;Atm$
PRINT Atm$
END
```

COLumn

The :DISPLAY:COLUMN command specifies the starting column for subsequent STRING and LINE commands.

The COLUMN query returns the column where the next LINE or STRING will start.

Command Syntax: :DISPlay:COLumn <number>

Where:

<number> ::= 0 through 72

Example:

```
OUTPUT 707;":DISPLAY:COLUMN 50"
```

Query Syntax: :DISPlay:COLumn?

Returned Format:

[:DISPlay:COLumn] <number><NL>

Where:

<number> ::= 0 through 72 (integer–NR1 format)

Example:

```
DIM Clmn$[100]
OUTPUT 707;":DISPLAY:COLUMN?"
ENTER 707;Clmn$
PRINT Clmn$
END
```

CONNect

The :DISPLAY:CONNECT command turns the connect-the-dots function on and off.

The CONNECT query returns the current setting of the connect-the-dots function. The returned status is indicated by using a 1 for on and a 0 for off.

Command Syntax: :DISPlay:CONNect {{ON | 1} | {OFF | 0}}

Example:

```
OUTPUT 707;":DISPLAY:CONNECT ON"
```

Query Syntax: :DISPlay:CONNect?

Returned Format:

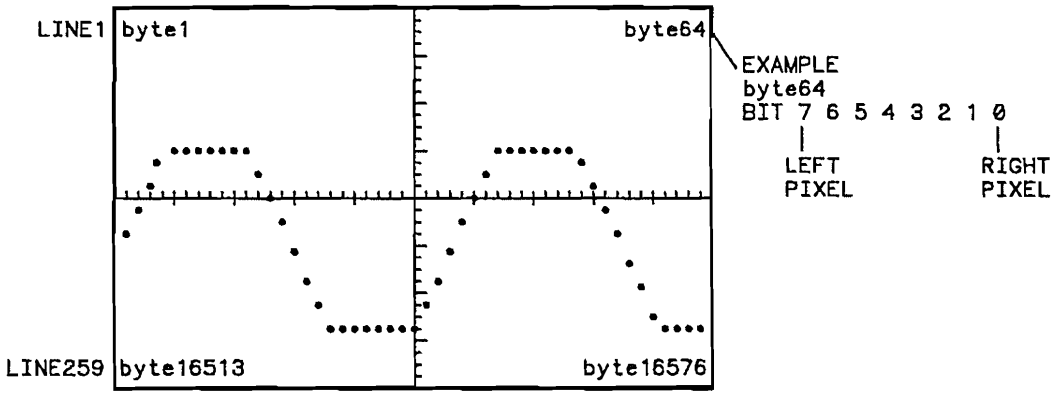
[:DISPlay:CONNect] {1 | 0}<NL>

Example:

```
DIM Cnn$[100]
OUTPUT 707;":DISP:CONNECT?"
ENTER 707;Cnn$
PRINT Cnn$
END
```

DATA

The :DISPLAY:DATA command is used to write waveform data to one of the graphic pixel planes in the Peak Power Analyzer. The DATA command is followed by a block of binary data that is transferred from the controller to a specific displayed pixel plane in the Peak Power Analyzer. Only pixel memories 1 and 2 may be written to. The pixel data is in the IEEE 488.2 definite block form with 16576 bytes of data preceded by seven block header bytes. The 16576 bytes consist of 259 lines of data, and each line contains 64 bytes. Each byte consists of 8 pixels, where bit 7 is the left pixel and bit 0 is the right pixel. Bytes are counted top to bottom, left to right. Refer to the illustration below for more information. The block header contains the ASCII characters "#800016576".



The :DISPLAY:DATA query is used to read waveform data from one of the pixel planes in the Peak Power Analyzer. The pixel planes available are planes 0 through 3. The DATA query causes the Peak Power

Analyzer to output pixel data from the specified plane. If plane 0 is specified, the Peak Power Analyzer will transfer the active display. If PMEMORY1 or PMEMORY2 is specified, that memory will be transferred. When PMEMORY3 is specified, the half-bright portion of the display (graticule, markers, and displayed memories) will be transferred.

Note

The pixel planes are specified by the :DISPLAY:SOURCE command with PMEMORY0 through PMEMORY3.

Command Syntax: :DISPlay:DATA <binary block>

Where:

<binary block> ::= block data in IEEE 488.2 # format

Query Syntax: :DISPlay:DATA?

Returned Format:

[:DISPlay:DATA] #800016576<16576 bytes of binary data><NL>

Example:

```
10   CLEAR 707
20   DIM Plane$ [17000]
30   OUTPUT 707;":SYST:HEAD ON;:EOI ON"
40   OUTPUT 707;":DISPLAY:SOURCE PMEMORY0;DATA?"
50   ENTER 707 USING "-K";Plane$
60   OUTPUT 707;":DISPLAY:SOURCE PMEM1"
70   OUTPUT 707 USING"#,-K";Plane$
80   END
```

This example transfers data from the active display memory to the controller, and then, transfers the data back to pixel memory 1 in the Peak Power Analyzer .

FORMat

The :DISPLAY:FORMAT command sets the number of display areas on the CRT. FORMAT 1 provides one display area and uses eight divisions for the full scale range. FORMAT 2 sets the number of screens to 2 and uses four divisions for the full scale range. FORMAT 4 provides four display areas and uses two divisions for the full scale range.

The FORMAT query returns the current display format.

Command Syntax: :DISPlay:FORMat {1 | 2 | 4}

Example:

```
OUTPUT 707;":DISP:FORMAT 1"
```

Query Syntax: :DISPlay:FORMat?

Returned Format:

[:DISPlay:FORMat] {1 | 2 | 4}<NL>

Example:

```
DIM Frmt$[100]
OUTPUT 707;":DISPLAY:FORMAT?"
ENTER 707;Frmt$
PRINT Frmt$
END
```

GRATicule

The :DISPLAY:GRATICULE command selects the type of graticule that is displayed.

The GRATICULE query returns the type of graticule displayed.

Command Syntax: :DISPlay:GRATicule {OFF | GRID | AXES | FRAMe}

Example:

```
OUTPUT 707;":DISPLAY:GRATICULE AXES"
```

Query Syntax: :DISPlay:GRATicule?

Returned Format:

[:DISPlay:GRATicule] <type><NL>

Where:

<type> ::= {OFF | GRID | AXES | FRAMe}

Example:

```
DIM Grat$[100]
OUTPUT 707;":DISPLAY:GRATICULE?"
ENTER 707;Grt$
PRINT Grt$
END
```

INVerse

The :DISPLAY:INVERSE command determines whether text sent with the :DISPLAY:LINE or :DISPLAY:STRING command is to be written with the inverse attribute. If the inverse attribute is on, the text will be written in inverse video.

The INVERSE query returns the current state of this command.

Command Syntax: :DISPlay:INVerse {{ON | 1} | {OFF | 0}}

Example:

```
OUTPUT 707;":DISPLAY:INVERSE OFF"
```

Query Syntax: :DISPlay:INVerse?

Returned format:

[:DISPlay:INVerse] {1 | 0}<NL>

Example:

```
DIM Iv$[100]
OUTPUT 707;":DISP:INVERSE?"
ENTER 707;Iv$
PRINT Iv$
END
```

LINE

The :DISPLAY:LINE command writes a text string to the screen. The text is displayed starting at the location of the current row and column. The row and column are set by the :DISPLAY:ROW and :DISPLAY:COLUMN commands prior to sending the :DISPLAY:LINE command. Text can be written over the entire screen with the LINE command.

If the text string is longer than the available space to the right of the starting column, defined by the COLUMN command, the text will wrap to column zero of the same row where the text began. The ROW value is then incremented by one, but, the COLUMN value remains the same. The next :DISPLAY:LINE command writes on the next row starting at the same column as the previous text. After writing line 24, the last line in the display area, the ROW is reset to 0.

Note

Text written to the display is removed in one of the following ways:

- Save the front panel settings before writing to the display. Remove the text by recalling the saved front panel settings. Common commands *SAV (Save) and *RCL (Recall) are used for this function.
- Using the LINE command, write blank spaces to the display at the same row and column that the text was written to.
- Use common command *RST (reset). Default parameters are recalled.

Command Syntax: :DISPlay:LINE <quoted string>

Where:

<quoted string> ::= any series of ASCII characters enclosed in double quotes. The quotes are not displayed.

Example:

```
OUTPUT 707;":DISPLAY:LINE ""ENTER PROBE ATTENUATION"""
```

MASK

The :DISPLAY:MASK command inhibits the Peak Power Analyzer from writing to selected areas of the screen. The purpose of the command is to allow HP-IB text to be written anywhere on the screen and to prevent the Peak Power Analyzer from overwriting the text through its normal operation.

The MASK parameter is an 8-bit integer in which each bit controls writing to an area of the screen. A zero inhibits writing to the area represented by the bit, and a one enables writing to that area.

Text sent over HP-IB with the DISPLAY:LINE and DISPLAY:STRING commands, or the SYSTEM:DSP command is not affected by this command.

The MASK query returns the current value of the MASK.

Note 

The Peak Power Analyzer always forces bit three of the mask high (1).

Command Syntax: :DISPlay:MASK <value>

Where:

<value> ::= 0 to 255 (integer–NR1 format)

Example:

```
OUTPUT 707;":DISPLAY:MASK 67"
```

The previous example enables writing to the Menu Area (bit 6), the Status Line (bit 1), and the Advisory Area (bit 0).

Query Syntax: :DISPlay:MASK?

Return Format:

[:DISPlay:MASK] <value><NL>

Where:

<value> ::= 0 to 255 (integer–NR1 format)

Example:

```
DIM Msk$[100]
OUTPUT 707;":DISP:MASK?"
ENTER 707;Msk$
PRINT Msk$
END
```

Table 11-1. Display Mask Byte

Bit	Weight	Screen Area Effected
7	128	unused
6	64	Menu Area
5	32	Timebase Information
4	16	Measurement Result Area
3	8	Graticule Area
2	4	unused
1	2	Status Line
0	1	Advisory Area

PERSistence

The :DISPLAY:PERSISTENCE command sets the display persistence. The PERSISTENCE command is only effective in the Normal display mode.

The parameters for this command are INFINITE, SINGLE, or a real number from 0.15 through 11.0 representing the persistence in seconds. Any value less than 0.15 will set the PERSISTENCE to MINIMUM. Any value greater than 10 seconds will set the PERSISTENCE to INFINITE.

When SINGLE is sent as the argument for this command, the persistence value is set to MINIMUM.

The PERSISTENCE query returns the current persistence value. When MINIMUM persistence is displayed, the value returned is 0. When INFINITE persistence is displayed, the value returned is 11.

Command Syntax: :DISPlay:PERSistence {INFinite | SINGle | 0.1 through 11}

Example:

```
OUTPUT 707;":DISPLAY:PERSISTENCE 3.0"
```

Query Syntax: :DISPlay:PERSistence?

Returned Format:

[:DISPlay:PERSistence] <value><NL>

Where:

<value> ::= {0 | .2–10 | 11} (exponential–NR3 format)

Example:

```
DIM Prs$[100]
OUTPUT 707;":DISPLAY:PERSISTENCE?"
ENTER 707;Prs$
PRINT Prs$
END
```

ROW

The :DISPLAY:ROW command specifies the starting row on the display for subsequent :DISPLAY:STRING and :DISPLAY:LINE commands. The ROW number remains constant until another ROW command is received, or it is incremented by the LINE command. The ROW value is 0 through 24.

The ROW query returns the current ROW value.

Command Syntax: :DISPlay:ROW <row number>

Where:

<row number>::= 0 through 24

Example:

```
OUTPUT 707;":DISPLAY:ROW 10"
```

Query Syntax: :DISPLAY:ROW?

Returned Format:

[:DISPlay:ROW] <row number><NL>

Where:

<row number> ::= 0 through 24 (integer–NR1 format)

Example:

```
DIM Rw$[100]
OUTPUT 707;":DISPLAY:ROW?"
ENTER 707;Rw$
PRINT Rw$
END
```

SCReen

The :DISPLAY:SCREEN command turns the displayed screen on and off. The only part of the screen that remains on after the :DISPLAY:SCREEN OFF command is executed is the status line. The screen can be turned on again with the ON parameter.

The SCREEN query returns the current setting of this function. The returned status is indicated by using a 1 for on and a 0 for off.

The command :DISPLAY:TEXT BLANK removes only the text from the display.

Note Setting :SCREEN to OFF increases the speed of the Peak Power Analyzer.

Command Syntax: :DISPlay:SCReen {{ON | 1} | {OFF | 0}}

Example:

```
OUTPUT 707;":DISPLAY:SCREEN ON"
```

Query Syntax: :DISPlay:SCReen?

Returned Format:

[:DISPlay:SCReen] {1 | 0}<NL>

Example:

```
DIM Srn$[100]
OUTPUT 707;":DISP:SCREEN?"
ENTER 707;Srn$
PRINT Srn$
END
```

SOURce

The :DISPLAY:SOURCE command specifies the source or destination for the :DISPLAY:DATA query and command. The SOURCE command has one parameter, PMEMORY0 through PMEMORY3.

The SOURCE query returns the currently specified SOURCE.

Command Syntax: :DISPlay:SOURce PMEMory{0 | 1 | 2 | 3}

Where:

PMEMory0 ::= active display
PMEMory1 ::= pixel memory 1
PMEMory2 ::= pixel memory 2
PMEMory3 ::= half-bright portion of the display (graticule, markers, and displayed memories)

Example:

```
10   CLEAR 707
20   DIM Plane$ [17000]
30   OUTPUT 707;":SYST:HEAD ON;:EOI ON
40   OUTPUT 707;":DISPLAY:SOURCE PMEMORY0;DATA?"
50   ENTER 707 USING "-K";Plane$
60   OUTPUT 707;":DISPLAY:SOURCE PMEM1"
70   OUTPUT 707 USING "#,-K";Plane$
80   END
```

This example transfers data from the active display to the controller and then back to pixel memory 1 in the Peak Power Analyzer.

Query Syntax: :DISPlay:SOURce?

Returned Format:

[:DISPlay:SOURce] PMEMory{0 | 1 | 2 | 3}<NL>

Example:

```
DIM Src$[100]
OUTPUT 707;":DISP:SOUR?"
ENTER 707;Src$
PRINT Src$
END
```

STRing

The :DISPLAY:STRING command writes a text string to the display of the Peak Power Analyzer . The text is written starting at the current :DISPLAY:ROW and :DISPLAY:COLUMN values. If the column limit is reached (column 72), the excess text is written over the text on the left side of that line. If 90 or more characters are sent, an error is produced. The :DISPLAY:LINE command does increment the ROW value, the STRING command does not.

Note

Text written to the display is removed in one of the following ways:

- Save the front panel settings before writing to the display. Remove the text by recalling the saved front panel settings. Common commands *SAV (Save) and *RCL (Recall) are used for this function.
- Using the STRING command, write blank spaces to the display at the same row and column that the text was written to.
- Use common command *RST (reset). Default parameters are recalled.

Command Syntax: :DISPlay:STRing <quoted string>

Example:

```
OUTPUT 707;":DISP:STRING ""INPUT SIGNAL TO CHANNEL 2"""
```

TEXT

The :DISPLAY:TEXT command allows you to blank the user text area on the display. All text on the entire screen is blanked. This command has only one parameter. The text is displayed again by using the :DISPLAY:SCREEN command.

There is no query form of this command.

Command Syntax: :DISPlay:TEXT BLANk

Example:

```
OUTPUT 707;":DISPLAY:TEXT BLAN"
```

TMARker

The :DISPLAY:TMARKER command turns the time markers on and off.

The TMARKER query returns the state of the time markers.

Command Syntax: :DISPlay:TMARker {{ON | 1} | {OFF | 0}}

Example:

```
OUTPUT 707;":DISP:TMAR OFF"
```

Query Syntax: :DISPlay:TMARker?

Returned Format:

[:DISPlay:TMARker] {1 | 0}<NL>

Example:

```
DIM Tmr$[100]
OUTPUT 707;":DISP:TMARKER?"
ENTER 707;Tmr$
PRINT Tmr$
END
```

Note It is a recommended practice to turn the time markers on before attempting to set them using :MEASURE:TSTART or MEASURE:TSTOP. If TMARKER is not set to on, the markers will not be visible.

VMARker

The :DISPLAY:VMARKER command turns the vertical amplitude markers on and off.

The VMARKER query returns the state of the amplitude markers.

Command Syntax: :DISPlay:VMARker {{ON | 1} | {OFF | 0}}

Example:

```
OUTPUT 707;":DISP:VMARKER ON"
```

Query Syntax: :DISPlay:VMARker?

Returned Format:

[:DISPlay:VMARker] {1 | 0}<NL>

Example:

```
DIM Vmrk$[100]
OUTPUT 707;":DISP:VMARKER?"
ENTER 707;Vmrk$
PRINT Vmrk$
END
```

Note 

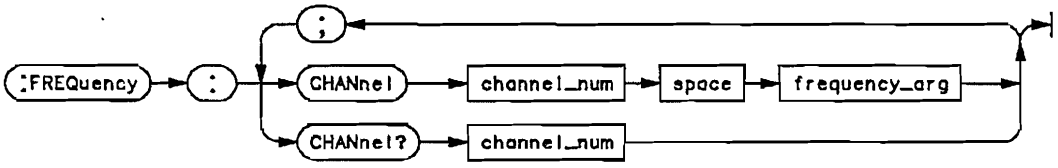
It is a recommended practice to turn the vertical amplitude markers on before attempting to set them using :MEASURE:TSTART or MEASURE:TSTOP. If VMARKER is not set to on, the markers will not be visible.

12

Frequency Subsystem

Introduction

The Frequency Subsystem has only one command. This subsystem is used to specify the carrier frequency of the CW/pulsed source being measured. The carrier frequency is used by the Peak Power Analyzer to look up calibration data which is stored in the peak power sensor. Specifying this parameter is necessary to make accurate power measurements.



channel_num= 1 or 4

frequency_arg= 1 MHz to 200 GHz

Figure 12-1. Frequency Subsystem Syntax Diagram

CHANnel

The :FREQUENCY:CHANNEL command is used to specify the carrier frequency of the source being measured. Valid frequency entries are 1 MHz to 200 GHz. The frequency range which the Peak Power Analyzer will accept is sensor dependent.

The CHANNEL query outputs the current carrier frequency for the specified channel.

Note The front panel Carrier Frequency menu is set to "ch1 ≠ ch4" whenever one or both channels are programmed remotely. To set both channels to the same frequency, the controller must program then separately.

Command Syntax: :FREQuency:CHANnel<number> <value>

Note When using mega hertz as a suffix, use the abbreviation MAHZ>

Where:
<number> ::= 1 or 4
<value> ::= 1 MHz to 200 GHz

Example:

```
OUTPUT 707;":FREQUENCY:CHANNEL1  10E09"
```

Query Syntax: :FREQuency:CHANnel<number>?

Where:
<number> ::= 1 or 4

Returned Format

[:FREQuency:CHANnel<number>]<value>

Where:
<value> ::= 1 MHz to 200 GHz (exponential–NR3 format)

Example:

```
DIM Freq$ [100]
OUTPUT 707;":FREQ:CHAN1?"
ENTER 707;Freq$
PRINT Freq$
END
```

13

Function Subsystem

Introduction

The FUNCTION subsystem defines five functions using the displayed channels and/or the waveform memories as operands.

A function is generated by mathematically manipulating one or two operands with known operations. The mathematical operations employed by the Peak Power Analyzer are: ADD, SUBTRACT, DIVIDE, VERSUS, and ONLY.

When the function is calculated, it can be used for the following:

- Display
- Evaluation with the measurement features
- Storage in memory
- Output over HP-IB

See Figure 13-1 for a syntax diagram of the function subsystem commands.

Channels 1 through 4 and waveform Memories 1 through 4 are available for functions.

Note

A function operates on linear quantities. And the displayed function is always in linear units.

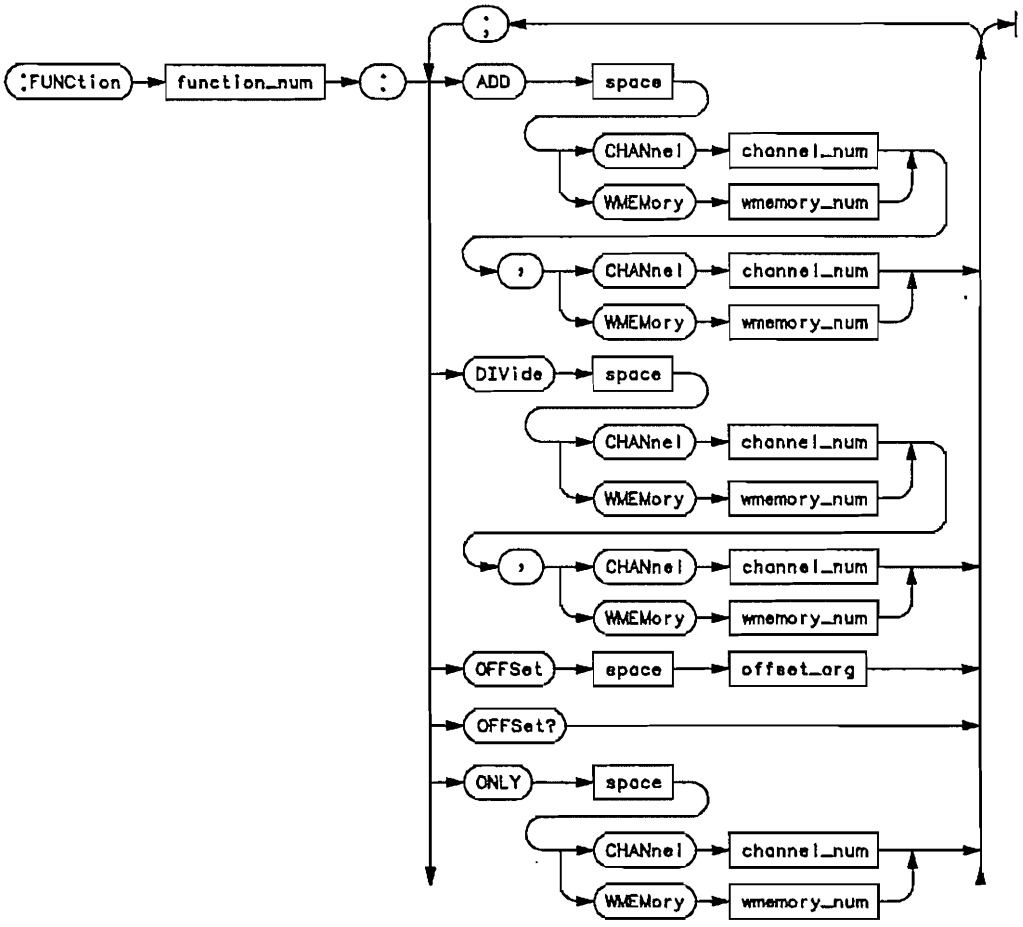
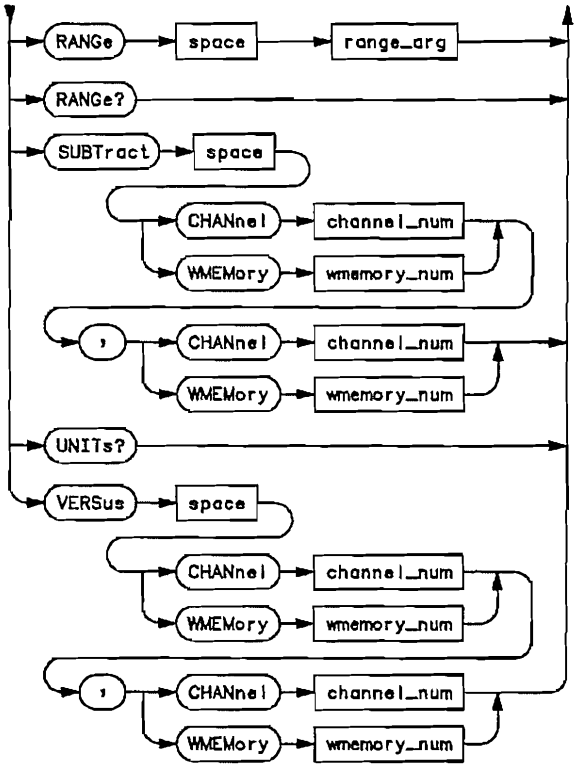


Figure 13-1. Function Subsystem Syntax Diagrams



channel_num = 1, 2, 3, or 4
function_num = 1 or 2
offset_arg = Shifts amplitude of function.
range_arg = Full scale vertical axis.
wmemory_num = 1, 2, 3, or 4

Figure 13-1. Function Subsystem Syntax Diagrams (continued)

ADD

The :FUNCTION<N>:ADD command algebraically sums the two defined operands.

Note

When Option 001, Single Sensor Input, is installed, channel 4 can't be specified. Error −360, "Missing a Baseband Board" is placed in the error queue when an attempt is made to specify channel 4.

Command Syntax: :FUNCtion<N>:ADD <operand>,<operand>

Where:

<N> ::= 1 or 2
<operand> ::= {CHANnel1 | CHANnel2 | CHANnel3 | CHANnel4 | WMEMory1 | WMEMory2 | WMEMory3 | WMEMory4}

Example:

```
OUTPUT 707;":FUNCTION2:ADD WMEMORY3,WMEMORY4"
```

DIVide

The :FUNCTION<N>:DIVIDE command algebraically divides the two operands.

Note

When the divisor is a small number or zero, the waveform may be clipped or not displayed at all.

If Option 001, Single Sensor Input, is installed, channel 4 can't be specified. Error −360, "Missing a Baseband Board" is placed in the error queue when an attempt is made to specify channel 4.

Command Syntax: :FUNCtion <N>:DIVide <operand>, <operand>

Where:

<N> ::= 1 or 2
<operand> ::= {CHANnel1 | CHANnel2 | CHANnel3 | CHANnel4 | WMEMory1 | WMEMory2 | WMEMory3 | WMEMory4}

Example:

```
OUTPUT 707;":FUNCTION2:DIVIDE CHANNEL1,CHANNEL4"
```

OFFSet

The :FUNCTION<N>:OFFSET command shifts the amplitude for the selected function.

The OFFSET query returns the current offset value for the selected function.

Note

When a function consists of one or two power operands and the offset is set so the bottom of the function is clipped, an error may occur when a measurement is made on the function.

Command Syntax: :FUNCtion<N>:OFFSet <offset>

Where:

<N> ::= 1 or 2
<offset> ::= offset value (The offset does not require a suffix, it uses the current units.

Example:

```
OUTPUT 707;":FUNCTION1:OFFSET 650E-4"
```

Query Syntax: :FUNCtion<N>:OFFSet?

Where:

<N> ::= 1 or 2

Returned Format:

[:FUNCtion<N>:OFFSet] <offset><NL>

Where:

<N> ::= 1 or 2
<offset> ::= offset value (see above)
(exponential–NR3 format)

Example:

```
DIM Off$[100]
OUTPUT 707;":FUNCTION2:OFFSET?"
ENTER 707;Off$
PRINT Off$
END
```

ONLY

The :FUNCTION<N>:ONLY command produces a copy of the operand. The ONLY command is useful when scaling channels and memories with the :FUNCTION<N>:RANGE and :FUNCTION<N>:OFFSET commands to settings that are not achievable in the channel menu.

Note

When Option 001, Single Sensor Input, is installed, channel 4 can't be specified. Error −360, "Missing a Baseband Board" is placed in the error queue when an attempt is made to specify channel 4.

Command Syntax: :FUNCtion<N>:ONLY<operand>

Where:

<N> ::= 1 or 2
<operand> ::= {CHANnel1 | CHANnel2 | CHANnel3 | CHANnel4 | WMEMory1 | WMEMory2 | WMEMory3 | WMEMory4}

Example:

```
OUTPUT 707;":FUNCTION2:ONLY WMEMORY4"
```

RANGe

The :FUNCTION<N>:RANGE command defines the full scale vertical axis of the selected function.

The RANGE query returns the current range setting for the specified function.

Command Syntax: :FUNCtion<N>:RANGe <range>

Where:

<N> ::= 1 or 2
<range> ::= amplitude value (Range does not require a suffix, it uses the current units.

Example:

```
OUTPUT 707;":FUNCTION2:RANGE .260"
```

Query Syntax: :FUNCtion<N>:RANGe?

Where:

<N> ::= 1 or 2

Returned Format:

[:FUNCtion<N>:RANGe] <range><NL>

Where:

<N> ::= 1 or 2
<range> ::= current range setting
(exponential–NR3 format)

Example:

```
DIM Rng$[100]
OUTPUT 707;":FUNCTION2:RANGE?"
ENTER 707;Rng$
PRINT Rng$
END
```

SUBTract

The :FUNCTION<N>:SUBTRACT command algebraically subtracts operand 2 from operand 1.

Note

When Option 001, Single Sensor Input, is installed, channel 4 can't be specified. Error −360, "Missing a Baseband Board" is placed in the error queue when an attempt is made to specify channel 4.

Command Syntax: :FUNCtion<N>:SUBTract <operand>,<operand>

Where:

<N> ::= 1 or 2
<operand> ::= {CHANnel1 | CHANnel2 | CHANnel3 | CHANnel4 | WMEMory1 | WMEMory2 | WMEMory3 | WMEMory4}

Example:

```
OUTPUT 707;":FUNCTION2:SUBTRACT WMEMORY3,WMEMORY2"
```

In this example Waveform Memory 2 would be algebraically subtracted from Waveform Memory 3.

UNITs?

The :FUNCTION<N>:UNITS? query allows the user to query the Peak Power Analyzer to determine the current units for the selected function. The query will only return the fundamental units. Any suffix multipliers, such as, nano, micro, kilo, or milli will not be returned. If the function results in undefined units, such as, dBm/Volts, UDF is returned.

Query Syntax: :FUNCtion<N>:UNITs?

Where:

<N> ::= 1 or 2

Returned Format:

[:FUNCtion<N>:UNITs]<units><NL>

Where:

<N> ::= 1 or 2
<units> ::= {W | UDF | W/V | V/W | V | W/W | V/V}

Example:

```
DIM Unit$[100]
OUTPUT 707;":FUNCTION2:UNITs?"
ENTER 707;Unit$
PRINT Unit$
END
```

VERSus

The :FUNCTION<N>:VERSUS command allows X versus Y displays with two operands. The first operand defines the Y axis and the second defines the X axis.

The Y axis range and offset are initially equal to that of the first operand and can be adjusted with the FUNCTION<N>:RANGE and FUNCTION<N>:OFFSET commands.

The X axis range and offset are always equal to that of the second operand. It can only be altered by changing the vertical settings of the second operand. This also changes the Y axis vertical sensitivity and offset.

Note

The following features do not work with a VERSUS function:

- Waveform Measurements
- Waveform memory
- Ampl@time

When Option 001, Single Sensor Input, is installed, channel 4 can't be specified. Error −360, "Missing a Baseband Board" is placed in the error queue when an attempt is made to specify channel 4.

Command Syntax: :FUNCtion<N>:VERSus <Y_operand>,<X_operand>
Where:

<N> ::= 1 or 2

<Y_operand> ::= {CHANnel1 | CHANnel2 | CHANnel3 | CHANnel4 | WMEMory1 | WMEMory2 | WMEMory3 | WMEMory4}

<X_operand> ::= {CHANnel1 | CHANnel2 | CHANnel3 | CHANnel4 | WMEMory1 | WMEMory2 | WMEMory3 | WMEMory4}

Example:

```
OUTPUT 707;":FUNCTION2:VERSUS CHAN1,CHAN4"
```

14

Hardcopy Subsystem

Introduction

The HARDCOPY subsystem commands set various parameters for printing waveforms from the Peak Power Analyzer. Everything on the display is printed when the root level command PRINT is sent.

The portion of the waveform to be copied must be placed on the display.

To actually make the hardcopy print, refer to the root level command :PRINT for the sequence of commands that actually get the data to the printer.

Refer to Figure 14-1 for the syntax diagrams of the HARDCOPY subsystem commands.

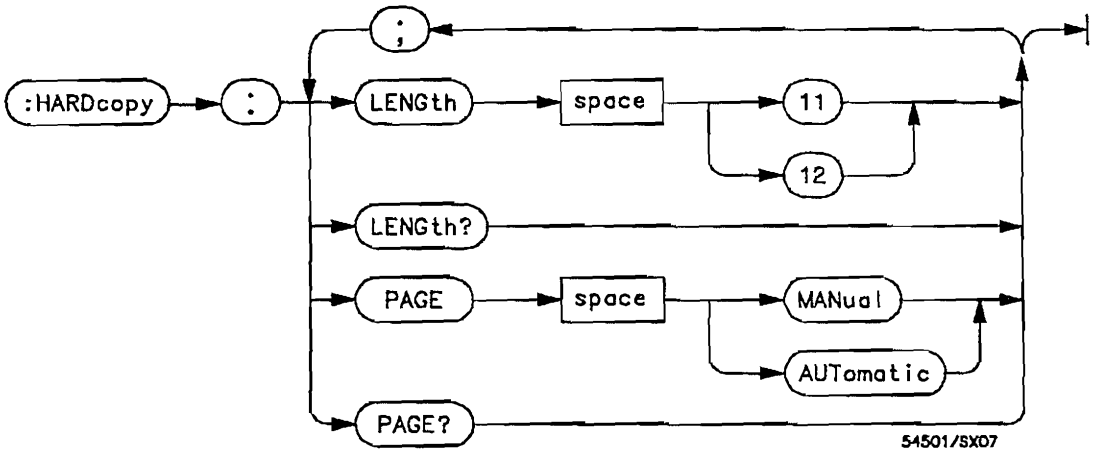


Figure 14-1. Hardcopy Subsystem Syntax Diagrams

LENGth

The :HARDCOPY:LENGTH command sets the length of the page to either 11 inches or 12 inches.

The LENGTH query returns the current length setting.

Command Syntax: :HARDcopy:LENGth {11 | 12}

Example:

```
OUTPUT 707;":HARDCOPY:LENGTH 12"
```

Query Syntax: :HARDcopy:LENGth?

Returned Format:

[:HARDcopy:LENGth] {11 | 12}<NL>

Example:

```
DIM Lgth$[100]
OUTPUT 707;":HARDCOPY:LENGTH?"
ENTER 707;Lgth$
PRINT Lgth$
END
```

PAGE

The :HARDCOPY:PAGE command enables the Peak Power Analyzer to send a formfeed to the printer after a hardcopy output.

If the PAGE command is set to AUTomatic, a formfeed occurs at the end of the hardcopy; otherwise, the page scrolls up by four lines.

The PAGE query returns the current state of the page command.

Command Syntax: :HARDcopy:PAGE {MANual | AUTomatic}

Example:

```
OUTPUT 707;":HARD:PAGE AUT"
```

Query Syntax: :HARDcopy:PAGE?

Returned Format:

[:HARDcopy:PAGE] {MANual | AUTomatic}<NL>

Example:

```
DIM Pg$[100]
OUTPUT 707;":HARDCOPY:PAGE?"
ENTER 707;Pg$
PRINT Pg$
END
```

15

Measure Subsystem

Introduction

The commands in the MEASURE subsystem are used to make parametric measurements on displayed waveforms and to report the settings of the amplitude and time markers. Some commands in this subsystem can be used to set the amplitude and time markers to specified amplitudes, times, or events.

Faster Automatic Measurements

The Peak Power Analyzer can be set to make faster automatic measurements. Three measurements are capable of the faster measurement speed:

- VERTical–This query makes an average power measurement at a random vertical point on an untriggered signal.
- VMAX–This query measures and outputs the absolute maximum amplitude present on the selected waveform.
- VTIMe–This query returns the amplitude at a specified time.

When the AUTodig command is used with these measurements, the measurements are performed faster than they normally would operate. AUTodig will only speed up these three measurements. For more information, refer to AUTodig and the individual measurement commands found in this section.

Measurement Setup

To make a measurement, the portion of the waveform required for that measurement must first be displayed on the Peak Power Analyzer. That is:

- For a PRI (Pulse Repetition Interval) or PRF (Pulse Repetition Frequency) measurement, at least two consecutive rising or falling edges must be displayed.
- For a pulse width measurement, the entire pulse must be displayed.
- For a risetime measurement, the leading (positive-going) edge of the waveform must be displayed.
- For a falltime measurement, the trailing (negative-going) edge of the waveform must be displayed.

Note

When the window function in the Timebase Subsystem is turned on, measurements are ONLY applied to the windowed portion of the waveform.

User-Defined Measurements

When User-Defined Measurements are made, the defined parameters must be set before actually sending the measurement command or query.

In User-Defined mode, when the lower threshold value is less than the upper threshold value, the mid threshold is the mid point between the upper and lower threshold .

Measurement Error

If a measurement cannot be made, the value returned for that parameter is +9.99999E+37. This is an error value that is output when a measurement cannot be made. Typically, this occurs because the proper portion of the waveform is not displayed or is clipped. The error is also returned when one of the following Utility Menus is displayed when a measurement is attempted:

- Selftest Menu
- Probe Cal Menu
- Instrument Cal Menu
- Service Menu
- Show Status Menu

Using the MENU command to select a menu other than one of the Utility Menus should allow the measurement to be made.

If the result of a power measurement is too small to be represented as a dBm value, the value returned for the measurement is −9.99999E+37. When −9.99999E+37 is returned, −99 dBm is the result output to the front panel.

If the horizontal scaling is questionable, an "error 11" is placed in the error queue. In this case, the value returned is the most accurate that can be made using the current scaling. You might be able to obtain a more accurate measurement by rescaling the horizontal to obtain more data points on the edge.

Measurement Procedure

To maintain specified measurement accuracy, the following procedure must be used when making a measurement:

Note

Before a measurement is made, a one hour warm-up period is required.

- **Setup** the parameters for the measurement.
- **Acquire** the data.
- **Query** the Peak Power Analyzer.

First, setup the Peak Power Analyzer for the measurement. Setup might include specifying the vertical sensitivity, timebase, and triggering. Autoscale is part of setup.

Second, acquire the data using the DIGITIZE command. If a setup parameter is changed after the data is acquired, the data must be acquired again to reflect the setup change.

Finally, use the query form of the measurement, and have the controller read the measurement results.

Making Measurements

If more than one waveform, edge, or pulse is displayed, time measurements are made on the first (left-most) portion of the displayed waveform that can be used. When any of the defined measurements are requested, the Peak Power Analyzer first determines the top (100%) and base (0%) linear amplitudes of the waveform. From this information, it can determine the other important linear amplitude values (10%, 90%, and 50%) for making the measurements.

The 10% and 90% linear amplitude values are used in the risetime and falltime measurements when standard measurements are selected. The 50% linear amplitude value is used for measuring PRF, pulse width, offtime, and duty cycle with standard measurements selected.

Note

The linear values are used for channels 1 and 4, in linear mode and for channels 2 and 3. However, for channels 1 and 4 in log mode, the Peak Power Analyzer converts the linear values to dBm.

The measurements can also be made using user defined parameters instead of the standard measurement values.

When the command form of a measurement is used, the Peak Power Analyzer is placed into the continuous measurement mode. When the query form of the measurement is used, continuous measurement mode is turned off. Each specified measurement is made once, and the measurement result is returned.

Except for measurements VAVERAGE and VRMS, amplitude measurements are made using the entire display. Therefore, if you want to make a measurement on a particular cycle, display only that cycle on the screen.

All voltage values are returned in volts. Returned voltage values are measured with zero volts as the reference. The value returned for the VDELTA? query is the difference of amplitude Marker 2 minus amplitude Marker 1 in volts.

All time values are returned in seconds. Returned time values are measured with the trigger point (time 0) as the reference. The value returned for TDELTA? is the time difference between the stop and start markers.

Except for VDELTA, all power values are returned as dBm or watts depending on the display mode. In log mode, VDELTA returns a ratio in dB.

Measurements are made on the displayed waveform(s) specified by the SOURCE command. The SOURCE command allows two sources to be specified. When two sources are specified, amplitude Marker 1 is assigned to the first specified source and amplitude Marker 2 is assigned to the second specified source. VDELTA is the only measurement that uses two sources.

Most measurements can only be made on a single source. If one of these measurements is made with two specified sources, the measurement is made on the first source specified.

More information about measurement algorithms can be found in Appendix A.

Refer to Figure 15-1 for the MEASURE subsystem syntax diagrams .

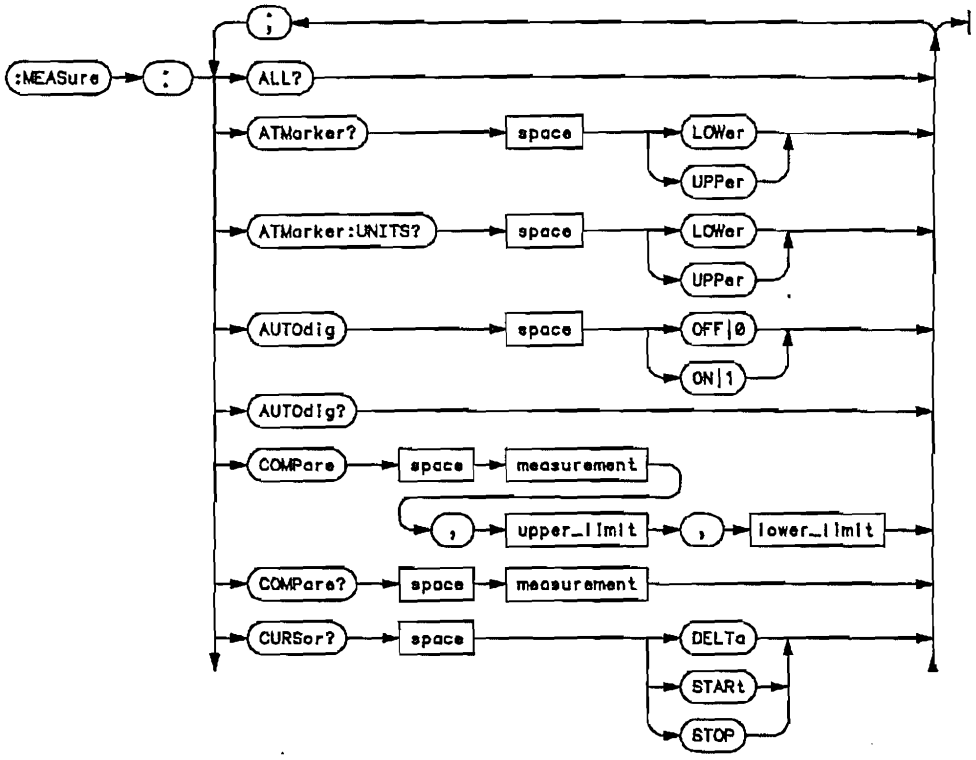


Figure 15-1. Measure Subsystem Syntax Diagrams

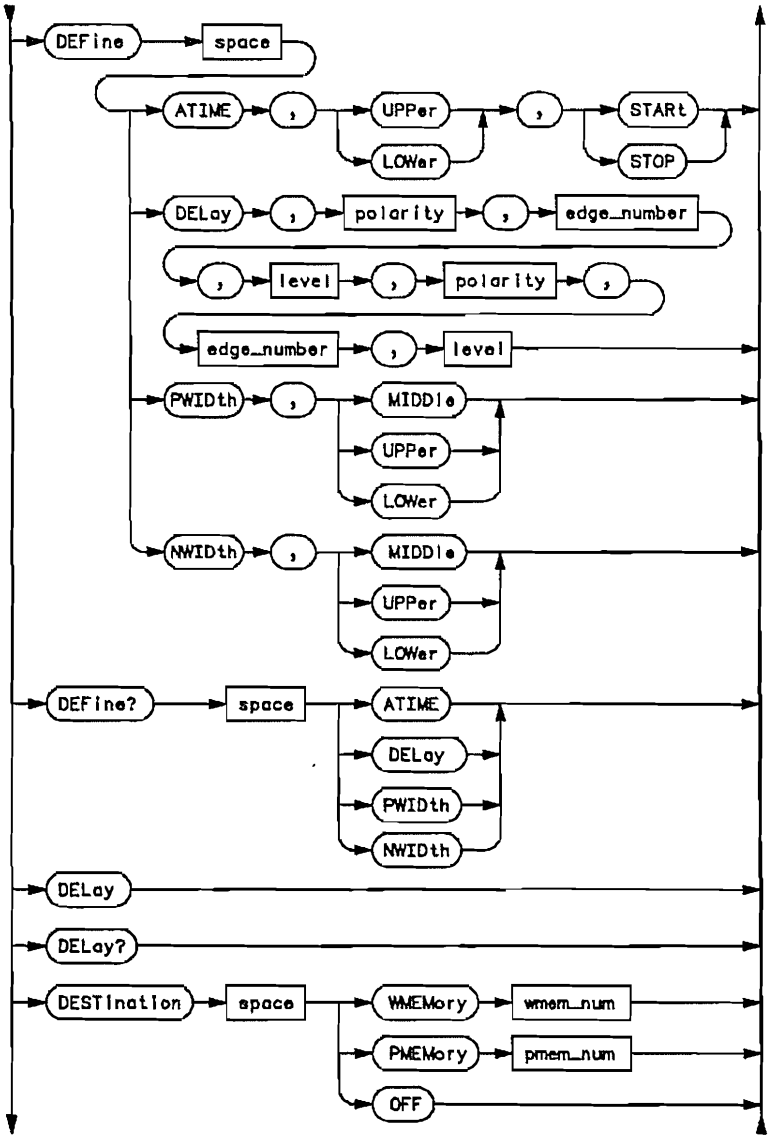


Figure 15-1. Measure Subsystem Syntax Diagrams (continued)

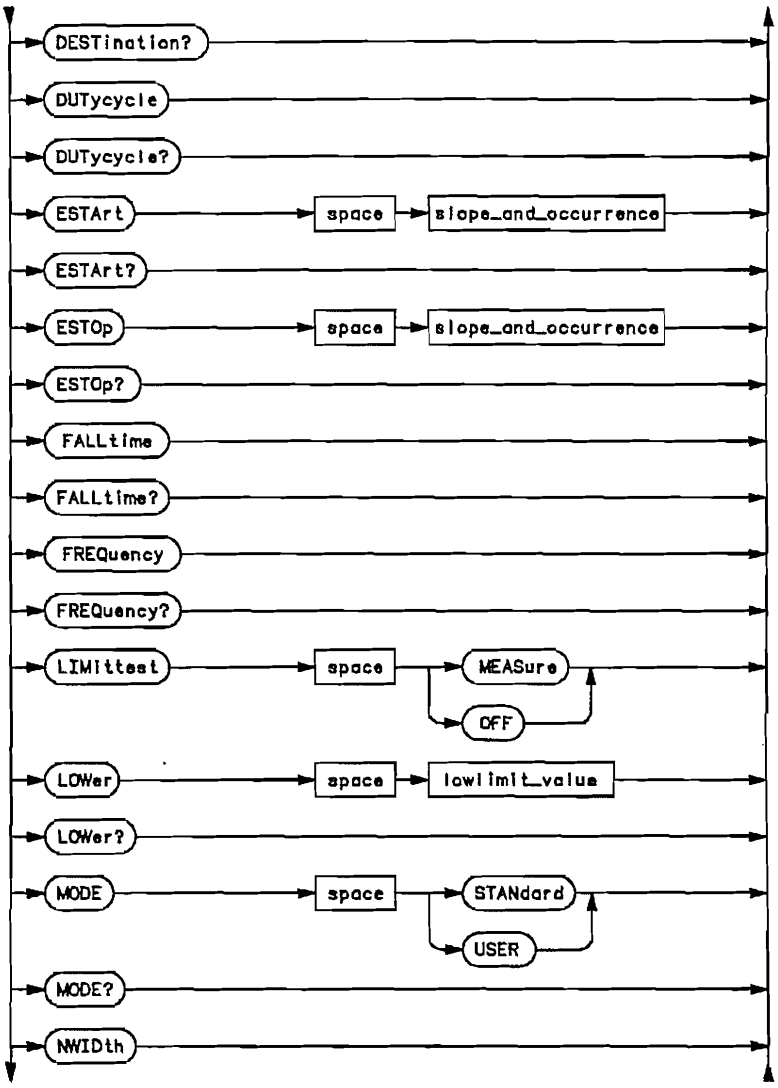


Figure 15-1. Measure Subsystem Syntax Diagrams (continued)

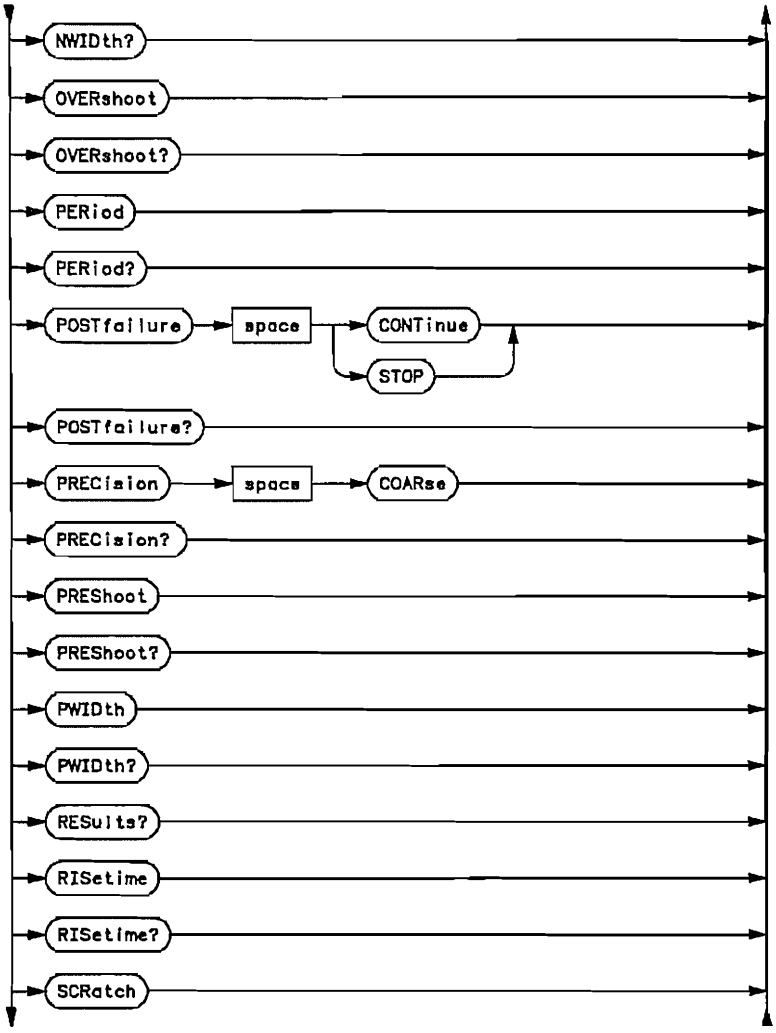


Figure 15-1. Measure Subsystem Syntax Diagrams (continued)

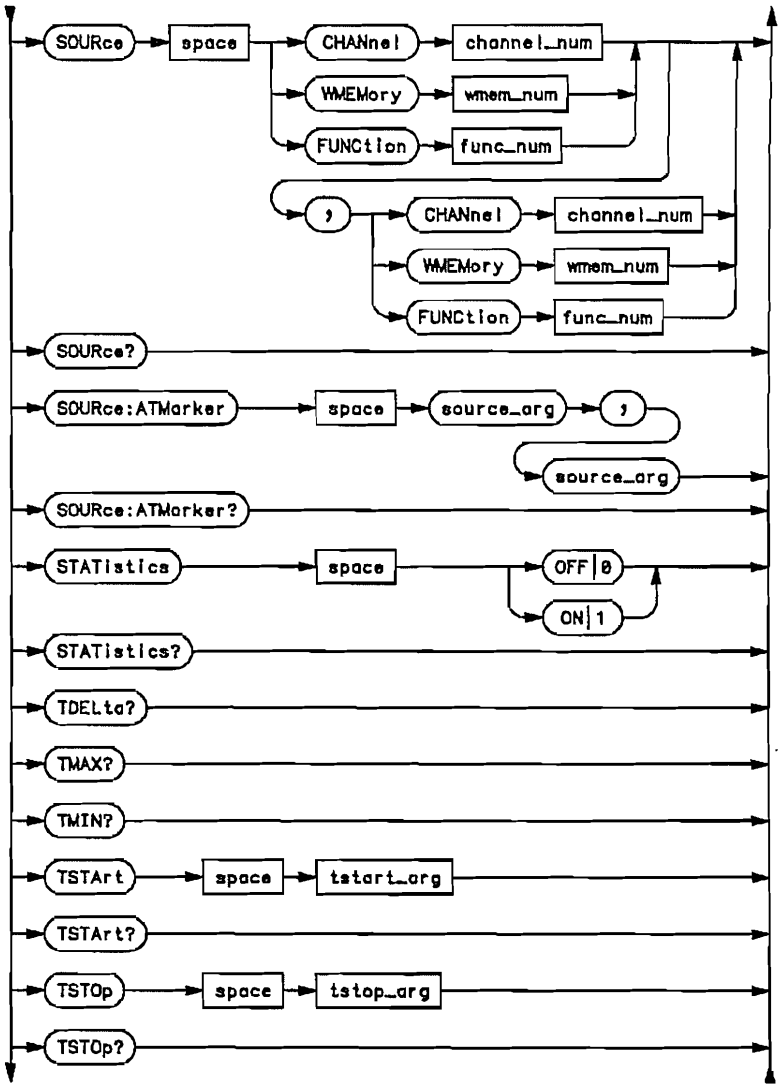


Figure 15-1. Measure Subsystem Syntax Diagrams (continued)

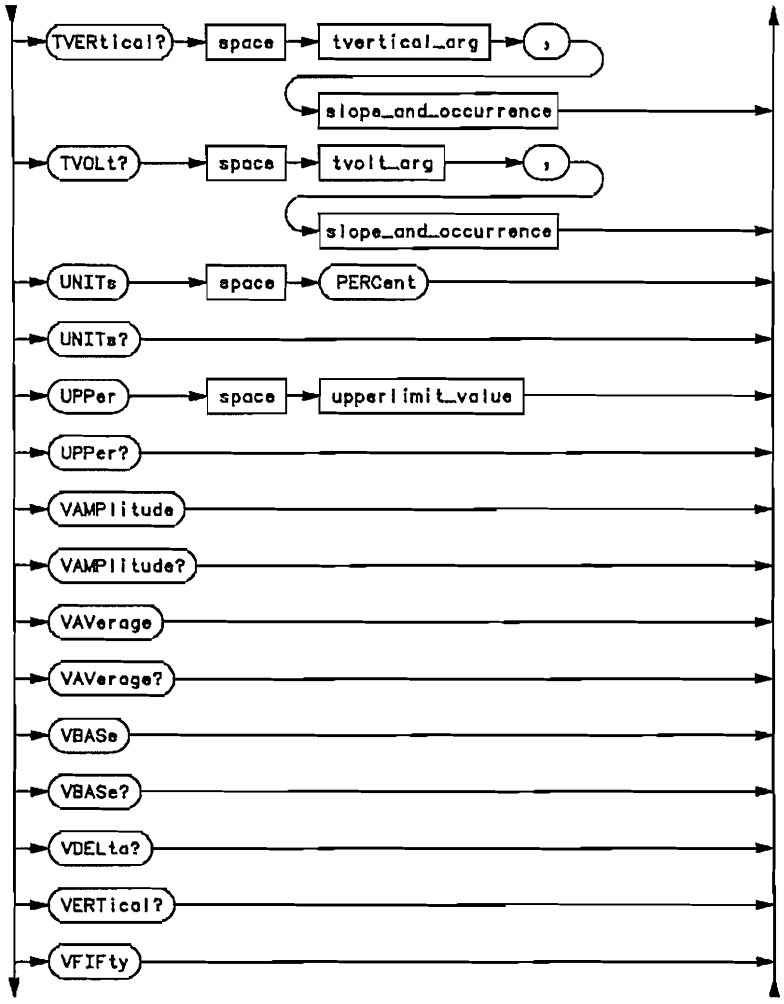


Figure 15-1. Measure Subsystem Syntax Diagrams (continued)

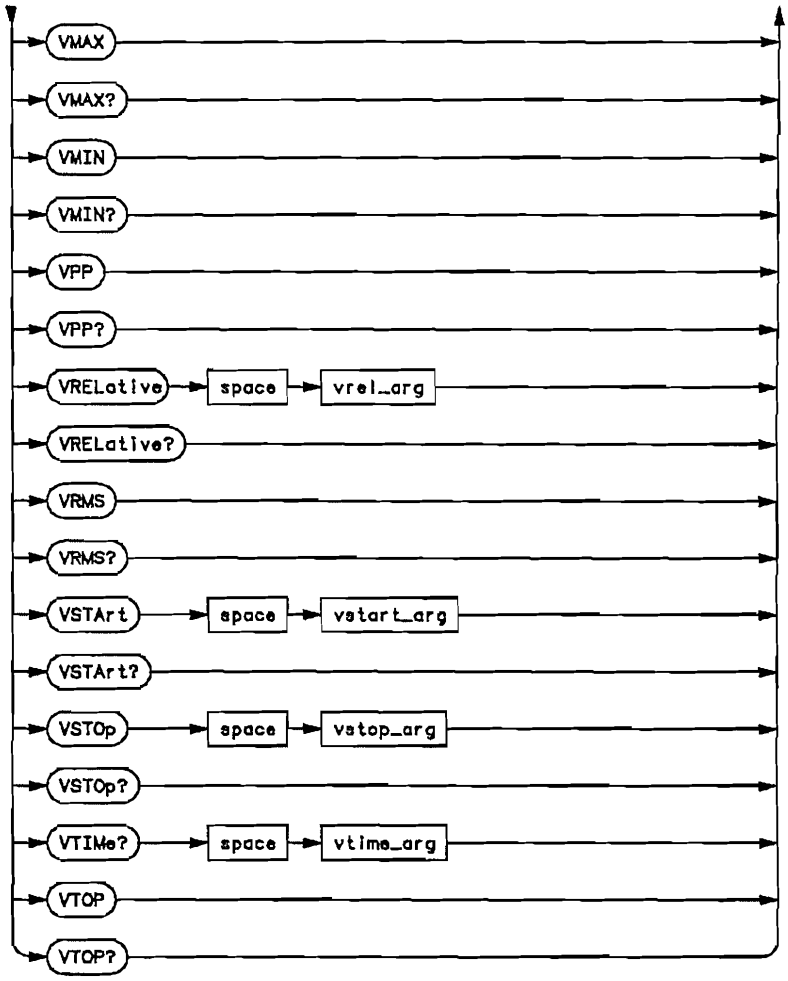


Figure 15-1. Measure Subsystem Syntax Diagrams (continued)

channel_number = An integer, 1, 2, 3, or 4.
edge_number = An integer 1 through 100.
func_num = An integer, 1 or 2.
level = MIDDle, UPPer, or LOWer.
lower_limit = Lower limit for compare.
lowlimit_value = Lower threshold value in percent.
measurement = Name of measurement to be compared.
pmem_num = An integer, 1 or 2.
polarity = Positive or negative
slope_and_occurrence = An integer −100 to 100 (excluding 0) specifying a displayed edge.
source_arg = Channel 1, 2, 3, or 4, function 1 or 2, or waveform memory 1, 2, 3, or 4
tstart_arg = Time in seconds from trigger.
tstop_arg = Time in seconds from trigger.
tvertical_arg = A real number specifying voltage or power.
tvolt_arg = A real number specifying voltage.
upper_limit = Upper limit for compare.
upperlimit_value = Upper threshold value in percent.
vrel_arg = An integer, 0 to 100.
vstart_arg = A real number within amplitude range.
vstop_arg = A real number within amplitude range.
vtime_arg = A real number in the horizontal display window.
wmem_num = An integer, 1 through 4.

Figure 15-1. Measure Subsystem Syntax Diagrams (continued)

ALL?

The :MEASURE:ALL query makes a set of measurements on the displayed signal and buffers the measurement results for output over the HP-IB. The measurement results are not output to the Peak Power Analyzer display.

To make a measurement, the portion of the waveform required for the measurement must be displayed. Time measurements are made on the first (left-most) displayed edges of the waveforms. To obtain the most accurate measurement possible, use proper horizontal scaling.

Refer to the individual commands for information on how the measurements are made and for the returned format of the measurement results.

Query Syntax: :MEASure:ALL?

Returned Format:

[:MEASure:FREQuency] <result>;[PERiod] <result>;
[PWIDth]<result>;[NWIDth] <result>;[RISetime]<result>;
[FALLtime]<result>;[OVERshoot]<result>;[DUTycycle]
<result>;[VMAX]<result>;[VMIN]<result>;[VTOP] <result>;
[VBASe]<result>;[VAVerage]<result><NL>

Where:

<result> ::= individual measurement results (exponential– NR3 format)

Example:

```
DIM A$[500]
OUTPUT 707;":MEASURE:ALL?"
ENTER 707;A$
PRINT A$
END
```

Note

These values can be returned to numeric variables instead of the BASIC string variables as shown, If numeric variables are used, the headers must be turned off.

ATMarker?

The :MEASURE:ATMARKER query is used with an amplitude-at-time measurement. The amplitude-at-time function allows the user to position one or both time markers on the displayed waveform and read the amplitude of the signal at that time. The ATMARKER query returns the amplitude at the specified time marker.

Related Commands

- :DISPlay:ATMarker
- :MEASure:SOURce:ATMarker
- :MEASure:DEFine ATIME
- :MEASure:ATMarker:UNITS?

For more information about these commands, refer to the Display and Measure Subsystems.

Note

Turn the time markers on (:DISPlay:TMARker) and the amplitude-at-time markers (:DISPlay:ATMarker) before using the :MEASure:ATMarker command.

When making an amplitude-at-time measurement on a waveform in memory, set the current timebase (:TIMEBASE:RANGE) to the timebase of the stored waveform.

Query Syntax: :MEASure:ATMarker? {LOWer | UPPer}

Where:
Lower and Upper ::= Time marker(s) as defined by the :MEASURE:DEFINE ATIME command.

Note

The :MEASURE:DEFINE ATIME command defines the upper and lower marker fields of the front panel **ampl@time** softkey (Marker Menu) to be either start or stop markers.

Returned Format:

[:MEASure:ATMarker] <result><NL>

Where:
<result> ::= The amplitude at the specified time. (exponential–NR3 format)

Note

The units associated with ATMARKER can be queried using the :MEASURE:ATMARKER:UNITS? query.

Example:

```
DIM Res$[100]
OUTPUT 707;"MEASure:ATMarker? LOWer"
ENTER 707;Res$
PRINT Res$
END
```

ATMarker:UNITs?

The :MEASURE:ATMARKER:UNITS query is used with an amplitude-at-time measurement. The amplitude-at-time function allows the user to position one or both time markers on the displayed waveform and read the amplitude of the signal at that time. The ATMARKER:UNITS query returns the measurement units of the displayed source at the specified time marker.

Related Commands

- :DISPlay:ATMarker
- :MEASure:SOURce:ATMarker
- :MEASure:DEFine ATIME
- :MEASure:ATMarker?

For more information about these commands, refer to the Display and Measure Subsystems.

Query Syntax: :MEASure:ATMarker:UNITs? {LOWer | UPPer}

Where:
Lower and Upper ::= Time marker as defined by the :MEASURE:DEFINE ATIME command.

Note

The :MEASURE:DEFINE ATIME command defines the upper and lower marker fields of the front panel **ampl@time** softkey (Marker Menu) to be either start or stop markers.

Returned Format:

[:MEASure:ATMarker:UNITs] <unit><NL>

Where:

<unit> ::= The current unit (W, V, or DBM) for the specified marker.

Example:

```
DIM Unit$[100]
OUTPUT 707;"MEASure:ATMarker:UNIT? LOWer"
ENTER 707;Unit$
PRINT Unit$
END
```

AUTodig

The AUTODIG command enables the Peak Power Analyzer to automatically execute the Root Level DIGITIZE command before one of the following measurements is made:

- :MEASure:VERTical?
- :MEASure:VMAX?
- :MEASure:VTIMe?

AUTODIG only works with these three measurements.

Note

Use AUTodig when one or more of the listed measurements is to be made on a number of different waveforms. If multiple measurements, such as, RISETIME, FALLTIME, PRF, and VMAX are to be made on a single waveform, digitize the waveform and then make the measurements with AUTodig set to off.

The measurement is completed faster when AUTodig is on compared to the combination of digitizing the waveform and then making the measurement. The results are placed in the output queue. However, the results are not placed in the measurement queue or displayed on the screen. Refer to the desired measurement for more information.

Command Syntax :MEASure:AUTodig {{ON | 1} {OFF | 0}}

Example

```
OUTPUT 707;":MEASURE:AUTodig ON"
OUTPUT 707;":MEASURE:VMAX?"
```

Query Syntax: :MEASure:AUTodig?

Returned Format: [:MEASure:AUTodig] {1 | 0 }<NL>

Example:

```
DIM Adig$[100]
OUTPUT 707;"MEASure:AUTodig?"
ENTER 707;Adig$
PRINT Adig$
END
```

COMPare

The :MEASURE:COMPARE command specifies the measurement and limits to be used for the :MEASURE:LIMITTEST command. The first limit is the upper limit; the second is the lower limit.

This command does not start the test, but only sets the test parameters. The :MEASURE:LIMITTEST command starts the test.

The COMPARE query returns the current specification

Related Commands

- :MEASure:DESTination
- :MEASure:LIMittest
- :MEASure:POSTfailure

Note

The remote-only measurements are not supported by the measurement limit test. If any of these measurements are active during the measurement limit test, the measurement will always indicate "PASSED". These are the remote-only measurements: PREShoot, TMAX, TMIN, TVOLt, VAMPlitude, VERTical, VMIN, VPP, VRMS, and VTIMe.

Command Syntax: :MEASure:COMPare <measurement>,<upper_limit>,<lower_limit>

Note

If the argument is in volts or watts, no units suffix is required. However, if the argument is in dBm, the suffix dBm is required. The Peak Power Analyzer will convert the argument to the current display units.

Where:

<measurement> ::= {FALLtime | FREQuency | OVERshoot | PERiod | PWIDth | NWIDth | RISetime | VBASe | VTOP | VAVerage | VMAX | DUTycycle | DELay}

<upper_limit> ::= High limit value.
<lower_limit> ::= Low limit value.

Example:

```
OUTPUT 707;":MEASURE:COMPARE RISETIME,10,4"
```

Query Syntax:

:MEASure:COMPare? <measurement>

Where:

<measurement> ::={FALLtime | FREQuency | OVERshoot | PERiod | PWIDth | NWIDth | RISetime | VBASe | VTOP | VAVerage | VMAX | DUTycycle | DELay}

Returned Format:

[:MEASure:COMPare] <measurement>, <upper_limit>, <lower_limit><NL>

Example:

```
DIM Cmp$[100]
OUTPUT 707;"MEASure:COMPare? VPP"
ENTER 707;Cmp$
PRINT Cmp$
END
```

For example, the sequence required to do a limit test on frequency is:

```
OUTPUT 707;":MEASURE:SOURCE CHAN1"
!Selects channel 1 as the source.
OUTPUT 707;":MEASURE:FREQ"
!Select measurement
OUTPUT 707;":MEASURE:COMPARE FREQ,1000HZ,10HZ"
!Set measurement limits
OUTPUT 707;":MEASURE:LIMITTEST MEASURE"
!Start test
```

Note

The only way to see if a limit test failure has occurred over the bus is by checking if bit 3 (LTER?) of the status byte is set to a 1.

CURSor?

The :MEASURE:CURSOR query returns the time and amplitude values of the specified marker as an ordered pair of time/amplitude values.

When the CURSOR query is sent, no measurement is made, and the cursors are not moved.

If DELTA is specified:

the instrument returns the value of delta time and delta vertical.

If START is specified:

the positions of the start marker and amplitude Marker 1 are returned.

If STOP is specified:

the positions of the stop marker and amplitude Marker 2 are returned.

Query Syntax: :MEASure:CURSor? {DELTa | STARt | STOP}

Returned Format:

[:MEASure:CURSor] <time>,<amplitude><NL>

Where:

<time> ::= delta time, start time or stop time
<amplitude> ::= delta amplitude, Marker 1 amplitude or Marker 2 amplitude (Amplitude will be in volts or power, depending on the channel that is being used.) (exponential—NR3 format)

Example:

```
DIM Cur$[100]
OUTPUT 707;":MEAS:SOURCE CHAN1"
OUTPUT 707;":MEAS:CURSOR? START"
ENTER 707;Cur$
PRINT Cur$
END
```

DEFine

The :MEASURE:DEFINE command sets up the definition for a measurement. This command is only valid in "user mode". To set "user mode", refer to the MEASURE:MODE command in this section.

The DEFINE query returns the current setup.

Related Commands

- :MEASure:MODE
- :MEASure:UPPer
- :MEASure:LOWer
- :MEASure:UNITs

Command Syntax:

:MEASure:DEFine <measurement_spec>

Where:

<measurement_spec> ::= {ATIMe, {{UPPer | LOWer}, {STARt |STOP}} | DELay <polarity>, <edge_num>, <level>, <polarity>, <edge_num>, <level> | PWIDth, {MIDDle | UPPer | LOWer} | NWIDth, {MIDDle | UPPer | LOWer}}

Example:

```
OUTPUT 707;":MEAS:DEFINE DELAY,POSITIVE,1,
UPPER,NEGATIVE,2,MIDDLE"
```

This example will set the parameters for a time measurement from the first positive edge at the upper threshold level to the second negative edge at the middle threshold. If one source is specified, both parameters apply to that signal. If two sources are specified the

measurement is from the first positive edge on source 1 to the second negative edge on source 2.

Query Syntax: :MEASure:DEFine? {ATIMe,{UPPer | LOWer} | DELay | PWIDth | NWIDth}

Returned Format:

[:MEASure:DEFine] <measurement_spec><NL>

Where:

<measurement_spec> ::= {ATIMe, {{UPPer | LOWer}, {STARt |STOP}} | DELay <polarity>, <edge_num>, <level>, <polarity>, <edge_num>, <level> | PWIDth, {MIDDle | UPPer | LOWer} | NWIDth, {MIDDle | UPPer | LOWer}}

Example:

```
DIM Dfn$[100]
OUTPUT 707;":MEASure:DEFine? DELay"
ENTER 707;Dfn$
PRINT Dfn$
END
```

DELay

The :MEASURE:DELAY command causes the Peak Power Analyzer to determine the delay either between two edges on the same source or between an edge on one source and an edge on another source.

The sources being measured are specified with the :MEASURE:SOURCE command.

If user defined measurement specifications are selected, ensure the defined measurement is displayed.

The DELAY query returns the specified delay value.

Command Syntax: :MEASure:DELay

Example:

```
OUTPUT 707;":MEAS:DEL"
```

Query Syntax: :MEASure:DELay?

Returned Format:

[:MEASure:DELay] <delay_value><NL>

Where:

<delay_value> ::= time value in seconds (exponential–NR3 format)

Example:

```
DIM Dly$[100]
OUTPUT 707;":MEAS:DELAY?"
ENTER 707;Dly$
PRINT Dly$
END
```

DESTination

The :MEASURE:DESTINATION command specifies the destination to be used when a limittest violation is found.

If a waveform memory is specified, the memory is overwritten each time a violation is found.

The DESTINATION query returns the destination currently specified.

Note

If Waveform Memory is the destination, the source must be set up separately using the Root Level command :STORE.

Related Commands

- :MEASure:COMPare
- :MEASure:LIMittest
- :MEASure:POSTfailure

Command Syntax: :MEASure:DESTination {WMEMory{1 | 2 | 3 | 4} | PMEMory{1 | 2} | OFF}

Example:

```
OUTPUT 707;":MEAS:DEST PMEMORY2"
```

Query Syntax: :MEASure:DESTination?

Returned Format:

[:MEASure:DESTination] {WMEMory{1 | 2 | 3 | 4} | PMEMory{1 | 2} | OFF}<NL>

Example:

```
DIM Dst$[100]
OUTPUT 707;":MEAS:DEST?"
ENTER 707;Dst$
PRINT Dst$
END
```

DUTycycle

The :MEASURE:DUTYCYCLE command places the Peak Power Analyzer in the continuous measurement mode and starts the Dutycycle measurement.

The DUTYCYCLE query measures and outputs the duty cycle of the signal specified by the SOURCE command. The signal must be displayed for the measurement to be made. The value returned for duty cycle is the ratio of the positive pulse width to period.

The positive pulse width and the period of the specified signal are measured, and then the duty cycle is calculated.

The duty cycle is calculated with the following formula:

duty cycle = +pulse width/period

Command Syntax: :MEASure:DUTycycle

Example:

```
OUTPUT 707;":MEASURE:DUTYCYCLE"
```

Query Syntax: :MEASure:DUTycycle?

Returned Format:

[MEASure:DUTycycle] <value><NL>

Where:

<value> ::= ratio of +pulse width to period (exponential –NR3 format)

Example:

```
DIM Dc$[100]
OUTPUT 707;":MEASURE:DUTYCYCLE?"
ENTER 707;Dc$
PRINT Dc$
END
```

ESTArt

The :MEASURE:ESTART command causes the Peak Power Analyzer to position the start marker on the specified edge and slope of the displayed waveform. All edges must be displayed and are counted from the left edge of the display. The start marker is positioned where amplitude Marker 1 intersects the waveform. The desired edge is specified by sending an integer value after the command name. If a positive integer is sent, the Peak Power Analyzer will place the start marker on a positive-going waveform edge. If a negative integer is sent, the start marker will be placed on a negative-going waveform edge. If amplitude Marker 1 does not intersect the waveform as specified, the error message "Edges required not found" is displayed.

The ESTART query returns the currently specified edge.

Note

Amplitude marker 1 needs to be positioned on the waveform in order to use the ESTART command. The :MEASURE:VFIFTY command, or the combination of :MEASURE:VTOP and :MEASURE:VRELATIVE can be used to place the amplitude marker on the waveform.

The short form of this command does not follow the defined convention. The short form "EST" is the same for ESTART and ESTOP, so be careful not to send this form for the ESTART command. Sending "EST" will produce an error.

Command Syntax:

:MEASure:ESTArt <edge>

Where:

<edge> ::= sign and number (if a positive value is sent the + sign may be omitted or a space may be used)

Example:

```
OUTPUT 707;":MEASURE:ESTART 2"
```

This example places the start marker at the second displayed positive-going intersection of the waveform and amplitude Marker 1.

Query Syntax:

:MEASure:ESTArt?

Returned Format:

[:MEASure:ESTART] <edge><NL>

Where:

<edge> ::= edge number (integer–NR1 format)

Example:

```
DIM Estart$[100]
OUTPUT 707;":MEAS:ESTART?"
ENTER 707;Estart$
PRINT Estart$
END
```

ESTOp

The :MEASURE:ESTOP command causes the Peak Power Analyzer to position the stop marker on the specified edge and slope of the displayed waveform. All edges must be displayed and are counted from the left edge of the display. The stop marker is positioned where amplitude Marker 2 intersects the waveform. The desired edge is specified by sending an integer value after the command name. If a positive integer is sent, the Peak Power Analyzer places the stop marker on a positive-going waveform edge. If a negative integer is sent, the stop marker is placed on a negative-going waveform edge.

If amplitude Marker 2 does not intersect the waveform as specified, the error message "Edges required not found" is displayed.

The ESTOP query returns the currently specified edge.

Note

Amplitude marker 2 needs to be positioned on the waveform in order to use the ESTOP command. The :MEASURE:VFIFTY command, or the combination of :MEASURE:VTOP and :MEASURE:VRELATIVE can be used to place the amplitude markers on the waveform.

The short form of this command does not follow the defined convention. The short form "EST" is the same for ESTART and ESTOP, so be careful not to send this form for the ESTOP command. Sending "EST" will produce an error.

Command Syntax:

:MEASure:ESTOp <edge>

Where:

<edge> ::= sign and number (if a positive value is sent the + sign may be omitted or a space may be used)

Example:

```
OUTPUT 707;":MEAS:ESTOP -2"
```

This example places the stop marker at the second displayed negative-going intersection of the waveform at amplitude Marker 2.

Query Syntax:

:MEASure:ESTOp?

Returned Format:

[:MEASure:ESTOP] <edge><NL>

Where:

<edge> ::= edge number (integer–NR1 format)

Example:

```
DIM Estop$[100]
OUTPUT 707;":MEASURE:ESTOP?"
ENTER 707;Estop$
PRINT Estop$
END
```

FALLtime

The :MEASURE:FALLTIME command places the Peak Power Analyzer in the continuous measurement mode and starts a Falltime measurement.

The FALLTIME query measures and outputs the fall time of the first displayed falling (negative-going) edge. For best measurement accuracy, set the sweep speed as fast as possible while leaving the falling edge of the waveform on the display. The falltime is determined by measuring the time at the upper threshold of the falling edge then measuring the time at the lower threshold of the falling edge and calculating the falltime with the formula:

fall time = time at lower threshold point−time at upper threshold point.

If the horizontal scaling is questionable when this measurement is made, "error 11" is placed in the error queue. For more information, refer to the Measurement Error section at the beginning of this chapter.

Command Syntax: :MEASure:FALLtime

Example:

```
OUTPUT 707;":MEAS:FALL"
```

Query Syntax: :MEASure:FALLtime?

Returned Format:

[:MEASure:FALLtime] <value><NL>

Where:

<value> ::= time in seconds between lower threshold and upper threshold amplitude points (exponential–NR3 format)

Example:

```
DIM F11$[100]
OUTPUT 707;":MEASURE:FALLTIME?"
ENTER 707;F11$
PRINT F11$
END
```

FREQuency

The :MEASURE:FREQuency command places the Peak Power Analyzer in the continuous measurement mode and starts a Pulse Repetition Frequency measurement.

The FREQuency query measures and outputs the frequency of the first complete cycle on screen using the 50% levels when Standard measurements are selected and at the mid threshold value when User Defined measurements are selected.

The algorithm is:

if first edge on screen is rising
then
frequency = 1/(time at second rising edge
−time at first rising edge)
else
frequency = 1/(time at second falling edge
−time at first falling edge)

Command Syntax: :MEASure:FREQuency

Example:

```
OUTPUT 707;":MEASURE:FREQuency"
```

Query Syntax: :MEASURE:FREQuency?

Returned Format:

[:MEASure:FREQuency] <value><NL>

Where:

<value> ::= frequency in Hertz (exponential–NR3 format)

Example:

```
DIM Freq$[100]
OUTPUT 707;":MEASURE:FREQuency?"
ENTER 707;Freq$
PRINT Freq$
END
```

LIMittest

The :MEASURE:LIMITTEST command enables a limit test to be performed. The Peak Power Analyzer can run limit tests on up to three measurements. If LIMITTEST is sent with the MEASURE parameter, the Peak Power Analyzer starts the test. If the OFF parameter is sent, the test is stopped.

The LTF (limit test failure) bit of the status byte is set when a failure is found.

Related Commands

- :MEASure:COMPare
- :MEASure:DESTination
- :MEASure:POSTfailure

The measurement and the measurement limits are set with the :MEASURE:COMPARE command. The results of the measurement are sent to the location specified with the :MEASURE:DESTINATION command. Testing is stopped or continued after a failure with the :MEASURE:POSTFAILURE command.

Command Syntax:

:MEASure:LIMittest {MEASure | OFF}

Example:

```
OUTPUT 707;":MEAS:LIM MEAS"
```

LOWer

The :MEASURE:LOWER command sets the lower measurement threshold. The command only applies in "user mode". To set "user mode", refer to the MEASURE:MODE command in this chapter. The value sent to the Peak Power Analyzer is in percent.

The LOWER query returns the current setting of the lower measurement threshold.

Related Commands

- :MEASure:MODE
- :MEASure:UPPer
- :MEASure:DEFine
- :MEASure:UNITs

Command Syntax:

:MEASure:LOWer <lowlimit_value>

Where:

<lowlimit_value> ::= user defined lower threshold in percent.

Example:

```
OUTPUT 707;":MEASURE:LOWER 47"
```

Query Syntax: :MEASure:LOWer?

Returned Format:

[:MEASure:LOWer] <lowlimit_value><NL>

Where:

<lowlimit_value> ::= user defined lower threshold in percent. (exponential–NR3 format)

Example:

```
DIM Lwr$[100]
OUTPUT 707;":MEAS:LOW?"
ENTER 707;Lwr$
PRINT Lwr$
END
```

MODE

The :MEASURE:MODE command sets the measurement mode (definitions and thresholds).

The MODE query returns the current mode setting.

Related Commands

- :MEASure:DEFine
- :MEASure:LOWer
- :MEASure:UPPer
- :MEASure:UNITs

Command Syntax: :MEASure:MODE {STANdard | USER}

Example:

```
OUTPUT 707;":MEAS:MODE STAN"
```

Query Syntax: :MEASure:MODE?

Returned Format:

[:MEASure:MODE] {STANdard | USER}<NL>

Example:

```
DIM Md$[100]
OUTPUT 707;":MEASURE:MODE?"
ENTER 707;Md$
PRINT Md$
END
```

NWIDth

The :MEASURE:NWIDth command places the Peak Power Analyzer in the continuous measurement mode and starts a pulse offtime measurement.

The NWIDth query measures and outputs the width of the first negative pulse on screen. The NWIDTH query uses the 50% levels when Standard measurements are selected.

If User Defined measurements are selected, the measurement is made at the specified threshold value.

The algorithm is:

if the first edge on screen is rising
then
width = (time at second rising edge
−time at first falling edge)
else
width = (time at first rising edge
−time at first falling edge)

Command Syntax: :MEASure:NWIDth

Example:

```
OUTPUT 707;":MEAS:NWIDth"
```

Query Syntax: :MEASure:NWIDth?

Returned Format:

[:MEASure:NWIDth] <value><NL>

Where:

<value> ::= negative pulse width in seconds (exponential –NR3 format)

Example:

```
DIM Nwd$[100]
OUTPUT 707;":MEASURE:NWIDth?"
ENTER 707;Nwd$
PRINT Nwd$
END
```

OVERshoot

The :MEASURE:OVERSHOOT command places the Peak Power Analyzer in the continuous measurement mode and selects the Overshoot measurement.

The OVERSHOOT query measures and outputs the overshoot of a selected signal. Overshoot measures the first edge on screen with the following algorithm:

overshoot = (peak−top)/top−base

Note

The Peak Power Analyzer isn't able make automatic overshoot measurements when overshoot is greater than 50%. It may be necessary to use the markers to make the measurement.

Command Syntax: :MEASure:OVERshoot

Example:

```
OUTPUT 707;":MEAS:OVER"
```

Query Syntax: :MEASure:OVERshoot?

Returned Format:

[:MEASure:OVERshoot] <value><NL>

Where:

<value> ::= ratio of overshoot to amplitude, top−base (exponential −NR3 format)

Example:

```
DIM Ovr$[100]
OUTPUT 707;":MEASURE:OVERSHOOT?"
ENTER 707;Ovr$
PRINT Ovr$
END
```

PERiod

The :MEASURE:PERIOD command places the Peak Power Analyzer in the continuous measurement mode and selects the PRI (Pulse Repetition Interval) measurement.

The PERIOD query measures and outputs the PRI of the first complete cycle on screen. The PRI is measured at the 50% point if Standard measurements are selected and at the mid threshold level of the waveform if User-Defined measurements are selected.

The algorithm for this measurement is:

if the first edge on screen is rising
then
PRI = (time at second rising edge
−time at first rising edge)
else
PRI = (time at second falling edge
−time at first falling edge)

Command Syntax: :MEASure:PERiod

Example:

```
OUTPUT 707;":MEAS:PERIOD"
```

Query Syntax: :MEASure:PERiod?

Returned Format:

[:MEASure:PERiod] <value><NL>

Where:

<value> ::= waveform PRI in seconds (exponential–NR3 format)

Example:

```
DIM Prd$[100]
OUTPUT 707;":MEASURE:PERIOD?"
ENTER 707;Prd$
PRINT Prd$
END
```

POSTfailure

The :MEASURE:POSTFAILURE command specifies what will occur after a violation has been found by the limit test. If CONTINUE is selected, the Peak Power Analyzer will continue to look for another violation. If STOP is selected, the Peak Power Analyzer stops the limit test.

If CONTINUE is selected and a violation is found, the violation is written to the desired location. If a waveform memory is selected as the destination, then, all subsequent violations will overwrite the previous violation.

The POSTFAILURE query returns the current selection.

Related Commands

- :MEASure:COMPare
- :MEASure:DESTination
- :MEASure:LIMittest

Command Syntax: :MEASure:POSTfailure {CONTinue | STOP}

Example:

```
OUTPUT 707;":MEAS:POST CONT"
```

Query Syntax: :MEASure:POSTfailure?

Returned Format:

[:MEASure:POSTfailure] {CONTinue | STOP}<NL>

Example:

```
DIM Pf$[100]
OUTPUT 707;":MEASURE:POSTFAILURE?"
ENTER 707;Pf$
PRINT Pf$
END
```

PRECision

The :MEASURE:PRECISION command is included in the Peak Power Analyzer for compatibility with other Hewlett-Packard instruments. It has **no effect** on the Peak Power Analyzer.

The PRECISION query always returns COARSE in the Peak Power Analyzer.

Command Syntax: :MEASure:PRECision COARse

Example:

```
OUTPUT 707;":MEAS:PREC COARSE"
```

Query Syntax: :MEASure:PRECision?

Returned Format:

[:MEASure:PRECision] COARse<NL>

Example:

```
DIM Pc$[100]
OUTPUT 707;":MEAS:PRECISION?"
ENTER 707;Pc$
PRINT Pc$
END
```

PREShoot

Note

This command can only be used on a non-power measurement source, such as, channel 2 or 3, a function, or waveform memory.

The :MEASURE:PRESHOOT command places the Peak Power Analyzer in the continuous measurement mode and starts the Preshoot measurement.

The PRESHOOT query measures and outputs the preshoot of the selected signal. Preshoot measures the first edge on screen with the following algorithm:

if the first edge on screen is rising
then
preshoot = (Vbase−Vmin)/Vamplitude
else
preshoot = (Vmax−Vtop)/Vamplitude

Command Syntax: :MEASure:PREShoot

Example:

```
OUTPUT 707;":MEASURE:PRES"
```

Query Syntax: :MEASure:PREShoot?

Returned Format:

[:MEASure:PREShoot] <value><NL>

Where:

<value> ::= ratio of preshoot to Vamplitude (exponential –NR3 format)

Example:

```
DIM Prs$[100]
OUTPUT 707;":MEASURE:PRESHOOT?"
ENTER 707;Prs$
PRINT Prs$
END
```

PWIDth

The :MEASURE:PWIDTH command places the Peak Power Analyzer in the continuous measurement mode and starts the pulse width measurement.

The PWIDTH query measures and outputs the width of the first displayed positive pulse. Pulse width is measured at the 50% amplitude level with Standard measurements selected and at the specified level threshold value with User-Defined measurements selected. The algorithm for this measurement is:

if the first edge on screen is falling
then
width = (time at second falling edge
−time at first rising edge)
else
width = (time at first falling edge
−time at first rising edge)

Command Syntax: :MEASure:PWIDth

Example:

```
OUTPUT 707;":MEAS:PWIDTH"
```

Query Syntax: :MEASure:PWIDth?

Returned Format:

[:MEASure:PWIDth] <value><NL>

Where:

<value> ::= width of positive pulse in seconds (exponential –NR3 format)

Example:

```
DIM Pwd$[100]
OUTPUT 707;":MEASURE:PWIDTH?"
ENTER 707;Pwd$
PRINT Pwd$
END
```

RESults?

The :MEASURE:RESULTS query tells the Peak Power Analyzer to return the currently active measurements. If statistics are on, the current, minimum, maximum, and average are returned for each measurement. If the limit test is on and POSTFAILURE is set to CONTINUED, then, the pass ratio is returned instead of the average. This is a pure ratio and not a percentage.

If the number of measurements returned is 0 then no <measurement>s are returned.

Query Syntax: :MEASure:RESults?

Returned Format:

[:MEASure:RESults] <No. of Meas>[;<measurement> ...]<NL>

Where:

<No. of Meas> ::= number of measurements displayed on the CRT, 0 through 8. (integer–NR1 format)
<measurement> ::= measurement_name measurement_result.

Example:

```
DIM Mr$[100]
OUTPUT 707;":MEASURE:RESULTS?"
ENTER 707;Mr$
PRINT Mr$
END
```

RISetime

The :MEASURE:RISETIME command places the Peak Power Analyzer in the continuous measurement mode and starts a Risetime measurement.

The RISETIME query measures and outputs the risetime of the first displayed rising (positive-going) edge. For best measurement accuracy, set the sweep speed as fast as possible while leaving the leading edge of the waveform on the display. The risetime is determined by measuring the time at the lower threshold of the rising edge, and the time at the upper threshold of the rising edge. The risetime is then calculated with the following formula:

risetime = (time at upper threshold point−time at lower threshold point)

If the horizontal scaling is questionable when this measurement is made an "error 11" is placed in the error queue. For additional information, refer to the Measurement Error section at the beginning of this chapter.

Command Syntax: :MEASure:RISetime

Example:

```
OUTPUT 707;":MEAS:RIS"
```

Query Syntax: :MEASure:RISetime?

Returned Format:

[:MEASure:RISetime] <value><NL>

Where:

<value> ::= rise time in seconds (exponential –NR3 format)

Example:

```
DIM Rs$[100]
OUTPUT 707;":MEAS:RIS?"
ENTER 707;Rs$
PRINT Rs$
END
```

SCRatch

The :MEASURE:SCRATCH command clears the measurement results from the Peak Power Analyzer display.

Command Syntax: :MEASure:SCRatch

Example:

```
OUTPUT 707;":MEASURE:SCRATCH"
```

SOURce

The :MEASURE:SOURCE command selects the source(s) for the measurements. The source specified becomes the source for the Measure Subsystem commands. The source can be a channel, function, or waveform memory.

One or two sources can be specified with this command. All measurements except DELAY are made on the first specified source. The DELAY measurement uses two sources if two have been specified. If only one source is specified, the DELAY measurement uses that source for both of its parameters.

The SOURCE query returns the current source selection. If the specified sources are different both are returned. Otherwise, one source is returned.

Note

When Option 001, Single Sensor Input, is installed, channel 4 can't be specified. Error −360, "Missing a Baseband Board" is placed in the error queue when an attempt is made to specify channel 4.

Command Syntax: :MEASure:SOURce <source1>[,<source2>]

Where: <source1> and <source2> ::= {CHANnel{1 | 2 | 3 | 4} | FUNCtion{1 | 2} | WMEMory{1 | 2 | 3 | 4}}

Example:

```
OUTPUT 707;":MEASURE:SOURCE CHANNEL1, WMEMORY1"
```

Query Syntax: :MEASure:SOURce?

Returned Format:

[:MEASure:SOURce] <source1>[,<source2>]<NL>

Where:

<source1> and <source2> ::= {CHANnel{1 | 2 | 3 | 4} | FUNCtion{1 | 2} | WMEMory{1 | 2 | 3 | 4}

Example:

```
DIM Src$[100]
OUTPUT 707;":MEAS:SOUR?"
ENTER 707;Src$
PRINT Src$
END
```

SOURce:ATMarker

The :MEASURE:SOURCE:ATMARKER command selects the source(s) for an amplitude-at-time measurement. The amplitude-at-time function allows the user to position one or both time markers on the displayed waveform and read the amplitude at the marker. The source(s) specified become the source(s) for the amplitude-at-time measurement.

The SOURCE:ATMARKER query returns the current source selection. If the specified sources are different, both are returned. Otherwise, one source is returned.

Related Commands

- :DISPlay:ATMarker
- :MEASure:ATMarker?
- :MEASure:DEFine ATIME
- :MEASure:ATMarker:UNITS?

For more information about these commands, refer to the Display and Measure Subsystems.

Note

When the :MEASURE:SOURCE:ATMARKER command is received, the specified source(s) are turned on if not already on.

If Option 001, Single Sensor Input, is installed, channel 4 can't be specified. Error −360, "Missing a Baseband Board" is placed in the error queue when an attempt is made to specify channel 4.

Command Syntax: :MEASure:SOURce:ATMarker <Upper>[,<Lower>]

Note

<Upper> and <Lower> are associated with the two sources of the front panel **ampl@time** softkey (Marker Menu). <Upper> sets the top source and <Lower> sets the bottom source.

If one argument is sent both sources will be the same. Two arguments are needed to have two different sources.

Where:

<Upper> and <Lower> ::= {CHANnel{1 | 2 | 3 | 4} | FUNCtion{1 | 2} | WMEMory{1 | 2 | 3 | 4}}

Example:

```
OUTPUT 707;":MEAS:SOUR:ATM CHAN1, WMEM1"
```

Query Syntax: :MEASure:SOURce:ATMarker?

Returned Format:

[:MEASure:SOURce:ATMarker]
<Upper>[,<Lower>]<NL>

Where:

<Upper> and <Lower> ::= {CHANnel{1 | 2 | 3 | 4} | FUNCtion{1 | 2} | WMEMory{1 | 2 | 3 | 4}

Example:

```
DIM Sat$[100]
OUTPUT 707;":MEAS:SOUR:ATM?"
ENTER 707;Sat$
PRINT Sat$
END
```

STATistics

The :MEASURE:STATISTICS command enables the statistics mode to be controlled. When this mode is on and the measurements are in the continuous mode, the min, max, avg, and current measurement are shown as the active measurements. If a RESULTS query is executed, all of the displayed data is returned to the controller.

The STATISTICS query returns the current mode.

Note

"Average" is replaced by "pass ratio" when limit test is selected and "after failure" is set to continue. Pass ratio lists the percentage of times a certain test passed.

Command Syntax: :MEASure:STATistics {{ON | 1} |{OFF | 0}}

Example:

```
OUTPUT 707;":MEASURE:STAT ON"
```

Query Syntax: :MEASure:STATistics?

Returned Format:

[:MEASure:STATistics] {1 | 0}<NL>

Example:

```
DIM Stt$[100]
OUTPUT 707;":MEASURE:STAT?"
ENTER 707;Stt$
PRINT Stt$
END
```

TDELta?

The :MEASURE:TDELTA query returns the time difference between the start and stop time markers; that is:

Tdelta = Tstop−Tstart

where Tstart is the time at the start marker, and Tstop is the time at the stop marker

Query Syntax: :MEASure:TDELta?

Returned Format:

[:MEASure:TDELta] <value><NL>

Where:

<value> ::= difference between start and stop markers (exponential–NR3 format)

Example:

```
DIM Td1$[100]
OUTPUT 707;":MEASURE:TDELTA?"
ENTER 707;Td1$
PRINT Td1$
END
```

TMAX?

The :MEASURE:TMAX query returns the time at which the first maximum amplitude occurred. This assumes that the waveform is not clipped.

Query Syntax: :MEASure:TMAX?

Returned Format:

[:MEASure:TMAX] <time at max amplitude><NL> (exponential–NR3 format)

Example:

```
DIM Tmx$[100]
OUTPUT 707;":MEASURE:TMAX?"
ENTER 707;Tmx$
PRINT Tmx$
END
```

TMIN?

The :MEASURE:TMIN query returns the time at which the first minimum amplitude occurred. This assumes that the waveform is not clipped.

Query Syntax: :MEASure:TMIN?

Returned Format:

[:MEASure:TMIN] <time at min amplitude><NL>
(exponential–NR3 format)

Example:

```
DIM Tmn$[100]
OUTPUT 707;":MEAS:TMIN?"
ENTER 707;Tmn$
PRINT Tmn$
END
```

TSTArt

The :MEASURE:TSTART command moves the start marker to the specified time with respect to the trigger time.

The TSTART query returns the time at the start marker.

Note

The short form of this command does not follow the defined convention. The short form "TST" is the same for TSTART and TSTOP, so be careful not to send this form for the TSTART command. Sending "TST" will produce an error.

Command Syntax: :MEASure:TSTArt <start marker time>

Where:

<start marker time> ::= time at start marker in seconds

Example:

```
OUTPUT 707;":MEASURE:TSTART 30 NS"
```

Query Syntax: :MEASure:TSTArt?

Returned Format:

[:MEASure:TSTArt] <value><NL>

Where:

<value> ::= time at start marker in seconds (exponential- NR3 format)

Example:

```
DIM Tst$[100]
OUTPUT 707;":MEASURE:TSTART?"
ENTER 707;Tst$
PRINT Tst$
END
```

TSTOp

The :MEASURE:TSTOP command moves the stop marker to the specified time with respect to the trigger time.

The TSTOP query returns the time at the stop marker.

Note

The short form of this command does not follow the defined convention. The short form "TST" is the same for TSTART and TSTOP, so be careful not to send this form for the TSTOP command. Sending "TST" will produce an error.

Command Syntax: :MEASure:TSTOp <stop marker time>

Where:

<stop marker time> ::= time at stop marker in seconds

Example:

```
OUTPUT 707;":MEAS:TSTOP 40 NS"
```

Query Syntax: :MEASure:TSTOp?

Returned Format:

[:MEASure:TSTOp] <value><NL>

Where:

<value> ::= time at stop marker in seconds (exponential– NR3 format)

Example:

```
DIM Tst$[100]
OUTPUT 707;":MEASURE:TSTOP?"
ENTER 707;Tst$
PRINT Tst$
END
```

TVERtical?

When the :MEASURE:TVERtical? query is sent, the displayed signal is searched for the defined amplitude and transition. The time interval between the trigger event and this defined occurrence is returned as the response to this query.

For voltage waveforms, the <amplitude> can be specified as a negative or positive value. To specify a negative value, use a minus (−) sign. The sign of <slope> selects a rising (+) or falling (−) edge.

The magnitude of <occurrence> defines the occurrence to be reported. For example, +3 will return the time for the third time the waveform crosses the specified level in the positive direction. Once this level crossing is found, the Peak Power Analyzer will output the time at that crossing in seconds with the trigger point (time zero) as the reference.

If the indicated crossing cannot be found, the Peak Power Analyzer outputs +9.99999E+37. This will happen if the waveform does not cross the specified amplitude, or if the waveform does not cross the specified amplitude for the indicated number of times in the specified direction.

Query Syntax: :MEASure:TVERtical?
<amplitude>,<slope><occurrence>

Note If the argument is in volts or watts, no units suffix is required. However, if the argument is in dBm, the suffix dBm is required.

Where:

<amplitude> ::= level the waveform must cross.
<slope> ::= direction of waveform when <amplitude> is crossed, rising (+) or falling (−)
<occurrence> ::= number of crossing to be reported (if one –first crossing reported, if two–second crossing is reported, etc.)

Returned Format:

[:MEASure:TVERtical] <time><NL>

Where:

<time> ::= time in seconds of specified amplitude crossing (exponential–NR3 format)

Example:

```
DIM Tver$[100]
OUTPUT 707;"MEASURE:SOURCE CHANNEL1"
OUTPUT 707;"MEASURE:TVERtical? 5M,+3"
ENTER 707;Tver$
PRINT Tver$
END
```

TVOLt?

Note

This command can only be used with a voltage measurement source, such as, channel 2 or 3, a function, or waveform memory. For the command that works with a power or voltage measurement source, see :MEASURE:TVERTICAL.

When the :MEASURE:TVOLT query is sent, the displayed signal is searched for the defined voltage level and transition. The time interval between the trigger event and this defined occurrence is returned as the response to this query.

The <voltage> can be specified as a negative or positive voltage. To specify a negative voltage, use a minus (−) sign. The sign of <slope> selects a rising (+) or falling (−) edge.

The magnitude of <occurrence> defines the occurrence to be reported. For example, +3 will return the time for the third time the waveform crosses the specified voltage level in the positive direction. Once this voltage crossing is found, the Analyzer will output the time at that crossing in seconds with the trigger point (time zero) as the reference.

If the indicated crossing cannot be found, the Analyzer outputs +9.99999E+37. This will happen if the waveform does not cross the specified voltage, or if the waveform does not cross the specified voltage for the indicated number of times in the specified direction.

Query Syntax: :MEASure:TVOLt? <voltage>[V], <slope><occurrence>

Where:

<voltage> ::= voltage level the waveform must cross. This can be a positive or negative voltage.
[V] ::= voltage suffix
<slope> ::= direction of waveform when <voltage> is crossed, rising (+) or falling (−)
<occurrence> ::= number of crossing to be reported (if one –first crossing reported, if two–second crossing is reported, etc.)

Returned Format:

[:MEASure:TVOLt] <time><NL>

Where:

<time> ::= time in seconds of specified voltage crossing (exponential–NR3 format)

Example:

```
DIM Tvlt$[100]
OUTPUT 707;"MEASURE:TVOLT? -.250,+3"
ENTER 707;Tvlt$
PRINT Tvlt$
END
```

UNITs

The :MEASURE:UNITS command sets the measurement threshold units when user defined measurement mode is selected. The command has been included for compatibility with other Hewlett-Packard instruments. Percent is the default value.

Note

Since percent is the only unit available and the Peak Power Analyzer defaults to percent, the UNIT command is optional.

The UNITS query will return percent.

Related Commands

- :MEASure:DEFine
- :MEASure:LOWer
- :MEASure:MODE
- :MEASure:UPPer

Command Syntax: :MEASure:UNITs {PERCent}

Example:

```
OUTPUT 707;":MEASURE:UNITS PERCENT"
```

Query Syntax: :MEASure:UNITs?

Returned Format:

[:MEASure:UNITs] {PERCent}<NL>

Example:

```
DIM Unt$[100]
OUTPUT 707;":MEASURE:UNITS?"
ENTER 707;Unt$
PRINT Unt$
END
```

UPPer

The :MEASURE:UPPER command sets the upper measurement threshold. The command applies only to "user mode". To set "user mode", refer to the MEASURE:MODE command in this section. The units of the upper measurement threshold are always in percentages.

The UPPER query returns the value of the upper measurement threshold.

Related Commands

- :MEASure:DEFine
- :MEASure:LOWer
- :MEASure:MODE
- :MEASure:UNITs

Command Syntax:

:MEASure:UPPer <value>

Where:

<value> ::= upper threshold value in percent

Example:

```
OUTPUT 707;":MEAS:UPPER 90"
```

Query Syntax: :MEASure:UPPer?

Returned Format:

[:MEASure:UPPer] <upper_threshold value><NL>

Where:

<upper_threshold value> ::= upper threshold value in percent (exponential–NR3 format)

Example:

```
DIM Upp$[100]
OUTPUT 707;":MEAS:UPP?"
ENTER 707;Upp$
PRINT Upp$
END
```

VAMPlitude

Note

This command can only be used with a non-power measurement source, such as, channel 2 or 3, a function, or waveform memory.

The :MEASURE:VAMPLITUDE command places the Peak Power Analyzer in the continuous measurement mode and starts a Vamplitude measurement.

The VAMPLITUDE query returns the top-base amplitude of the displayed signal. The VAMPLITUDE value will not normally be the same as the Vp-p value if the input signal is a pulse.

The top-base value is calculated with the following formula:

amplitude = Vtop−Vbase

Command Syntax: :MEASure:VAMPlitude

Example:

```
OUTPUT 707;":MEAS:VAMP"
```

Query Syntax: :MEASure:VAMPlitude?

Returned Format:

[:MEASure:VAMPlitude] <value><NL>

Where:

<value> ::= difference between top and base amplitude– delta Vertical value (exponential–NR3 format)

Example:

```
DIM Vmp$[100]
OUTPUT 707;":MEASURE:VAMPLITUDE?"
ENTER 707;Vmp$
PRINT Vmp$
END
```

VAVerage

The :MEASURE:VAVERAGE command places the Peak Power Analyzer in the continuous measurement mode and starts a Vaverage measurement.

When a full cycle of a periodic waveform is detected, the VAVERAGE query returns the average over only one period of the signal not the overall average of the displayed signal.

Command Syntax: :MEASure:VAVerage

Example:

```
OUTPUT 707;":MEAS:VAV"
```

Query Syntax: :MEASure:VAVerage?

Returned Format:

[:MEASure:VAVerage] <avg_value><NL>

Where:

<avg_value> ::= calculated average amplitude (exponential- NR3 format)

Example:

```
DIM Vv$[100]
OUTPUT 707;":MEAS:VAV?"
ENTER 707;Vv$
PRINT Vv$
END
```

VBASe

The :MEASURE:VBASE command places the Peak Power Analyzer in the continuous measurement mode and starts a pulse base measurement.

The VBASE query measures and outputs the amplitude value at the base of the waveform. The base amplitude of a pulse is normally not the same as the minimum value.

Command Syntax: :MEASure:VBASe

Example:

```
OUTPUT 707;":MEAS:VBASE"
```

Query Syntax: :MEASure:VBASe?

Returned Format:

[:MEASure:VBASe] <value><NL>

Where:

<value> ::= amplitude at base of selected waveform (exponential–NR3 format)

Example:

```
DIM Vbs$[100]
OUTPUT 707;":MEASURE:VBASE?"
ENTER 707;Vbs$
PRINT Vbs$
END
```

VDELta?

The :MEASURE:VDELTA query outputs the amplitude difference between Marker 2 and Marker 1. No measurement is made when the VDELTA query is received by the Peak Power Analyzer. The delta amplitude value that is output is the current value. This is the same as the front panel delta amplitude value.

VDELTA = Amplitude at Marker 2−Amplitude at Marker 1

Query Syntax: :MEASure:VDELta?

Returned Format:

[:MEASure:VDELta] <value><NL>

Where:

<value> ::= delta amplitude (exponential–NR3 format; units are in volts, watts, or a ratio in dB, in log mode.)

Example:

```
DIM Vdl$[100]
OUTPUT 707;":MEAS:VDELTA?"
ENTER 707;Vdl$
PRINT Vdl$
END
```

VERTical?

The :MEASURE:VERTICAL query makes an instantaneous amplitude measurement on an untriggered signal at an unspecified time.

When AUTODIG (refer to :MEASURE:AUTODIG) is used with the VERTICAL query, measurement speed is increased. Using AUTODIG and VERTICAL together, only one point on the waveform is digitized before the measurement is made. To gain more speed, the command :DISPLAY:SCREEN must be set to "off".

Note 

Whether :DISPLAY:SCREEN is set to "on" or "off", the waveform will not be displayed when VERTICAL is used with AUTODIG.

Query Syntax: :MEASure:VERTical?

Returned Format:

[:MEASure:VERTical] <value><NL>

Where:

<value> ::= Random amplitude point (exponential–NR3 format; units are in volts or watts.)

Example:

```
DIM Vt$[100]
OUTPUT 707;":DISPLAY:SCREEN OFF"
OUTPUT 707;":MEAS:AUTODIG ON"
OUTPUT 707;":MEAS:VERTICAL?"
ENTER 707;Vt$
PRINT Vt$
END
```

VFIFty

The :MEASURE:VFIFTY command instructs the Peak Power Analyzer to find the top and base values of the specified source(s), and then places the amplitude markers at the 50% amplitude point on the specified source(s).

If only one source has been specified with the :MEASURE:SOURCE command, the VFIFty command sets both amplitude markers (Marker 1 and Marker 2) to the 50% amplitude level on that source.

If two sources are specified with the :MEASURE:SOURCE command, Marker 1 is set to the 50% level of the first specified source and Marker 2 is set to the 50% level of the second specified source.

There is no query form of this command.

Note 

If the display is in log mode, the markers are set to the 50% point in linear units (watts) and not in dBm.

Command Syntax: :MEASure:VFIFty

Example:

```
OUTPUT 707;":MEASURE:VFIFTY"
```

VMAX

The :MEASURE:VMAX command places the Peak Power Analyzer in the continuous measurement mode and starts a VMAX measurement.

The VMAX query measures and outputs the absolute maximum amplitude present on the selected waveform.

When AUTODIG (refer to :MEASURE:AUTODIG) is used with the VMAX query, measurement speed is increased. AUTODIG together with VMAX enable a complete digitize to take place before the measurement is made. The query form (VMAX?) must be used with AUTODIG to achieve the faster measurement speed. AUTODIG has no effect on the command form of VMAX. Also, for more speed, the command :DISPLAY:SCREEN must be set to "off".

Note

When the VMAX query is used with AUTODIG and :DISPLAY:SCREEN is set to "on", the digitized waveform will be displayed.

Command Syntax: :MEASure:VMAX

Example:

```
OUTPUT 707;":MEAS:VMAX"
```

Query Syntax: :MEASure:VMAX?

Note

A small difference may exist between the autodig VMAX value and the VMAX value measured on the digitized data due to resolution differences in the algorithm.

Returned Format:

[:MEASure:VMAX] <value><NL>

Where:

<value> ::= maximum amplitude of selected waveform (exponential–NR3 format)

Example:

```
DIM Max$[100]
OUTPUT 707;":DISPLAY:SCREEN OFF"
OUTPUT 707;":MEASURE:AUTODIG ON"
OUTPUT 707;":MEASURE:VMAX?"
ENTER 707;Max$
PRINT Max$
END
```

VMIN

The :MEASURE:VMIN command places the Peak Power Analyzer in the continuous measurement mode and starts a VMIN command.

The VMIN query measures and outputs the absolute minimum amplitude present on the selected waveform.

Command Syntax:

:MEASure:VMIN

Example:

```
OUTPUT 707;":MEAS:VMIN"
```

Query Syntax:

:MEASure:VMIN?

Returned Format:

[:MEASure:VMIN] <value><NL>

Where:

<value> ::= minimum amplitude value of the selected waveform (exponential–NR3 format)

Example:

```
DIM Vmn$[100]
OUTPUT 707;":MEASURE:VMIN?"
ENTER 707;Vmn$
PRINT Vmn$
END
```

VPP

Note 

This command applies to non-power measurements from channel 2 or 3, a function, or waveform memory.

The :MEASURE:VPP command places the Peak Power Analyzer in the continuous measurement mode and starts a VPP measurement.

The VPP query measures the maximum and minimum voltages for the selected source, and then, calculates the peak-to-peak voltage and outputs that value. The peak-to-peak voltage (Vpp) is calculated with the following formula:

Vpp = Vmax−Vmin

where Vmax and Vmin are the maximum and minimum voltages present on the selected source.

Command Syntax: :MEASure:VPP

Example:

```
OUTPUT 707;":MEAS:VPP"
```

Query Syntax: :MEASure:VPP?

Returned Format:

[:MEASure:VPP] <value><NL>

Where:

<value> ::= voltage peak to peak (exponential–NR3 format)

Example:

```
DIM Vp$[100]
OUTPUT 707;":MEAS:VPP?"
ENTER 707;Vp$
PRINT Vp$
END
```

VRELative

The :MEASURE:VRELATIVE command moves the amplitude markers to the specified percentage points of their last established position. The last established position is not necessarily on the currently displayed waveform.

For example, after a :MEAS:VAMPLITUDE? query has been sent, Marker 1 is located at the base (0%) of the signal and Marker 2 is at the top (100%) of the signal. If the VRELATIVE 10 command was executed, Marker 1 is moved to the 10% level and Marker 2 to the 90% level of the signal.

Any value between 0% and 100% can be used. If VRELATIVE 0 is sent, the markers are not moved because the command indicates 0% movement from the current position.

As an example, when the following values are sent, the markers are moved to the following percentage values of their current position.

10 moves Marker 1 to 10% and Marker 2 to 90%
20 moves Marker 1 to 20% and Marker 2 to 80%
50 moves both markers to 50%
90 moves Marker 1 to 10% and Marker 2 to 90%

The position of the markers starting position must be known for this command to be meaningful. The markers can be set to a known position using the :MEAS:VAMPLITUDE? query.

The VRELATIVE query returns the current relative position of Marker2 which is always in the range 50% through 90%.

Note

The VRELATIVE command does not affect the upper and lower thresholds selected by the UPPER and LOWER commands.

When the Peak Power Analyzer is in log mode, the markers are moved to linear percents, and then, the linear values are converted to dB.

Command Syntax: :MEASure:VRELative <percent>

Where:

<percent> ::= {0 through 100}

Example:

```
OUTPUT 707;":MEASURE:VRELATIVE 20"
```

Query Syntax: :MEASure:VRELative?

Returned Format:

[:MEASure:VRELative] <value><NL>

Where:

<value> ::= marker2 position in percent {50 through 100} (exponential–NR3 format)

Example:

```
DIM Vrl$[100]
OUTPUT 707;":MEAS:VREL?"
ENTER 707;Vrl$
PRINT Vrl$
END
```

VRMS

Note This command applies to non-power measurements from channel 2 or 3, a function, or waveform memory.

The :MEASURE:VRMS command places the Peak Power Analyzer in the continuous measurement mode and starts a Vrms measurement.

When a full cycle of a periodic waveform is detected, the VRMS query returns the RMS voltage over only one period of the signal not the overall RMS voltage of the displayed signal.

Note This RMS measurement is an AC RMS measurement. This means that the average value of the displayed waveform is subtracted from each data point before RMS is computed.

Command Syntax: :MEASure:VRMS

Example:

```
OUTPUT 707;":MEAS:VRMS"
```

Query Syntax: :MEASure:VRMS?

Returned Format:

[:MEASure:VRMS] <value><NL>

Where:

<value> ::= rms voltage of displayed points (exponential–NR3 format)

Example:

```
DIM Vrm$[100]
OUTPUT 707;":MEASURE:VRMS?"
ENTER 707;Vrm$
PRINT Vrm$
END
```

VSTArt

The :MEASURE:VSTART command moves Marker 1 to the specified amplitude. The values are limited to the currently defined channel, function, or waveform memory measurement source.

The VSTART query returns the current amplitude level of Marker 1.

Note

The short form of this command does not follow the defined convention. The short form "VST" is the same for VSTART and VSTOP, so, be careful not to send this form for the VSTART command. Sending "VST" will produce an error.

Command Syntax: :MEASure:VSTArt <amplitude>

Note

If the argument is in volts or watts, no units suffix is required. However, if the argument is in dBm, the suffix dBm is required.

Example:

```
OUTPUT 707;":MEAS:VSTA -10MV"
```

Query Syntax: :MEASure:VSTArt?

Returned Format:

[:MEASure:VSTArt] <value><NL>

Where:

<value> ::= amplitude at Marker 1 (exponential–NR3 format) For power measurement sources <value> will be in watts or dBm.

Example:

```
DIM VST$[100]
OUTPUT 707;":MEASURE:VSTART?"
ENTER 707;Vst$
PRINT Vst$
END
```

VSTOp

The :MEASURE:VSTOP command moves Marker 2 to the specified amplitude. The values are limited to the currently defined channel, function, or waveform memory measurement source.

The VSTOP query returns the current amplitude level of Marker 2.

Note

The short form of this command does not follow the defined convention. The short form "VST" is the same for VSTART and VSTOP, so be careful not to send this form for the VSTOP command. Sending "VST" will produce an error.

Command Syntax: :MEASure:VSTOp <amplitude>

Note

If the argument is in volts or watts, no units suffix is required. However, if the argument is in dBm, the suffix dBm is required.

Where:

<amplitude> ::= amplitude value.

Example:

```
OUTPUT 707;":MEAS:VSTO -100MV"
```

Query Syntax: :MEASure:VSTOp?

Returned Format:

[:MEASure:VSTOp] <value><NL>

Where:

<value> ::= amplitude at Marker 2 (exponential–NR3 format) For power measurement sources <value> will be in watts or dBm.

Example:

```
DIM Vst$[100]
OUTPUT 707;":MEASURE:VSTOP?"
ENTER 707;Vst$
PRINT Vst$
END
```

VTIMe?

The :MEASURE:VTIME query returns the amplitude at a specified time. The time is referenced to the trigger event. The specified time must be on screen.

When AUTODIG (refer to :MEASURE:AUTODIG) is used with the VTIME query, measurement speed is increased. AUTODIG together with VTIME enable only a partial digitize to be completed. The waveform is digitized at the specified time. To gain more speed, the command :DISPLAY:SCREEN must be set to "off".

Note

When the VTIME query is used with AUTODIG, and :DISPLAY:SCREEN is set to "on" or "off", the digitized waveform will not be displayed.

Query Syntax: :MEASure:VTIMe? <time>

Where:

<time> ::= displayed time from trigger in seconds

Returned Format:

[:MEASure:VTIMe] <value><NL>

Where:

<value> ::= amplitude at specified time (exponential–NR3 format)

Example:

```
DIM Vtm$[100]
OUTPUT 707;":DISPLAY:SCREEN OFF"
OUTPUT 707;":MEASURE:AUTODIG ON"
OUTPUT 707;":MEASURE:VTIME? .001"
ENTER 707;Vtm$
PRINT Vtm$
END
```

VTOP

The :MEASURE:VTOP command places the Peak Power Analyzer in the continuous measurement mode and starts a pulse top measurement.

The VTOP query returns the amplitude at the top of a waveform.

Command Syntax:

:MEASure:VTOP

Example:

```
OUTPUT 707;":MEASURE:VTOP"
```

Query Syntax:

:MEASure:VTOP?

Returned Format:

[:MEASure:VTOP] <value><NL>

Where:

<value> ::= amplitude at top of waveform (exponential–NR3 format)

Example:

```
DIM Vtp$[100]
OUTPUT 707;":MEASURE:VTOP?"
ENTER 707;Vtp$
PRINT Vtp$
END
```

16

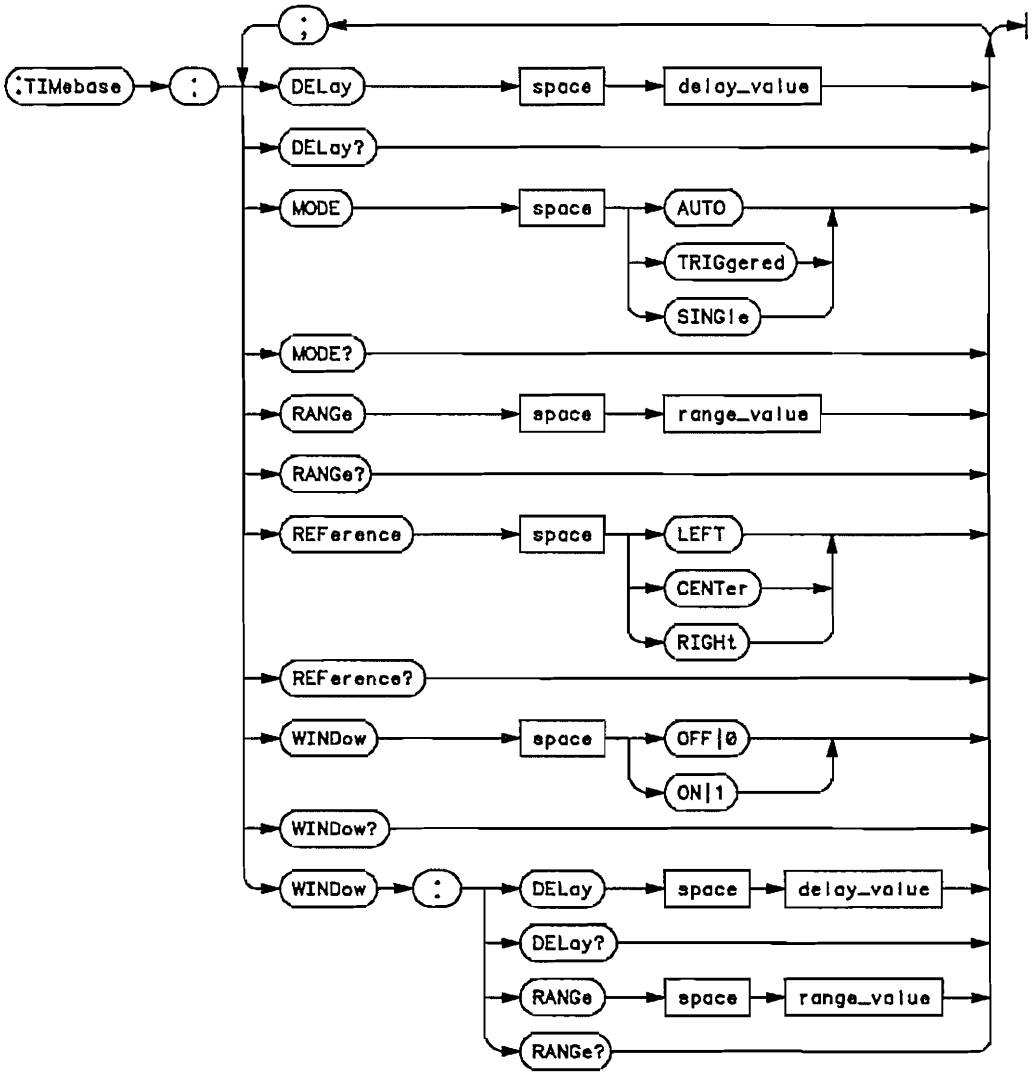
Timebase Subsystem

Introduction

The TIMEBASE subsystem commands control the horizontal axis, "X axis," Peak Power Analyzer functions.

The TIMEBASE subsystem also contains the commands that control the Timebase Window mode.

The Timebase Window mode allows a second timebase to be used. The Window settings are WINDOW:DELAY (window position) and WINDOW:RANGE (window width).



delay_value = A real number, maximum depends on sweep range.
range_value = A real number, 2 ns through 50 s (in a 1,2,5 sequence).

Figure 16-1. Timebase Subsystem Syntax Diagrams

DELay

The :TIMEBASE:DELAY command sets the timebase delay; it can be positive or negative. The delay is the time interval between the trigger event and the on screen delay reference point. The delay reference point is the left edge of the display, the right edge of the display, or center of the display. It is set with the :TIMEBASE:REFERENCE command.

The DELAY query returns the current delay value.

Command Syntax: :TIMebase:DELay <delay>

Where:

<delay> ::= time in seconds from trigger to on screen delay reference point. Maximum value depends on time/division setting.

Example:

```
OUTPUT 707;":TIM:DEL 2MS"
```

Query Syntax: :TIMebase:DELay?

Returned Format:

[:TIMebase:DELay] <delay><NL>

Where:

<delay> ::= time from trigger to display reference in seconds. Display reference is left, center or right (exponential–NR3 format)

Example:

```
DIM D1$[100]
OUTPUT 707;":TIMEBASE:DELAY?"
ENTER 707;D1$
PRINT D1$
END
```

MODE

The :TIMEBASE:MODE command selects the timebase mode. This function is the same as the Trigger Menu Trig'd/Auto key and the SINGLE key on the front panel.

If AUTO mode is selected, the unit will provide a baseline on the display in the absence of a signal. If a signal is present, but the Peak Power Analyzer is not triggered, the display will be unsynchronized but will not be a baseline. If a signal is present and the Peak Power Analyzer is triggered, the display will be synchronized.

If TRIGGERED mode is selected and no trigger is present, the unit will not sweep, and the data acquired on the previous trigger will remain on screen.

If SINGLE mode is selected, the screen will be cleared and the Peak Power Analyzer will be stopped. The RUN command, will arm the trigger, and data will be acquired when the trigger is found. To make a single acquisition, a RUN command should be sent.

The MODE query returns the current mode.

Command Syntax: :TIMebase:MODE {AUTO | TRIGgered | SINGle}

Example:

```
OUTPUT 707;":TIM:MODE TRIGgered"
```

Query Syntax: :TIMebase:MODE?

Returned Format:

[:TIMebase:MODE] <mode><NL>

<mode> ::= {AUTO | TRIGgered | SINGle}

Example:

```
DIM Mode$[100]
OUTPUT 707;"TIMEBASE:MODE?"
ENTER 707;Mode$
PRINT Mode$
END
```

RANGe

The :TIMEBASE:RANGE command sets the full scale horizontal time in seconds. The RANGE value is ten times the time per division.

The RANGE query returns the current range value.

Command Syntax: :TIMebase:RANGe <range>

Where:

<range> ::= 20 ns to 50 s in a 1,2,5 sequence.

Example:

```
OUTPUT 707;":TIM:RANG 100MS"
```

Query Syntax: :TIMebase:RANGe?

Returned Format:

[:TIMebase:RANGe] <range><NL>

Where:

<range> ::= 20 ns to 50 s (exponential–NR3 format)

Example:

```
DIM Rng$[100]
OUTPUT 707;":TIMEBASE:RANGE?"
ENTER 707;Rng$
PRINT Rng$
END
```

REFerence

The :TIMEBASE:REFERENCE command sets the delay reference point to the left side of the screen, the right side of the screen, or to the center of the screen. Delay is set with the :TIMEBASE:DELAY command.

If :TIMEBASE:DELAY is set to 0, the reference point consists of pre-trigger data to the left and post-trigger data to the right.

The REFERENCE query returns the current display reference.

Command Syntax: :TIMebase:REFerence {LEFT | CENTer | RIGHt}

Example:

```
OUTPUT 707;":TIMEBASE:REFERENCE LEFT"
```

Query Syntax: :TIMebase:REFerence?

Returned Format:

[:TIMebase:REFerence] {LEFT | CENTer | RIGHt}<NL>

Example:

```
DIM Rf$[100]
OUTPUT 707;":TIMEBASE:REFERENCE?"
ENTER 707;Rf$
PRINT Rf$
END
```

WINDow

The :TIMEBASE:WINDOW command controls whether or not the second timebase is in use. If this command is set to ON, the second timebase is displayed on the bottom half of the screen. When the Timebase Window is on, all measurements are taken on the second (expanded) timebase.

When the Timebase Window is on, the second timebase data can be acquired over the HP-IB by sending the command WAVEFORM:DATA?. However, care must be taken in order to ensure valid data is present when the data is requested.

The WINDOW query returns the current state of this command.

Command Syntax: :TIMebase:WINDow {{ON | 1} | {OFF | 0}}

Example:

```
OUTPUT 707;":TIM:WIND 1"
```

Query Syntax: :TIMebase:WINDow?

Returned Format:

[:TIMebase:WINDow] {1 | 0}

Example:

```
DIM Tw$[100]
OUTPUT 707;":TIMEBASE:WINDOW?"
ENTER 707;Tw$
PRINT Tw$
END
```

WINDow:DELay (Position)

The :TIMEBASE:WINDOW:DELAY command sets the timebase window delay. The window delay actually sets the position of the timebase window on the main sweep. The range for this command is determined by the main sweep seconds/division and delay values. The value for this command must position the window on the main sweep display.

The WINDOW:DELAY query returns the current value.

Command Syntax: :TIMebase:WINDow:DELay <wid_delay>

Where:

<wid_delay> ::= time in seconds from trigger to on screen delay reference point. Maximum value depends on time/division setting.

Example:

```
OUTPUT 707;":TIM:WIND:DEL 20N"
```

Query Syntax: :TIMebase:WINDow:DELay?

Returned Format:

[:TIMebase:WINDow:DELay] <wid_delay><NL>

Where:

<wid_delay> ::= current setting in seconds (exponential –NR3 format)

Example:

```
DIM Dly$[100]
OUTPUT 707;":TIMEBASE:WINDOW:DELAY?"
ENTER 707;Dly$
PRINT Dly$
END
```

WINDow:RANGe (Timebase)

The :TIMEBASE:WINDOW:RANGE command sets the full scale horizontal time in seconds for the second (expanded) timebase. The RANGE value is ten times the time per division of the second timebase.

The WINDOW:RANGE query returns the current range value.

Command Syntax: :TIMebase:WINDow:RANGe <range>

Where:

<range> ::= 1 ns to 50 s (Depends on the main sweep setting.)

Example:

```
OUTPUT 707;":TIM:WIND:RANG 100N"
```

Query Syntax: :TIMebase:WINDow:RANGe?

Returned Format:

[:TIMebase:WINDow:RANGe] <range><NL>

Where:

<range> ::= 1 ns to 50 s (Depends on the main sweep setting) (exponential–NR3 format)

Example:

```
DIM Wrng$[100]
OUTPUT 707;":TIMEBASE:WINDOW:RANGE?"
ENTER 707;Wrng$
PRINT Wrng$
END
```

17

Trigger Subsystem

Introduction

The commands in the Trigger subsystem are used to define the conditions for a trigger. Many of the commands are used in more than one trigger mode. If the command is a valid command for a trigger mode, that setting will be accepted. If a command does not apply to the current trigger mode, error −211 may be generated. There are some commands that apply to only one trigger mode, but are accepted under any trigger mode. One example is the :TRIGGER:DELAY:SLOPE command.

Be sure that the Peak Power Analyzer is in the proper trigger mode for the command being sent. The Peak Power Analyzer can be placed in the trigger mode from the front panel or over the bus. One method of insuring the Peak Power Analyzer is in the proper trigger mode is to send the :TRIGGER:MODE command in the same program message as the parameter being set. As an example, the following program message could be used:

```
OUTPUT 707;":TRIGGER:MODE EDGE;LEVEL 200 MV"
```

This will place the Peak Power Analyzer in the Edge Trigger Mode and set the trigger level to 200 mV. This is necessary because the LEVEL command is also valid for the other trigger modes.

On the next few pages, the Trigger modes are described prior to the command syntax. Table 17-1 lists the different TRIGGER subsystem commands that are available for each trigger mode.

Note Auto or triggered mode is selected with the TIMEBASE:MODE command.

Table 17-1.

EDGE	PATTERN	STATE	DELAY
HOLDOFF	CONDITION	CONDITION	CONDITION
LEVEL	HOLDOFF	HOLDOFF	DELAY
SLOPE	LEVEL	LEVEL	DELAY:SLOPE
SOURCE	LOGIC	LOGIC	DELAY:SOURCE
	PATH	PATH	LEVEL
		SLOPE	LOGIC
		SOURCE	OCCURRENCE
			OCCURRENCE:SLOPE
			OCCURRENCE:SOURCE
			PATH
			QUALIFY
			SLOPE
			SOURCE

Note Auto or triggered mode is selected with the TIMEBASE:MODE command.

The EDGE Trigger Mode

Edge Trigger Mode is the easiest trigger mode to understand and use from the front panel or over the bus. This is true because Edge Trigger Mode has the least number of parameters to set. This explanation of the edge trigger commands follows the front panel keys very closely. The parameters for an edge trigger do not have to be set in the order described below. Refer to the *Operating Manual* for further explanation of edge trigger operation.

In edge trigger mode, you set the trigger source using the TRIGGER:SOURCE command. This selects the channel on which the Peak Power Analyzer is triggered. The argument for this command is channels 1 through 4.

The next field to be set in this mode is the trigger level. This value is set using the LEVEL command. A different LEVEL can be set for each trigger source. The trigger level values that are set in this mode are used for all modes. Conversely, the LEVELS set in the PATTERN, STATE, or DELAY modes set the EDGE LEVELS as well.

The trigger level in the PATTERN and STATE modes determines if the input amplitude is a logic high or a logic low for the logic triggers.

The next field to be set in the Edge Trigger Mode is the actual edge that creates the trigger. This command is the SLOPE command and can be set to POSITIVE or NEGATIVE for each of the four sources.

The last setting in this mode is the Trigger Holdoff value. This value is only used for EDGE mode; the edge trigger holdoff value is not carried over to the other trigger modes.

The following program could be used to set up an Edge Trigger:

```
10 OUTPUT 707;":TIMEBASE:MODE TRIGGERED"
20 !Peak Power Analyzer is set to triggered mode.
30 OUTPUT 707;":TRIGGER:MODE EDGE"
40 !Sets triggering to edge trigger.
50 OUTPUT 707;":TRIGGER:SOURCE CHANNEL1"
60 OUTPUT 707;":TRIGGER:LEVEL 8M"
70 OUTPUT 707;":TRIGGER:SLOPE POSITIVE"
80 OUTPUT 707;":TRIGGER:HOLDOFF TIME,40N"
90 END
```

The Pattern Trigger Mode

Pattern Trigger Mode is used to define a four-character pattern for the Peak Power Analyzer to recognize and then generate a trigger event. When the inputs satisfy the trigger pattern and conditions, the Peak Power Analyzer triggers and displays the desired portion of the waveform.

Pattern mode is very useful for glitch detection, because the Peak Power Analyzer triggers on a glitch and displays the resulting waveform.

The following description of Pattern Trigger Mode is also related to the front panel keys. There are additional parameters in this mode that are not used in Edge Trigger Mode and one parameter that is carried over from edge trigger mode.

The one parameter that is carried over is LEVEL. If the level command is used in this mode, it changes the level value for that source in Edge Trigger Mode.

The logic pattern for Pattern Trigger Mode is set using the PATH and LOGIC commands. The PATH command specifies which of the inputs is selected for the logic pattern. Once the path has been selected, the pattern can be set using the LOGIC command. The LOGIC command uses the arguments HIGH, LOW, and DONTCARE to set the "trigger on" bit pattern.

The next field on the front panel sets "when". This is set with the CONDITION command. This command is used in several of the trigger modes, therefore, it has parameters that are not valid in this mode. The valid parameters for the CONDITION command in Pattern Trigger Mode are the following: ENTER, EXIT, GT <value>, LT <value>, RANGE,<value>,<value>.

When the command TRIGGER:CONDITION ENTER or TRIGGER:CONDITION EXIT is sent, the Entered or Exited parameter is set on the front panel. When

the GT or LT option is used, a time value must be sent to define the limit. When the RANGE option is used, two time values must be sent to define the lower and upper limits. The correct syntax for the RANGE option is TRIGGER:CONDITION RANGE,<range_low>,<range_high>.

As in the Edge Trigger Mode, you can also set the holdoff time using the TRIGGER:HOLDOFF command.

The State Trigger Mode

State Trigger Mode is similar to pattern trigger mode except that one channel is selected as the clock edge and the other three channels define a pattern. When the pattern becomes true, the Peak Power Analyzer triggers on the next clock edge, if the pattern meets setup and hold criteria.

When State Trigger Mode is selected, the TRIGGER:SOURCE command selects the clock source. An example of selecting the clock source is shown below:

```
OUTPUT 707;":TRIGGER:SOURCE CHANNEL2"
```

The correct edge for the clock is selected using the TRIGGER:SLOPE command which is set to NEGATIVE or POSITIVE. The following is an example of selecting the slope:

```
OUTPUT 707;":TRIGGER:SLOPE POSITIVE"
```

The TRIGGER:PATH command is used with the TRIGGER:LOGIC command to set the three bit logic pattern to qualify the clock trigger. The following is an example of the PATH and LOGIC commands:

```
OUTPUT 707;":TRIGGER:PATH CHAN2"
OUTPUT 707;":TRIGGER:LOGIC LOW"
```

The TRIGGER:CONDITION command sets the "is/is not present" state using the parameters TRUE for "is present" and FALSE for "is not present." An example of the CONDITION command is presented below:

```
OUTPUT 707;":TRIGGER:CONDITION TRUE"
```

In this mode, as in most other modes, a holdoff value can be set.

The Delay Trigger Mode

Delay Trigger mode qualifies on a signal edge, pattern, or state. Delay is for a period of time or occurrence of edges. Triggering is on a selected edge from any source.

This trigger mode is versatile and accommodates most complex triggering situations. It has the flexibility to select different trigger sources and delay times or delay counts. Various points of the waveform are then displayed.

In Delay Trigger Mode, the TRIGGER:QUALIFY command is used to select EDGE, PATTERN, or STATE mode as a qualifier. When the EDGE qualifier is selected, all Edge parameters and commands are used to set the source and slope. When the PATTERN qualifier is selected, all Pattern commands are used to set the pattern mode parameters. When the STATE qualifier is selected, all State commands are used to set the state mode parameters.

The DELAY command is used to set the time or count parameter and the amount of delay. To set the delay to time, use the command TRIGGER:DELAY TIME. To set the delay to count, use the command TRIGGER:DELAY EVENT. The following examples could be used to set the delay time or delay count:

```
OUTPUT 707;":TRIGGER:DELAY TIME,40N"
OUTPUT 707;":TRIGGER:DELAY EVENT,50"
```

If the trigger delay is set to EVENT (count), you then set the delay source and slope. To set the delay source, use the command :TRIGGER:DELAY:SOURCE. To set the delay slope, use the command :TRIGGER:DELAY:SLOPE. The following examples present setting the source and slope:

```
OUTPUT 707;":TRIGGER:DELAY:SOURCE CHANNEL1"
OUTPUT 707;":TRIGGER:DELAY:SLOPE POSITIVE"
```

The values of "trigger on" are set with the OCCURRENCE command. To set the number of occurrences, the following example could be used:

```
OUTPUT 707;":TRIGGER:OCCURRENCE 3333"
```

To set the source for the number of occurrences, the example below might be used:

```
OUTPUT 707;":TRIGGER:OCC:SOURCE CHAN2"
```

To set the slope of the trigger occurrence, this example could be used:

```
OUTPUT 707;":TRIGGER:OCC:SLOPE NEGATIVE"
```

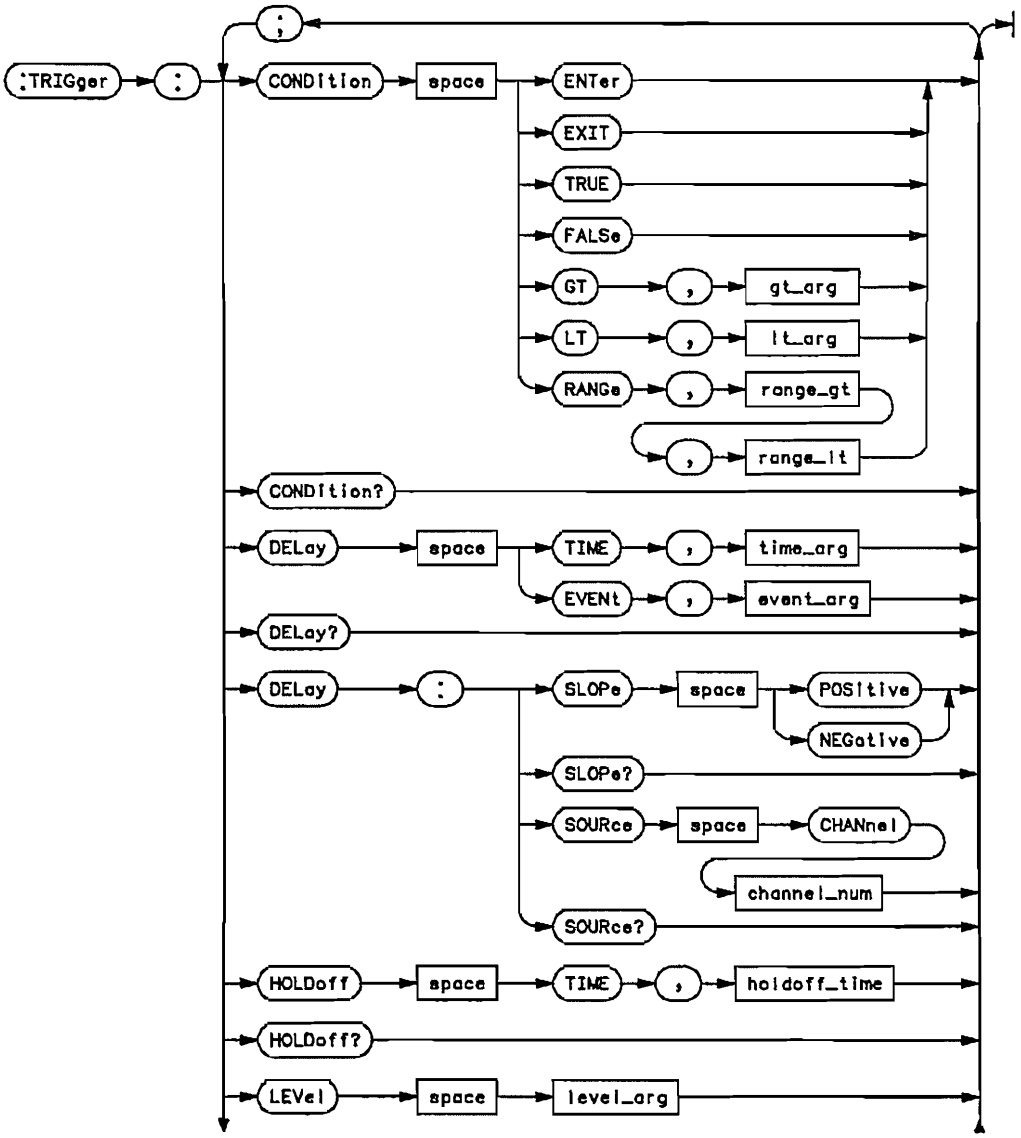


Figure 17-1. Trigger Subsystem Syntax Diagrams

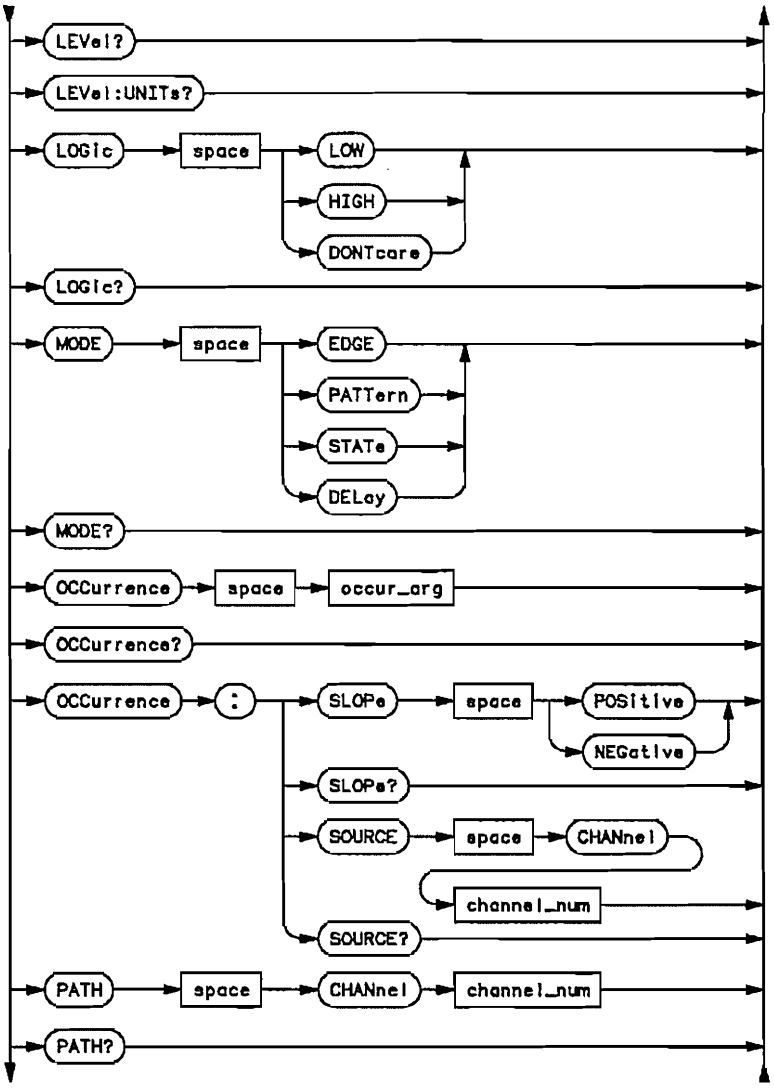
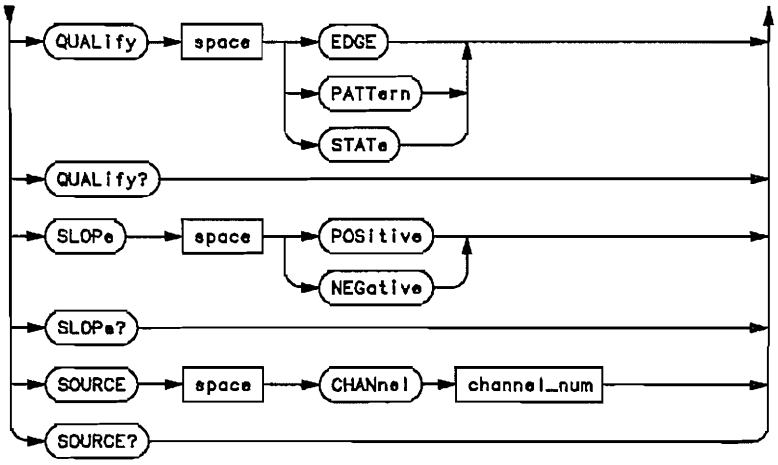


Figure 17-1. Trigger Subsystem Syntax Diagrams (continued)



channel_num = An integer, 1, 2, 3, or 4.
event_arg = An integer, 1 to 16000000.
gt_arg = A time value, 20 ns to 160 ms.
holdoff_time = A time value, 40 ns to 320 ms.
level_arg = A real number, specifying the trigger level.
lt_arg = A time value, 20 ns to 160 ms.
range_gt = A time value, 20 ns to 159.999 ms (value must be less than range_lt).
range_lt = A time value, 30 ns to 160 ms (value must be greater than range_gt).
time_arg = A time value, 30 ns to 160 ms.
occur_arg = An integer, 1 to 16000000.

Figure 17-1. Trigger Subsystem Syntax Diagrams (continued)

CONDition

The :TRIGGER:CONDITION command is valid in the PATTERN, STATE, and DELAY trigger modes. The function of the CONDITION command in each of these modes is described below.

Time values entered using this command are rounded to the nearest 10 ns.

In **Pattern Trigger Mode**, the valid arguments for CONDITION are ENTER, EXIT, GT, LT, and RANGE. The command is used to specify if the trigger will be generated upon entry to the specified logic pattern, upon exiting the specified logic pattern, or if the pattern must be present for a specified amount of time. The time in pattern trigger mode can be specified to be greater than a value (GT), less then a value (LT), or between two values (RANGE). These are the same settings that are specified using the front panel "when" key in Pattern Trigger Mode.

In **State Trigger Mode**, the valid parameters for the CONDITION command are TRUE (is present) and FALSE (is not present).

In **Delay Trigger Mode**, the CONDITION command is valid when PATTERN or STATE is selected as the qualifier. All arguments that are valid in the Pattern or State Trigger Modes are valid here.

The CONDITION query returns the currently selected condition for the currently selected mode.

Command Syntax:

:TRIGger:CONDition <argument>

Where in **PATTERN or DELAY:PATTERN** mode:

<argument> ::= {ENTer | EXIT | GT,<value> | LT,<value> | RANGe,<range_gt>,<range_lt>}

Where in **STATE or DELAY:STATE** mode:

<argument> ::= {TRUE | FALSe}

Where:

<value> ::= 20 ns to 160 ms
<range_gt> ::= 20 ns to 159.999 ms (must be less than range_lt)
<range_lt> ::= 30 ns to 160 ms (must be greater than range_gt)

Example:

```
OUTPUT 707;":TRIG:COND RANGE,22ms,33ms"
```

Query Syntax:

:TRIGger:CONDition?

Returned Format:

[:TRIGger:CONDition] <argument><NL>

Where in **PATTERN or DELAY PATTERN** mode:

<argument> ::= {ENTer | EXIT | GT,<value> | LT,<value> | RANGe,<range_gt>,<range_lt>}

Where in **STATE or DELAY STATE** mode:

<argument> ::= {TRUE | FALSe}

Where:

<value> ::= 20 ns to 160 ms
<range_gt> ::= 20 ns to 159.999 ms (must be less than range_lt)
<range_lt> ::= 30 ns to 160 ms (must be greater than range_gt)

Example:

```
DIM Con$[100]
OUTPUT 707;":TRIGGER:CONDITION?"
ENTER 707;Con$
PRINT Con$
END
```

DELay

The :TRIGGER:DELAY command only applies to the **Delay Trigger Mode**. This command allows you to set a delay value in either time or number of events. In the time delay mode, this command specifies the delay value in seconds. In the events delay mode, this command specifies the delay value in number of trigger events.

The Delay query returns the current delay setting.

Command Syntax: :TRIGger:DELay {TIME,<time_value> | EVENt,<event_value>}

Where:

<time_value> ::= time of delay from 30 ns to 160 ms
<event_value> ::= number of events from 1 to 16000000

Example:

```
OUTPUT 707;":TRIGGER:DELAY TIME,1.23E-01"
```

Query Syntax: :TRIGger:DELay?

Returned Format:

[:TRIGger:DELay] {TIME,<time_value> | EVENt,<event_value>}<NL>

Where:

<time_value> ::= time of delay from 30 ns to 160 ms
<event_value> ::= number of events from 1 to 16000000

Example:

```
DIM Value$[100]
OUTPUT 707;":TRIG:DELAY?"
ENTER 707;Value$
PRINT Value$
END
```

DELay:SLOPe

The :TRIGGER:DELAY:SLOPE command sets the edge that is counted by the delay command. The parameters for this command are NEGATIVE or POSITIVE. This command only applies to the **Delay Trigger Mode**. However, the command is accepted in any trigger mode.

The DELAY:SLOPE query returns the current delay slope.

Command Syntax: :TRIGger:DELay:SLOPe {POSitive | NEGative}

Example:

```
OUTPUT 707;":TRIG:DEL:SLOP POS"
```

Query Syntax: :TRIGger:DELay:SLOPe?

Returned Format:

[:TRIGger:DELay:SLOPe] {POSitive | NEGative}<NL>

Example:

```
DIM Tos$[100]
OUTPUT 707;"TRIGGER:DELAY:SLOP?"
ENTER 707;Tos$
PRINT Tos$
END
```

DELay:SOURce

The :TRIGGER:DELAY:SOURCE command sets the source that is counted by the delay command. The parameters for this command are CHANNEL1 through CHANNEL4. This command only applies to the **Delay Trigger Mode.**

The DELAY:SOURCE query returns the source of the delay in the Delay Trigger Mode.

Note

When Option 001, Single Sensor Input, is installed, channel 4 is not useable. No error is reported when the source is specified to be channel 4. Even though no error is reported, the Peak Power Analyzer won't trigger on the measured waveform.

Command Syntax: :TRIGger:DELay:SOURce {CHANnel1 | CHANnel2 | CHANnel3 | CHANnel4}

Example:

```
OUTPUT 707;":TRIG:DEL:SOURCE CHANNEL2"
```

Query Syntax: :TRIGger:DELay:SOURce?

Returned Format:

[:TRIGger:DELay:SOURce] {CHANnel1 | CHANnel2 | CHANnel3 | CHANnel4}<NL>

Example:

```
DIM Tos$[100]
OUTPUT 707;"TRIGGER:DELAY:SOUR?"
ENTER 707;Tos$
PRINT Tos$
END
```

HOLDoff

The :TRIGGER:HOLDOFF command is valid in the **Edge, Pattern, and State Trigger Modes.** :TRIGGER:HOLDOFF disables the trigger circuit for a selectable time period (in seconds) after the trigger event. The holdoff in one trigger mode IS NOT carried over to another trigger mode.

The HOLDOFF query returns the value of the holdoff for the current mode.

Command Syntax:

:TRIGger:HOLDoff TIME, <holdoff_value>

Where:

<holdoff_value> ::= 40 ns to 320 ms rounded to nearest 20 ns increment.

Example:

```
OUTPUT 707;":TRIGGER:HOLDOFF TIME,216U"
```

Query Syntax: :TRIGger:HOLDoff?

Returned Format:

[:TRIGger:HOLDoff] TIME, <holdoff_value><NL>

Where:

<holdoff_value> ::= 40 ns to 320 ms (exponential–NR3 format)

Example:

```
DIM Ho$[100]
OUTPUT 707;":TRIGGER:HOLD?"
ENTER 707;Ho$
PRINT Ho$
END
```

LEVel

The :TRIGGER:LEVEL command sets the trigger level of the active trigger. The level is in volts for channels 2 and 3 and in watts or dBm for channels 1 and 4. This command can be used with any trigger mode.

Note

If you are in Pattern Trigger Mode and set a level for the active trigger, that level is used for the Edge, State, and Delay Trigger Modes.

The LEVEL query returns the trigger level of the current trigger mode.

Command Syntax: :TRIGger:LEVel <level>

Note

If <level> is in volts or watts, no units suffix is required. However, if <level> is in dBm, the suffix dBm is required. For channels 1 and 4, the Peak Power Analyzer will convert the <level> to the current units.

For channels 2 and 3:

<level> ::= Maximum and minimum values are dependent on HP-IB commands :CHANNEL:RANGE, :CHANNEL:OFFSET, and :CHANNEL:PROBE. For more information, refer to the Channel Subsystem concerning setting these parameters.

For channels 1 and 4:

<level> ::= <level> can be in watts or dBm. Maximum and minimum values are dependent on HP-IB commands :CHANNEL:RANGE and :CHANNEL:FACTOR. For

more information, refer to the Channel Subsystem concerning setting these parameters.

Examples:

```
OUTPUT 707;":TRIGGER:LEVEL .3"    !(channel 2,3)
OUTPUT 707;":TRIGGER:LEV 300M"    !(channel 2,3)
OUTPUT 707;":TRIG:LEV 300E-03"    !(channel 2,3)

OUTPUT 707;":TRIGGER:LEVEL .1"    !(channel 1,4)
OUTPUT 707;":TRIGGER:LEV 100M"    !(channel 1,4)
OUTPUT 707;":TRIG:LEV 100E-03"    !(channel 1,4)
```

Query Syntax:

:TRIGger:LEVel?

Returned Format:

[:TRIGger:LEVel] <level><NL>

Where:

<level> ::= trigger level in volts, watts, or dBm (exponential–NR3 format) Refer to the :TRIGGER:LEVEL:UNITS? command to query the trigger level units.

Example:

```
DIM Tlevel$[100]
OUTPUT 707;":TRIGGER:LEVEL?"
ENTER 707;Tlevel$
PRINT Tlevel$
END
```

LEVel:UNITs?

The LEVEL:UNITS? query returns the units of the current trigger level.

Query Syntax: :TRIGger:LEVel:UNITs?

Returned Format:

[:TRIGger:LEVel:UNITs] <units><NL>

Where:

<units> ::= units associated with a :TRIGGER:LEVEL? query. The units returned are volts (V), watts (W), or dBm.

Example:

```
DIM Unit$[100]
OUTPUT 707;":TRIGGER:LEVEL:UNITS?"
ENTER 707;Unit$
PRINT Unit$
END
```

LOGic

The :TRIGGER:LOGIC command applies to the **Pattern and State Trigger Modes**, as well as the **DELAY Trigger Mode** when qualifying by PATTERN or STATE. The LOGIC command is used to specify the relation between the signal and the defined amplitude level that must exist before that part of the pattern is considered valid. If the signal on a selected path is greater than the trigger level, that signal is considered HIGH. If it is less than the trigger level, it is considered LOW. The signal path is selected with the :TRIGGER:PATH command.

The LOGIC query returns the last specified logic level of the currently enabled path.

Command Syntax: :TRIGger:LOGic {HIGH | LOW | DONTcare}

Example:

```
OUTPUT 707;":TRIG:LOGIC DONT"
```

Query Syntax: :TRIGger:LOGic?

Returned Format:

[:TRIGger:LOGic] {HIGH | LOW | DONTcare}<NL>

Example:

```
DIM L$[100]
OUTPUT 707;":TRIGGER:LOGIC?"
ENTER 707;L$
PRINT L$
END
```

MODE

The :TRIGGER:MODE command selects the trigger mode.

The MODE query returns the currently selected trigger mode.

Command Syntax: :TRIGger:MODE {EDGE | PATTern | STATe | DELay}

Example:

```
OUTPUT 707;":TRIGGER:MODE PATT"
```

Query Syntax: :TRIGger:MODE?

Returned Format:

[:TRIGger:MODE] {EDGE | PATTern | STATe | DELay}<NL>

Example:

```
DIM Mode$[100]
OUTPUT 707;":TRIGGER:MODE?"
ENTER 707;Mode$
PRINT Mode$
END
```

OCCurrence

The :TRIGGER:OCCURRENCE command sets the number of trigger events that must occur before the Peak Power Analyzer sweep is actually triggered. This command applies to the **Delay Trigger Mode.**

The OCCURRENCE query returns the current value of occurrence if the Peak Power Analyzer is in the Delay Trigger Mode. Error −420, Query Unterminated is returned if Delay Trigger is not the current mode.

Command Syntax: :TRIGger:OCCurrence <occ_number>

Where:

<occ_number> ::= 1 to 16000000

Example:

```
OUTPUT 707;":TRIGGER:OCC 14"
```

Query Syntax: :TRIGger:OCCurrence?

Returned Format:

[:TRIGger:OCCurrence] <occ_number><NL>

Where:

<occ_number> ::= 1 to 16000000 (integer–NR1 format)

Example:

```
DIM Oc$[100]
OUTPUT 707;":TRIGGER:OCCURRENCE?"
ENTER 707;Oc$
PRINT Oc$
END
```

OCCurrence: SLOPe

The :TRIGGER:OCCURRENCE:SLOPE command sets the edge that is counted by the occurrence command. The parameters for this command are NEGATIVE or POSITIVE. This command applies to the **Delay Trigger Mode**.

The OCCURRENCE:SLOPE query returns the current selected slope.

Command Syntax: :TRIGger:OCCurrence:SLOPe {POSitive | NEGative}

Example:

```
OUTPUT 707;":TRIG:OCC:SLOP POS"
```

Query Syntax: :TRIGger:OCCurrence:SLOPe?

Returned Format:

[:TRIGger:OCCurrence:SLOPe {POSitive | NEGative}<NL>

Example:

```
DIM Tos$[100]
OUTPUT 707;"TRIGGER:OCCURRENCE:SLOP?"
ENTER 707;Tos$
PRINT Tos$
END
```

OCCurrence: SOURce

The :TRIGGER:OCCURRENCE:SOURCE command sets the source for the occurrence command. The parameters for this command are CHANNEL1 through CHANNEL4. This command applies to the **Delay Trigger Mode**.

The OCCURRENCE:SOURCE query returns the source of the occurrence in the Delay Trigger Mode.

Note

When Option 001, Single Sensor Input, is installed, channel 4 is not useable. No error is reported when the source is specified to be channel 4. Even though no error is reported, the Peak Power Analyzer won't trigger on the measured waveform.

Command Syntax: :TRIGger:OCCurrence:SOURce {CHANnel1 | CHANnel2 | CHANnel3 | CHANnel4}

Example:

```
OUTPUT 707;":TRIG:OCC:SOURCE CHANNEL2"
```

Query Syntax: :TRIGger:OCCurrence:SOURce?

Returned Format:

[:TRIGger:OCCurrence:SLOPe] {CHANnel1 | CHANnel2 | CHANnel3 | CHANnel4}<NL>

Example:

```
DIM Tos$[100]
OUTPUT 707;"TRIGGER:OCCURRENCE:SOUR?"
ENTER 707;Tos$
PRINT Tos$
END
```

PATH

The :TRIGGER:PATH command applies in the **Pattern Trigger Mode, State Trigger Mode,** and **Delay Trigger Mode** when "qualify on" pattern or state is selected. This command selects a pattern bit as the source for future :TRIGGER:LOGIC commands.

The PATH query returns the current trigger source of the present mode.

Note

When Option 001, Single Sensor Input, is installed, channel 4 is not useable. No error is reported when the path is specified to be channel 4. Even though no error is reported, the Peak Power Analyzer won't trigger on the measured waveform.

Command Syntax: :TRIGger:PATH <path_name>

Where:

<path_name> ::= {CHANnel1 | CHANnel2 | CHANnel3 | CHANnel4}

Example:

```
OUTPUT 707;":TRIGGER:PATH CHANNEL2"
```

Query Syntax: :TRIGger:PATH?

Returned Format:

[:TRIGger:PATH] CHANnel{1 | 2 | 3 | 4}<NL>

Example:

```
DIM Tp$[100]
OUTPUT 707;":TRIG:PATH?"
ENTER 707;Tp$
PRINT Tp$
END
```

QUALify

The :TRIGGER:QUALIFY command applies to the **Delay Trigger Mode.** The parameters for this command are listed below:

- EDGE
- PATTERN
- STATE

The QUALIFY query returns the current setting of the QUALIFY command.

Command Syntax: :TRIGger:QUALify <qualify_parameter>

Where:

<qualify_parameter> ::= {EDGE | PATTern | STATe}

Example:

```
OUTPUT 707;":TRIGGER:QUALIFY PATT"
```

Query Syntax: :TRIGger:QUALify?

Returned Format:

[:TRIGger:QUALify] {EDGE | PATTern | STATe}<NL>

Example:

```
DIM Tq$[100]
OUTPUT 707;":TRIG:QUALIFY?"
ENTER 707;Tq$
PRINT Tq$
END
```

SLOPe

The :TRIGGER:SLOPE command specifies the slope of the edge for the trigger. The SLOPE command applies to the **Edge Trigger Mode**, **State Trigger Mode**, and **Delay Trigger Mode** when EDGE or STATE is selected as the qualifier.

The SLOPE query returns the current slope for the currently selected trigger mode.

Command Syntax: :TRIGger:SLOPe {NEGative | POSitive}

Example:

```
OUTPUT 707;":TRIGGER:SLOPE POSITIVE"
```

Query Syntax: :TRIGger:SLOPe?

Returned Format:

[:TRIGger:SLOPe] {POSitive | NEGative}<NL>

Example:

```
DIM Ts$[100]
OUTPUT 707;":TRIG:SLOP?"
ENTER 707;Ts$
PRINT Ts$
END
```

SOURce

The :TRIGGER:SOURCE command selects the channel that produces the trigger. The SOURCE command applies to the **Edge Trigger Mode, State Trigger Mode, and Delay Trigger Mode.** In the Delay Trigger Mode, this command is valid when EDGE or STATE is selected as the qualifier.

The SOURCE query returns the current source for the selected trigger mode.

Note

When Option 001, Single Sensor Input, is installed, channel 4 is not useable. No error is reported when the trigger source is specified to be channel 4. Even though no error is reported, the Peak Power Analyzer won't trigger on the measured waveform.

Command Syntax: :TRIGger:SOURce {CHANnel1 | CHANnel2 | CHANnel3 | CHANnel4}

Example:

```
OUTPUT 707;":TRIGGER:SOURCE CHAN2"
```

Query Syntax: :TRIGger:SOURce?

Returned Format:

[:TRIGger:SOURce] {CHANnel1 | CHANnel2 | CHANnel3 | CHANnel4}<NL>

Example:

```
DIM Src$[100]
OUTPUT 707;":TRIGGER:SOURCE?"
ENTER 707;Src$
PRINT Src$
END
```

18

Waveform Subsystem

Introduction

The WAVEFORM subsystem is used to transfer waveform data between a controller and the Peak Power Analyzer's waveform memories. The waveform record is actually contained in two portions, the waveform data and the preamble. The waveform data is the actual data acquired for each point in the specified source. The preamble contains the information for interpreting the waveform data. This includes the number of points acquired, format of acquired data, and type of acquired data. The preamble also contains the X and Y increments, origins, and references for the acquired data, so that the raw data can be translated to time and amplitude values.

The values set in the preamble are determined when the :DIGITIZE command is executed on a channel or the front panel Waveform Memory STORE key is pressed. The preamble values are based on the settings of variables in the ACQUIRE subsystem or the front panel setup if the store key is pressed. Although the preamble values can be changed with a controller, the way that the data was acquired cannot be changed. Changing the preamble values cannot change the type of data that was actually acquired, the number of points actually acquired, etc. Therefore, when changing any waveform preamble values extreme caution must be used to ensure the data will still be useful. For example, setting POINTS in the preamble to a value different from the actual number of points in the waveform will result in inaccurate data.

The waveform data and preamble must each be read (by the controller) or sent (to the Peak Power Analyzer) with two separate commands: DATA and PREAMBLE.

If more than one channel is to be digitized, digitize the desired channels with a single :DIGITIZE command. Refer to the Root Level Command :DIGITIZE for more information.

Data Acquisition Types

There are three types of waveform acquisition that can be selected with the :ACQUIRE:TYPE command. The three types are NORMAL, AVERAGE, and ENVELOPE. When the data is acquired using the DIGITIZE command, the data is placed in the channel buffer of the specified source.

After a DIGITIZE command, the Peak Power Analyzer is stopped. If the Peak Power Analyzer is restarted (over the bus or from the Front Panel) the data acquired with the DIGITIZE command is overwritten.

Note

The on-screen time is divided into a specific number of horizontal time points as defined by the :ACQUIRE:POINTS command. Each of these increments in time is referred to as a time bucket with each time bucket having a fixed time associated with it.

Normal

Normal data consists of the last data point (hit) in each time bucket. This data is transmitted over HP-IB in a linear fashion starting with time bucket 0 and going through time bucket n-1, where n is the number returned by the WAVEFORM:POINTS query. Time buckets that don't have data in them return −1. Only the magnitude values of each data point are transmitted, the time values correspond to the position in the data array. The first amplitude value corresponds to the first time bucket on the left of the CRT, and the last value corresponds to the next to last time bucket on the right side of the CRT.

Average Average data consists of the average of n hits in a time bucket, where n is the value returned by the ACQUIRE:COUNT query. Time buckets that have fewer than n hits return the average of what data they do have. If the :ACQUIRE:COMPLETE parameter is set to 100%, then, each time bucket must contain the number of data hits specified with the :ACQUIRE:COUNT command. Again, if a time bucket doesn't have any data in it, it will return −1. This data is transmitted over the bus in a linear fashion starting with time bucket 0 and proceeding through time bucket n-1, where n is the number returned by the WAVEFORM:POINTS query. The first value corresponds to a point at the left side of the screen, and the last value is one point away from the right side of the screen.

Envelope Envelope data consists of two arrays of data, one containing the minimum of at least n hits in each time bucket and the other containing the maximum of at least n hits in each time bucket, where n is the value returned by the ACQUIRE:COUNT query. If a time bucket does not have any hits in it, then −1 is returned for both the minimum and maximum values. The two arrays are transmitted one at a time over the bus linearly, starting with time bucket 0 (on the left side of the CRT) and proceeding through time bucket m-1, where m is the value returned by the WAVEFORM:POINTS query. The array with the minimum values is sent first. The first value of each array corresponds to the data point on the left of the CRT. The last value is one data point away from the right side of the CRT.

The data is transferred from the channel buffer to a controller using the WAVEFORM:DATA query.

Data is transferred into the Peak Power Analyzer from a controller using the WAVEFORM:DATA command. Envelope data will be transferred into Waveform

Memories 1 and 3 , if WMEMORY 1 is specified as the source, or Waveform Memories 2 and 4 if WMEMORY 2 is specified as the source. The data is then transferred as two arrays. If Waveform Memory 1 is specified as the source, the first array is transferred into Waveform Memory 1 and the second array is transferred into Waveform Memory 3. If Waveform Memory 2 is specified as the source, the first array is transferred in Waveform Memory 2, and the second array is transferred into Waveform Memory 4. The data type is then changed to normal for each of the waveform memories.

Data Conversion

Data sent from the Peak Power Analyzer is raw data and must be scaled for useful interpretation. The values used to interpret the data are the X and Y references, X and Y origins, and X and Y increments. These values are read from the waveform preamble.

Note

Waveform data units for power quantities are always in watts, never in dBm.

Conversion from Data Value to Amplitude

The formula to convert a data value from waveform memories 1–4 to an amplitude value is:

amplitude = [(data value−yreference)(yincrement)] + yorigin

Conversion from Data Value to Time

The time value of a data point can be determined by the position of the data point. As an example, the third data point sent with XORIGIN = 16 ns, XREFERENCE = 0, and XINCREMENT = 2 ns. Using the formula

time = [((data point number−1) − xreference)(xincrement)] + xorigin

would result in the following calculation:

time = [((3 − 1)−0) * 2 ns] + 16 ns = 22 ns.

Data Format for HP-IB Transfer

There are four formats for transferring waveform data over the bus. These formats are WORD, BYTE, COMPRESSED, and ASCII.

WORD, BYTE, and COMPRESSED formatted waveform records are transmitted using the arbitrary block program data format specified in IEEE 488.2. ASCII format block data does not use a block header.

When you use this format, the ASCII character string "#8<DD ... D>" is sent before the actual data. The 8 indicates how many <D>'s will follow. The <D>'s are ASCII numbers which indicate how many data bytes will follow.

For example, if 512 points were acquired the Block Header "#800000512" would be sent. The 8 indicates that eight length bytes follow, and 512 indicates that 512 data bytes (binary) follow.

WORD Format

In the WORD format the number of data bytes is twice the number of words (data points). The number of data points is the value returned by the :WAVEFORM:POINTS? query. The number of data bytes is followed by a sequence of bytes representing the data points, with the most significant byte of each word transmitted first. In this format, the data is shifted so that the most significant bit after the sign bit contains the most significant bit of the data. If there is a hole in the data, it will be represented by the 2s-compliment 16-bit value of −1. The range of data in the WORD format is from 0 to 32640.

WORD format is useful in applications where the information is read directly into an integer array in a controller.

WORD and ASCII formatted data preserves the full data resolution values.

BYTE Format

BYTE format allows only seven bits to be used to represent the amplitude values with the first bit being the sign bit. If there is a hole in the data, it is represented by a 2s-compliment 8-bit value of −1.

BYTE formatted data will transfer over the bus faster than WORD formatted data since one byte per point is transferred in BYTE format and two bytes per point are transferred in WORD format. BYTE formatted data has less resolution than WORD formatted data.

COMPRESSED Format

The number of bytes transmitted when the format is COMPRESSED is the same as the value returned by the WAVEFORM:POINTS? query.

Eight bits of resolution are retained in the COMPRESSED format. So that a hole in the data may be represented, a data value of 255 is mapped to 254, and 255 is used to represent a hole. This mode will give greater vertical precision than BYTE formatted data, with faster transfer times than WORD formatted data, but will probably require more time once transferred to be unpacked.

ASCII Format

ASCII formatted waveform records are transmitted one value at a time, separated by a comma. The data values transmitted are the same as the values sent in the WORD FORMAT except that they are converted to an integer ASCII format (six or less characters) before being sent over the HP-IB.

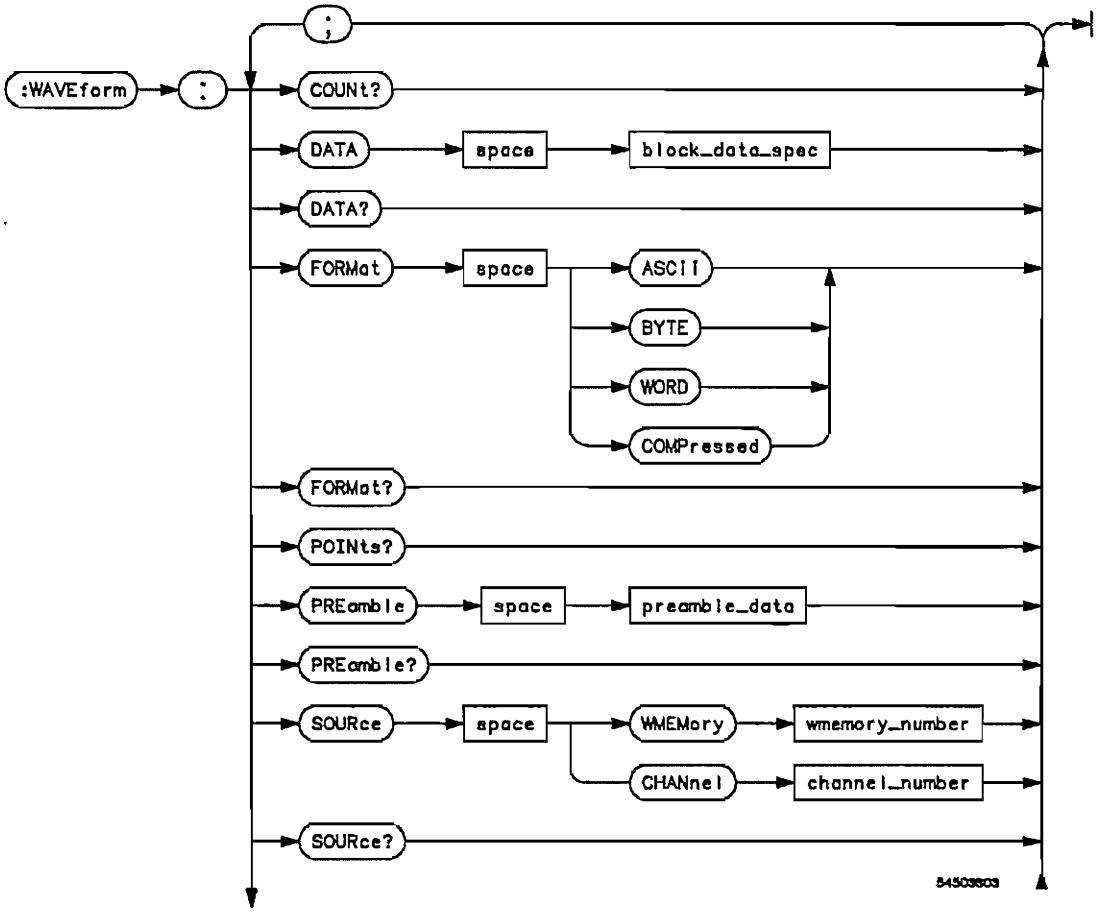
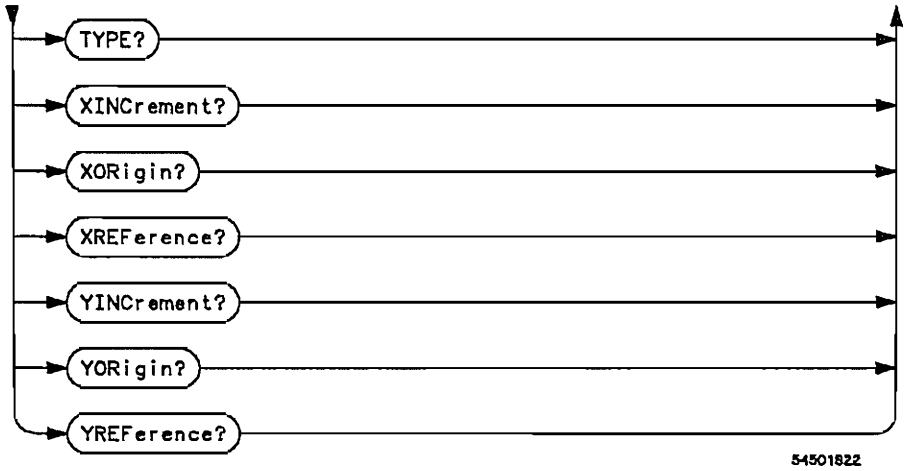


Figure 18-1. Waveform Subsystem Syntax Diagrams



channel_number = 1, 2, 3, or 4.
block_data_spec = A block of data in # format.
preamble_data = Refer to PREAMBLE command.
wmemory_number = An integer 1 through 4.

Figure 18-1. Waveform Subsystem Syntax Diagrams (continued)

COUNt?

The :WAVEFORM:COUNT query will always return a one in the Peak Power Analyzer. The query is included for compatibility with other Hewlett-Packard instruments.

The value returned has no meaning for the Peak Power Analyzer.

Query Syntax: :WAVeform:COUNt?

Returned Format:

[:WAVeform:COUNt] 1<NL>

Example:

```
DIM Cnt$[100]
OUTPUT 707;":WAVEFORM:COUNT?"
ENTER 707;Cnt$
PRINT Cnt$
END
```

DATA

The :WAVEFORM:DATA command causes the Peak Power Analyzer to accept a waveform data record over the bus and store it in the previously specified waveform memory. The waveform memory is specified with the WAVEFORM:SOURCE command. Waveform data may only be written to waveform memories.

Note

The format of the data being sent must match the format previously specified by the waveform preamble for the destination memory.

When new waveform data is downloaded to a waveform memory, the units of the data may not agree with the type of data. This could result in an incorrect units indication of the waveform memory when using the markers or making a measurement. The correct units may be set ahead of time by using the root level :STORE command to store an appropriate channel to the waveform memory before downloading the data. The preamble doesn't include the units of the waveform data in order to maintain compatibility with other Hewlett-Packard instruments.

The DATA query tells the Peak Power Analyzer to output the waveform record stored in a waveform memory or channel buffer, previously specified with the :WAVEFORM:SOURCE command.

Command Syntax: :WAVeform:DATA <binary block data in # format>

Example:

```
OUTPUT 707;":WAV:DATA"
```

Query Syntax: :WAVeform:DATA?

Returned Format:

[:WAVeform:DATA] <binary block length bytes><binary block><NL>

The following program moves data from the Peak Power Analyzer to the controller and then back to the Peak Power Analyzer with the :WAVEFORM:DATA query and command.

Example:

```
10  CLEAR 707
20  ! SET UP ACQUIRE SUBSYSTEM
30  OUTPUT 707;":ACQUIRE:TYPE NORMAL;COUNT 1;POINTS 512"
40  OUTPUT 707;":DIGITIZE CHANNEL1" ! STORE CHAN 1 DISPLAY TO WMEM1
50  OUTPUT 707;":SYSTEM:HEADER OFF;:EOI ON"
60  OUTPUT 707;":WAVEFORM:SOURCE CHANNEL1;FORMAT WORD"   !SELECT WAVEFORM DATA
70   ! SOURCE AND OUTPUT FORMAT
80  OUTPUT 707;":WAVEFORM:DATA?"
90  ENTER 707 USING "#,2A,8D";Headers$,Bytes ! READ LENGTH BYTE
100 Length=Bytes
110 Length=Length/2
120 ALLOCATE INTEGER Waveform(1:Length)
130 ENTER 707 USING "#,W";Waveform(*) ! ENTER WAVEFORM DATA TO INTEGER ARRAY
140 ENTER 707 USING "-K,B";End$ ! ENTER TERMINATOR
150 DIM Preamble$[200]
160 OUTPUT 707;":WAV:PREAMBLE?" ! OUTPUT WAVE SOURCE PREAMBLE TO CONTROLLER
170 ENTER 707 USING "-K";Preamble$ ! ENTER PREAMBLE INTO CONTROLLER
180 OUTPUT 707;":WAV:SOURCE WMEMORY4" !CHANGE SOURCE TO WMEMORY 4
190 OUTPUT 707 USING "#,K";":WAVEFORM:PREAMBLE ";Preamble$ ! SEND PREAMBLE FROM
200  ! CONTROLLER TO WMEMORY 4
210 OUTPUT 707 USING "#,K";":WAVEFORM:DATA #800001024"  !SEND HEADER
220 OUTPUT 707 USING "W";WAVEFORM(*)  ! SEND WAVEFORM DATA TO WMEMORY 4
220 OUTPUT 707;":BLANK CHANNEL1;VIEW WMEMORY4" ! TURN CHAN 1 OFF - WMEM 4 ON
230 END
```

Note

In program line 190, the space after :WAVEFORM:PREAMBLE and before the quote mark is necessary.

FORMat

The :WAVEFORM:FORMAT command sets the data transmission mode for waveform data output. This command controls how the data is formatted on the bus when sent from the Peak Power Analyzer.

When the ASCII mode is selected, the data is ASCII digits with each data value separated by a comma.

WORD formatted data transfers as 16-bit binary integers in two bytes with the most significant byte of each word sent first.

BYTE and COMPRESSED formatted data is transferred as 8-bit bytes.

The FORMAT query returns the current output format for transfer of waveform data.

Command Syntax: :WAVeform:FORMat {ASCii | WORD | BYTE | COMPressed}

Example:

```
OUTPUT 707;":WAV:FORMAT WORD"
```

Query Syntax: :WAVeform:FORMat?

Returned Format:

[:WAVeform:FORMat] <mode><NL>

Where:

<mode>::= {ASCii | WORD | BYTE | COMPressed}

Example:

```
DIM Frmt$[100]
OUTPUT 707;":WAV:FORMAT?"
ENTER 707;Frmt$
PRINT Frmt$
END
```

POINts?

The :WAVEFORM:POINTS query returns the points value in the currently selected waveform preamble. The points value is the number of time buckets contained in the waveform selected with the WAVEFORM:SOURCE command.

In most cases, the number of time buckets actually acquired will be the number of points set in the ACQUIRE subsystem. There are some sweep speeds where the actual number of points is less than requested. The maximum timebase resolution limits the number of points for certain sweep speeds. These are shown in the following table:

	Sweep Speed (Per Division)			
	2 ns	**5 ns**	**10 ns**	**>10 ns**
Points Allowed	32, 64, 128, 200	32, 64, 128, 256, 500	32, 64, 128, 256, 500, 512, 1000	32, 64, 128, 256, 500, 512, 1024

Query Syntax: :WAVeform:POINts?

Returned Format:

[:WAVeform:POINts] <value><NL>

Where:

<value> ::= number of acquired data points (integer–NR1 format)

Example:

```
Dim Pts$[100]
OUTPUT 707;":WAV:POINTS?"
ENTER 707;Pts$
PRINT Pts$
END
```

PREamble

The :WAVEFORM:PREAMBLE command sends a waveform preamble to the previously selected waveform memory in the Peak Power Analyzer. The preamble can only be written to waveform memory.

The PREAMBLE query sends a waveform preamble to the controller from the currently selected waveform source in the Peak Power Analyzer.

Command Syntax: :WAVeform:PREamble <preamble block>

Where:

<preamble block> ::= <format NR1>,<type NR1>,<points NR1>,<count NR1>,<xincrement NR3>,<xorigin NR3>,<xreference NR3>,<yincrement NR3>,<yorigin NR3>,<yreference NR3>

Query Syntax: :WAVeform:PREamble?

Returned Format:

[:WAVeform:PREamble] <preamble block><NL>

Where:

<preamble block> ::= <format NR1>,<type NR1>, <points NR1>,<count NR1>,<xincrement NR3>,<xorigin NR3>,<xreference NR1>,<yincrement NR3>,<yorigin NR3>,<yreference NR1>

Where:

<format> ::= 0 for ASCII format
1 for BYTE format
2 for WORD format
4 for COMPRESSED format

<type> ::= 1 for NORMAL type
2 for AVERAGE type
3 for ENVELOPE type

Example:

This example program uses both the command and query form of the PREAMBLE command. First, the preamble is queried (output to the controller). Then, the preamble is returned to the previously selected waveform memory.

```
10  DIM Pre$[120]
20  OUTPUT 707;"SYSTEM:HEADER OFF"
30  OUTPUT 707;":WAVEFORM:PREAMBLE?"
40  ENTER 707 USING "-K";Pre$
50  OUTPUT 707 USING "#,K";":WAV:PREAMBLE ";Pre$
60  END
```

Note

In line 50 of the program example, a space is inserted between the word "PREAMBLE " and the closed quotation mark. This space must be inside the quotation mark because in this format (#,K) the data is packed together. Failure to add the space will cause an illegal command to be sent to the Peak Power Analyzer.

Example:

The following program example brings the preamble into a BASIC numeric array.

```
10  DIM Preamble(1:10)
20  OUTPUT 707;":SYSTEM:HEADER OFF"
30  OUTPUT 707;":WAVEFORM:PREAMBLE?"
40  ENTER 707 ;Preamble(*)
50  OUTPUT 707;":WAV:PREAMBLE ";Preamble(*)
60  END
```

SOURce

The :WAVEFORM:SOURCE command selects the channel or waveform memory to be used as the source for the waveform commands. The source can only be set to waveform memory when sending data to the Peak Power Analyzer.

The SOURCE query returns the currently selected source for the waveform commands.

Note

When Option 001, Single Sensor Input, is installed, no error is reported when the waveform source is specified to be channel 4. The results from specifying channel 4 are unpredictable.

Command Syntax: :WAVeform:SOURce {CHANnel{1 | 2 | 3 | 4} | WMEMory{1 | 2 | 3 | 4}}

Example:

```
OUTPUT 707;":WAV:SOURCE WMEMORY3"
```

Query Syntax: :WAVeform:SOURce?

Returned Format:

[:WAVeform:SOURce] <source><NL>

Where:

<source> ::= {CHANnel{1 | 2 | 3 | 4} | WMEMory{1 | 2 | 3 | 4}}

Example:

```
DIM Src$[100]
OUTPUT 707;":WAVEFORM:SOURCE?"
ENTER 707;Src$
PRINT Src$
END
```

TYPE?

The :WAVEFORM:TYPE query returns the data type for the previously specified waveform source.

Query Syntax: :WAVeform:TYPE?

Returned Format:

[:WAVeform:TYPE] <mode><NL>

Where:

<mode> ::= {AVERage | ENVelope | NORMal}

Example:

```
DIM Typ$[100]
OUTPUT 707;":WAVEFORM:TYPE?"
ENTER 707;Typ$
PRINT Typ$
END
```

XINCrement?

The :WAVEFORM:XINCREMENT query returns the x-increment value currently in the preamble for the current specified source. This value is the time difference between consecutive data points for NORMAL, AVERAGE, or ENVELOPE data.

Query Syntax: :WAVeform:XINCrement?

Returned Format:

[:WAVeform:XINCrement] <value><NL>

Where:

<value> ::= x–increment in the current preamble (exponential–NR3 format)

Example:

```
DIM Xin$[100]
OUTPUT 707;":WAV:XINCREMENT?"
ENTER 707;Xin$
PRINT Xin$
END
```

XORigin?

The :WAVEFORM:XORIGIN query returns the x–origin value currently in the preamble for the current specified source. This value is the time of the first data point in memory with respect to the trigger point.

Query Syntax: :WAVeform:XORigin?

Returned Format:

[:WAVeform:XORigin] <value><NL>

Where:

<value> ::= x–origin value currently in preamble (exponential–NR3 format)

Example:

```
DIM Xr$[100]
OUTPUT 707;":WAV:XORIGIN?"
ENTER 707;Xr$
PRINT Xr$
END
```

XREFerence?

The :WAVEFORM:XREFERENCE query returns the current x-reference value in the preamble for the current specified source. This value specifies the data point associated with the x-origin data value. For the Peak Power Analyzer, this value is always zero.

Query Syntax: :WAVeform:XREFerence?

Returned Format:

[:WAVeform:XREFerence] <value><NL>

Where:

<value> ::= x-reference value in the current preamble, always 0 (integer-NR1 format)

Example:

```
DIM Xrf$[100]
OUTPUT 707;":WAV:XREFERENCE?"
ENTER 707;Xrf$
PRINT Xrf$
END
```

YINCrement?

The :WAVEFORM:YINCREMENT query returns the y-increment value currently in the preamble for the current specified source. This value is the amplitude difference between consecutive data points.

Query Syntax: :WAVeform:YINCrement?

Returned Format:

[:WAVeform:YINCrement] <value><NL>

Where:

<value> ::= y-increment value in the current preamble (exponential-NR3 format)

Example:

```
DIM Yin$[100]
OUTPUT 707;":WAV:YINCREMENT?"
ENTER 707;Yin$
PRINT Yin$
END
```

YORigin?

The :WAVEFORM:YORIGIN query returns the y–origin currently in the preamble for the current specified source. This value is the voltage at center screen. When the Peak Power Analyzer is in log mode, the value returned is the corresponding linear value at center screen.

Query Syntax: :WAVeform:YORigin?

Returned Format:

[:WAVeform:YORigin] <value><NL>

Where:

<value> ::= y–origin in the current preamble (exponential–NR3 format)

Example:

```
Dim Yr$[100]
OUTPUT 707;":WAV:YORIGIN?"
ENTER 707;Yr$
PRINT Yr$
END
```

YREFerence?

The :WAVEFORM:YREFERENCE query returns the current y–reference value in the preamble for the current specified source. This value specifies the data point where the y–origin occurs.

Query Syntax: :WAVeform:YREFerence?

Returned Format:

[:WAVeform:YREFerence] <value><NL>

Where:

<value> ::= y–reference value in the current preamble (integer–NR1 format)

Example:

```
DIM Yrf$[100]
OUTPUT 707;"WAV:YREFERENCE?"
ENTER 707;Yrf$
PRINT Yrf$
END
```

A

Algorithms

Introduction

One of the Peak Power Analyzer's primary features is its ability to make automatic measurements on displayed waveforms. This appendix provides details on how automatic measurements are calculated and offers some tips on how to improve results.

Measurement Setup

Measurements typically should be made at the fastest possible sweep speed for the most accurate measurement results. The entire portion of the waveform that is to be measured must be displayed on the Analyzer. That is:

- at least one complete cycle must be displayed for PRI (Pulse Repetition Interval) or PRF (Pulse Repetition Frequency) measurements
- the rising edge followed by the negative edge must be displayed for pulse width measurements
- the falling edge followed by the rising edge must be displayed for offtime measurements
- the leading edge of the waveform and enough of the pulse to define the top and base must be displayed for risetime measurements and all other edge measurements
- the trailing edge of the waveform must be displayed for falltime measurements and all other edge measurements

- the top, base, and peak of the waveform must be displayed for overshoot measurements.

Making Measurements

If more than one waveform, edge, or pulse is displayed, the measurements are made on the first (leftmost) portion of the displayed waveform that can be used. If there are not enough data points, the Analyzer will display $\leq$ with the measurement results. This is to remind you that the results may not be as accurate as possible. It is recommended that you re-scale the displayed waveform and make your measurement again.

When any of the standard measurements are requested, the Peak Power Analyzer first determines the top-base linear amplitude levels at 100%-0%. From this information, it can determine the other important linear amplitude values (10%, 90%, and 50%) needed to make the measurements. The 10% and 90% linear amplitude values are used in the risetime and falltime measurements as well as in all other edge measurements. The 10% and 90% values are also used to determine the 50% value. The 50% linear amplitude value is used for measuring PRF, PRI, pulse width, and duty cycle.

Automatic Top-Base

Top-Base is the heart of most automatic measurements. It is used to determine V_{top} and V_{base}, the 0% and 100% linear amplitude levels at the top and the bottom of the waveform. From this information the Analyzer can determine the 10%, 50%, and 90% points, which are also used in most measurements. The top or base of the waveform is not necessarily the maximum or minimum linear amplitude present on the waveform. Consider a

pulse that has slight overshoot. It would be wrong to select the highest point of the waveform as the top since the waveform normally rests below the perturbation.

Top-Base performs a histogram on the waveform and finds the most prevalent point above and below the waveform midpoint. The most prevalent point is one that represents greater than approximately 5% of the total display points (501) and is considered to be either the top or base. If no point accounts for more than 5% of the total, then the top is chosen as the absolute maximum and the base is chosen as the absolute minimum.

If the top or base arrived at in the measurement is different from what is expected, the measurement results may also be different than expected. If there is doubt, turn continuous measurements off in the Define Measure Menu and repeat the measurement to observe where the markers are placed on the waveform.

Edge Definition

Both rising and falling edges are defined as transitional edges that must cross three thresholds.

A rising edge must cross the lower threshold in a positive direction (defining it as a rising edge), cross the mid threshold (any number of crossings, both positive and negative are permissible) and then cross the upper threshold without any crossing of the lower threshold.

A falling edge must cross the upper threshold in a negative direction, cross the mid threshold (any number of times), and then cross the lower threshold without crossing the upper threshold.

Note Most time measurements are made based on the position of the first crossing of the middle threshold.

Algorithm Definitions

Following are the definitions that all measurements are based on:

delay

There are three types of delay measurement:

- jitter
- standard
- user-defined

Jitter occurs only under the following circumstances:

- standard/user-defined key is set to standard
- two delay parameters are the same
- display mode is envelope

if

first edge on minimum waveform is rising

then

delay = mid-threshold of first rising edge of max waveform minus mid-threshold of first rising edge on min waveform

else

delay = mid-threshold of first falling edge on min waveform minus mid-threshold of first falling edge on max waveform

The standard delay measurement occurs when in the standard mode (not user-defined) and is not a jitter measurement.

standard delay = mid-threshold of the first edge of second parameter minus mid-threshold of the first edge of the first parameter

Note 

Negative delay is possible

User defined delay = second channel edge minus first channel edge

Pulse Width

The pulse width algorithm has standard and user-defined considerations.

if

first edge is rising

then

Pulse width= mid-threshold crossing of first falling edge minus mid-threshold crossing of first rising edge

else

Pulse width= mid-threshold crossing of second falling edge minus mid-threshold crossing of first rising edge

User-defined is the same as Standard definition except user-defined threshold.

Offtime

The offtime algorithm has standard and user-defined considerations:

if

first edge is rising

then

offtime= second rising edge–first falling edge

else

offtime= first rising edge–first falling edge

PRI

if

first edge is rising

then

PRI = second rising edge–first rising edge

else

PRI = second falling edge–first falling edge

PRF

PRF= 1/PRI

Duty Cycle

duty cycle = (pulse width/PRI)(100)

Note

Pulse width is always calculated using mid-threshold.

Risetime

risetime = time at upper threshold–time at lower threshold

Falltime

falltime = time at lower threshold–time at upper threshold

Overshoot

$$Overshoot = \frac{(peak - top)}{(top - base)} * 100$$

Note

The algorithm will tend to fail for overshoot values exceeding 50%.

V_{max} V_{max} = amplitude of the maximum point on screen

V_{min} V_{min} = amplitude of the minimum point on screen

$V_{p\text{-}p}$ $V_{p\text{-}p} = V_{max} - V_{min}$ (For voltage only.)

V_{top} V_{top} = most prevalent point above waveform midpoint

V_{base} V_{base} = most prevalent point below waveform midpoint

V_{amp} $V_{amp} = V_{top} - V_{base}$ (For voltage only.)

V_{avg} Average amplitude of the first cycle of the displayed signal is measured. If a complete cycle is not present, the Analyzer will average all data points.

V_{rms} The rms voltage of the first cycle of the displayed signal is measured. If a complete cycle is not present, the measurement will compute rms on all data points.

$$Vrms(ac) = \{1/n \sum_{j=1}^{n} V_j^2 - 1/n \sum_{j=1}^{n} V_j\}^{1/2}$$

B

Programming Examples

Note

The following examples were written in HP BASIC 5.0 for an HP 9000 Series 200/300 Controller.

Spec Line Program

```
10  !
20  !
30  ! This program demonstrates how spec lines can be created and used
40  ! in the HP 8990A.  The overall steps of this program are as follows:
50  !
60  !   o  When RUN, the program causes data to be acquired on channel
70  !      'Chan' (as specified in the 'Initialize' subroutine below).
80  !
90  !   o  The acquired waveform is read into memory.  The program
100 !      computes some minimum and maximum spec limits based on the
110 !      values that were acquired.  Both min & max speclines  are
120 !      written to the HP 8990A waveform memories.  (Note: it takes a
130 !      few minutes to compute the spec limits!)
140 !
150 !    o The waveform math functions are set up to monitor differences
160 !      between the spec lines and any newly acquired data from the
170 !      previously specified channel.
180 !
190 !    o Finally, the measure compare function of the HP 8990A is enabled
200 !      to flag any deviations of level beyond the 'spec lines', with
210 !      those violations being written to the HP 8990A pixel memory.
220 !
230 !    o The program is stopped.  The HP 8990A is set to run mode, so
240 !      it can repeatedly monitor the named channel for changes.
250 !
260 !
270 ! HP 8990A RESOURCE USAGE:
280 !   Waveform Memory 1:  Holds the MAX spec limit, derived from 1st acq.
290 !   Waveform Memory 2:  Holds the MIN spec limit, derived from 1st acq.
```

```
300  !          Function 1:  Acquired Data  -   Max Spec (WMEM1)
310  !          Function 2:  Min Spec (WMEM2)   -   Acquired Data
320  !      Pixel Memory 1:  Holds any violations of the spec line.
330  !
340  !
350    COM Preamble$[200],Preamb(1:10)      ! Two versions of waveform preamble
360    COM INTEGER Waveform(1:1024)         ! The acquired waveform,unscaled.
370    COM INTEGER Wavemax(1:1024)          ! Max spec line of acquired data
380    COM INTEGER Wavemin(1:1024)          ! Min spec line of acquired data
390    !
400    INTEGER Chan                         ! Specifies which chan to acq data
410    DIM Chan_id$[5]                      ! String vars of data acq is channel
420    INTEGER Vdelta                       ! y-axis ellipse width about each pt
430    INTEGER Tdelta                       ! x-axis ellipse width about each pt
440    INTEGER Tcenter                      ! x-axis ellipse center of each pt.
450    INTEGER Telipse                      ! x-axis value during ellipse calc.
460    !
470    GOSUB Initialize                     ! Initialize the I/O paths and HP 8990A
480    !
490    GOSUB Acq_modes                      ! Set up the acquisition modes.
500    GOSUB Get_waveform                   ! Acquire waveform, read to memory
510    !
520    GOSUB Make_spec_lines                ! Make min/max spec line envelopes.
530    GOSUB Minspec2wmem2                  ! Write Min spec line to HP 8990A WMEM2
540    GOSUB Maxspec2wmem1                  ! Write Max spec line to HP 8990A WMEM1
550    !
560    GOSUB Compare_2_specs                ! Set waveform math to check w/specs
570    GOSUB Setup_compare                  ! Set up continuous spec meas compares
580    DISP USING "K";"Done.  The HP 8990A is now monitoring channel ",Chan," for changes."
590    PRINT
600    PRINT
610    PRINT "    -->NOTE: You can turn off the Display Connect Dots for more throughput"
620    !
630    PAUSE                                ! Pause so vars can be looked at
640                                         ! manually, if desired.
650    STOP                                 ! Stop pgm, but instr keeps running
660  !
670  !
680 Initialize:                             ! - - - - - - - - - - - - - - - -
690   ! Set up the I/O Paths and demo variables and reset the instrument.
700    OUTPUT KBD;CHR$(255)&"K";            ! Clear screen of previous text
710    PRINT TABXY(25,5);"S P E C   L I N E   D E M O";TABXY(1,7)
720    Tab_in=5                             ! Specify indent for status prints
730    DISP "Initializing & autoscaling the HP 8990A.."
740    Volt_tol=5                           ! Volt limits = +/- 5% of acq value
750    Time_tol=5                           ! Time limits = +/- 5% of acq value
760    Chan=2                               ! This is the channel to get data
770    OUTPUT Chan_id$ USING "#,K,D";"CHAN",Chan ! String vars of chan to acq.
780    ASSIGN @Ppa TO 707                   ! Use default address.
```

```
790   ASSIGN @Fast TO 707;FORMAT OFF        ! Set up a fast path w/out formats
800   CLEAR @Ppa                            ! Send an HP-IB Device Clear
810   OUTPUT @Ppa;"*RST"                    ! Preset hw settings of HP 8990A
820   OUTPUT @Ppa;":AUT"                    ! Autoscale to acquire signals.
830  !
840   OUTPUT @Ppa;"SYSTem:HEADer OFF"       ! Return only numbers with any query
850   OUTPUT @Ppa;":DISPlay:CONNect ON;GRATicule FRAMe  ;FORMat 1"
860   OUTPUT @Ppa;":DISPlay:CONNect ON"     ! Turned ON the connect dots mode.
870   PRINT TAB(Tab_in);"o Initializing and Autoscaling complete."
880   RETURN
890  !
900 Acq_modes:                             ! - - - - - - - - - - - - - - - -
910  DISP "Setting HP 8990A acquisition modes.."
920  ! This section sets up the acquisition modes and display.
930   OUTPUT @Ppa;":ACQuire:COMPlete 100;TYPE AVERage;COUNt 4;POINts 500"
940  !
950   PRINT TAB(Tab_in);"o HP 8990A Acquisition mode is set."
960   RETURN
970  !
980  !
990 Get_waveform:                          ! - - - - - - - - - - - - - - - -
1000  ! Acquire and then read the waveform into the computer memory arrays.
1010  ! The waveform is acquired and read from the Chan_id$ channel.  The
1020  ! min and max limits arrays are also allocated here.
1030  !
1040   DISP "Acquiring reference waveform.."
1050   OUTPUT @Ppa;"WAVeform:FORMat WORD"    ! Specify waveform format as word.
1060   OUTPUT @Ppa;"digitize "&Chan_id$      ! Capture data to ch2 and stop
1070   OUTPUT @Ppa;"CHANnel2:RANGe?"         ! Read the range for later use.
1080   ENTER @Ppa;V_range_value
1090  !
1100   OUTPUT @Ppa;":WAVeform:SOURce "&Chan_id$ ! Waveform system -> ChanN
1110  !
1120   OUTPUT @Ppa;"waveform:preamble?"      ! Read the real scale factors
1130   ENTER @Ppa;Preamb(*)                  ! Read x/y scale factors; preamble
1140  !
1150   OUTPUT @Ppa;"waveform:preamble?"      ! Read an equivalent ASCII string
1160   ENTER @Ppa USING "-K";Preamble$       ! for later outputting again.
1170  !
1180   OUTPUT @Ppa;"waveform:data?"          ! Read unscaled waveform data
1190  ENTER @Ppa USING "#,2A,8D";Header$,Bytes  ! Read header & byte count
1200  Length=Bytes/2                        ! Word format, 2 bytes/data elem.
1210  REDIM Waveform(1:Length),Wavemax(1:Length),Wavemin(1:Length)
1220  ENTER @Fast;Waveform(*)                ! Read copy of signal waveform in
1230  ENTER @Ppa USING "-K,B";End$           ! Swallow a linefeed
1240  PRINT TAB(Tab_in);"o Reference waveform acquired from channel";Chan
1250  RETURN
1260  !
1270  !
```

```
1280 Make_spec_lines:                        ! - - - - - - - - - - - - - - - - -
1290  ! Fill arrays Wavemin(*) and Wavemax(*) with envelopes that lie some
1300  ! percentage above & below the data in the prev acquired Waveform(*).
1310  ! Length is assumed to be the number of elements in the Waveform(*)
1320  ! array, which is the same as the number of time buckets.
1330  !
1340  ! In this code, every 'Tdelta' index values are evaluated;  at each
1350  ! of these evaluation points an elliptical envelope is formed with
1360  ! the largest +/- values at the Tcenter of the ellipse and with lower
1370  ! values as Telipse is varied +/- 2Tdelta's about Tcenter.  Each
1380  ! of the x,y pairs in the min & max of the ellipse are then compared
1390  ! to the corresponding x-y in the min,max arrays.  The larger of
1400  ! the two is saved for the max spec line, and the min of the two is saved
1410  ! for the min spec line.
1420  !
1430  DISP "Calculating spec limits envelope (takes a few minutes).."
1440  MAT Wavemin= Waveform                  ! Init MIN speclimit to acq data
1450  MAT Wavemax= Waveform                  ! Init MAX speclimit to acq data
1460  Tdelta=Length*Time_tol/100             ! = %Time delta in ary indices
1470  Vdelta=32767*Volt_tol/100              ! = %Vert delta in raw value.
1480  FOR Tcenter=1 TO Length                ! = Time Bucket Center of ellipse
1490    FOR Telipse=(Tcenter-Tdelta) TO (Tcenter+Tdelta)  ! Each pt on ellipse
1500      IF Telipse>0 AND Telipse<=Length THEN           ! Xindex is in range
1510        Volt_pt=Vdelta*SQR(1-((Telipse-Tcenter)/Tdelta)^2)
1520        V_wv=Waveform(Tcenter)           ! Establish ellipse y-center value
1530        Volt_max=V_wv+Volt_pt            ! Sets top part of the ellipse
1540        Volt_min=V_wv-Volt_pt            ! Sets bottom part of the ellipse
1550        IF Wavemax(Telipse)<Volt_max THEN  ! Then this ellipse is bigger
1560          Wavemax(Telipse)=Volt_max        ! than prev, so it's new limit
1570        END IF
1580        IF Wavemin(Telipse)>Volt_min THEN  ! Then prev min lower than this
1590          Wavemin(Telipse)=Volt_min        ! elip y-val, so use it now.
1600        END IF
1610      END IF
1620    NEXT Telipse
1630  NEXT Tcenter
1640  PRINT TAB(Tab_in);"o Spec limits envelope has been calculated."
1650  RETURN
1660  !
1670  !
1680 Minspec2wmem2:                          ! D   oad min spec line to WMEM2
1690   ! The waveform spec line envelope of min   values in Wavemin(*) are
1700   ! copied, along with the current preamble   to the HP 8990A waveform
1710   ! memory 2.
1720   DISP "Copying minimum spec line to HP 8990A waveform memory 2.."
1730   OUTPUT @Ppa;":WAVeform:SOURce WMEMory2"                ! Destination
1740   OUTPUT @Ppa USING "#,K";":WAVeform:PREamble ";Preamble$ ! Scaling info
1750   OUTPUT @Ppa USING "#,K,ZZZZ";":WAVeform:DATA #80000";Bytes ! Header
1760   OUTPUT @Fast;Wavemin(*)                                ! The data.
```

```
1770   OUTPUT @Ppa;""                                  ! End-Of-Line Terminator
1780   OUTPUT @Ppa;":view wmemory2"              ! Now put it on display
1790   PRINT TAB(Tab_in);"o The Minimum spec line has been downloaded to the HP 8990A WMEM2."
1800   RETURN
1810  !
1820  !
1830 Maxspec2wmem1:                            ! Download MAX spec line to WMEM1
1840   ! The waveform spec line envelope of MAXimum values in Wavemax(*) are
1850   ! copied, along with the current preamble, into the HP 8990A waveform
1860   ! memory 1.
1870   DISP "Copying Maximum spec line to HP 8990A waveform memory 1.."
1880   OUTPUT @Ppa;":waveform:SOURce wmemory1"                 ! Destination
1890   OUTPUT @Ppa USING "#,K";":waveform:preamble ";Preamble$ ! Scaling info
1900   OUTPUT @Ppa USING "#,K,ZZZZ";":waveform:data #80000";Bytes  ! Header
1910   OUTPUT @Fast;Wavemax(*)                                 ! The data
1920   OUTPUT @Ppa;""                                  ! End-Of-Line Terminator
1930   OUTPUT @Ppa;":view wmemory1"            ! Now put it on display
1940   PRINT TAB(Tab_in);"o The Maximum spec line has been downloaded to the HP 8990A WMEM1."
1950   RETURN
1960  !
1970  !
1980 Compare_2_specs:                          ! - - - - - - - - - - - - - - - - - -
1990   ! Set up the waveform math to compare the minimum and maximum against
2000   ! each newly acquired waveform.  Function 1 checks if the acquired
2010   ! signal exceeds the max spec; Acquired - Max >0.  Function 2 is set
2020   ! to monitor when the acquired data falls below the minimum spec line,
2030   ! as in Min - Acquired > 0.  The function's range is set to the same
2040   ! range in which the signal was first acquired.
2050   !
2060   DISP "Setting up HP 8990A Waveform Math to compare data with spec lines.."
2070   Funct_range=V_range_value*2           ! Set vertical range = acquired rng
2080   OUTPUT @Ppa;":FUNCtion1:SUBTract "&Chan_id$&", WMEMory1" ! ChanAcq - Max
2090   OUTPUT @Ppa;":FUNCtion1:RANGe ";Funct_range             ! Set the scale
2100   !
2110   OUTPUT @Ppa;":FUNCtion2:SUBTract WMEMory2, "&Chan_id$   ! Min - ChanAcq
2120   OUTPUT @Ppa;":FUNCtion2:RANGe ";Funct_range             ! Set scale too
2130   !
2140   OUTPUT @Ppa;":VIEW FUNCtion1;:VIEW FUNCtion2"        ! Display both functions
2150   PRINT TAB(Tab_in);"o The HP 8990A Waveform math has been set up to monitor the data."
2160   RETURN
2170   !
2180   !
2190 Setup_compare:                            ! - - - - - - - - - - - - - - - - - -
2200   ! Set up the measure compare test to FAIL when either of the two waveform
2210   ! math functions go above zero, which would mean the signal has either
2220   ! gotten bigger or smaller than the signal that was first acquired.
2230   !
2240   DISP "Setting up the HP 8990A measurement compare mode.."
2250   OUTPUT @Ppa;"MEASure:STATistics ON"         ! Enable continuous meas
```

```
2260   OUTPUT @Ppa;"MEASure:DESTination PMEM1"     ! Store out-of-specs->PMEM1.
2270   !
2280   OUTPUT @Ppa;":MEASure:SOURce FUNCtion1"     ! Set up to monitor minimum
2290   OUTPUT @Ppa;":MEASure:VMAX"                 !  values of Max - Acquired
2300   !
2310   OUTPUT @Ppa;":MEASure:SOURce FUNCtion2"     ! Set to monitor the minimum
2320   OUTPUT @Ppa;":MEASure:VMAX"                 ! values of Acquired - Min
2330   !
2340   OUTPUT @Ppa;":MEASure:COMPare VMAX, 0,-250E3" ! Error if Vmin is > zero.
2350   OUTPUT @Ppa;":MEASure:LIMIttest measure"    ! Enable the meas compares
2360   OUTPUT @Ppa;":MEASure:POSTfailure CONT"     ! Keep running on failure
2370   OUTPUT @Ppa;":RUN"                          ! Continuously run and display
2380   PRINT TAB(Tab_in);"o The measurement compare has begun on the HP 8990A."
2390   RETURN
2400   !
2410   !
2420 END
```

Pulse Droop Measurement Program

```
10   !
20   !
30   ! This program demonstrates how to measure the slope droop of a pulse.
40   ! A demo pulse is either downloaded from BASIC, or a real pulse is
50   ! measured on the HP 8990A.  Either way, the pulse waveform is saved in
60   ! the HP 8990's waveform memory 1, where the measurements are done.
70   ! Intermediate results are shown on the HP 8990A display via the other
80   ! waveform memories.
90   !
100   !
110   ! HP 8990A Waveform memory usage:
120   !   WMEM1 = The waveform whose droop is to be measured.  This is
130   !           obtained from an acquisition or from the Build_demo()
140   !           subprogram in BASIC.
150   !   WMEM2 = The linear regression curve fit of the droopy top of
160   !           the pulse.
170   !   WMEM3 = The rising and falling edge line fits.
180   !
190   !
200   ! List of Subprograms:
210   ! ==============  ====  ========================================
220   !      NAME        SIZE                 DESCRIPTION
230   ! ==============  ====  ========================================
240   ! MAIN             5980  This is the main program.
250   ! Get_a_waveform   1916  Acquires or synthesizes a pulse for measuring
260   ! Measure_droop    4332  Measure droop of waveform in the HP 8990A WMEM1
270   ! Top_of_display    388  Simple routine to clear display, put title up
280   ! Instrument_init  1768  Does standard instr reset sequence, and setup.
290   ! Waveform2wmem1   2988  Copy the waveform in computer to HP 8990A's WMEM1.
300   ! FNFind_edges     3708  Finds times where rise & fall edge = 50% ampl.
310   ! Measure_slope    2260  Does the linear regression on top of the pulse
320   ! Line2wmem2       2742  Copies the top-of-pulse line fit to HP 8990A WMEM2
330   ! Measure_sides    3392  Pulse 10 & 50% ampl at rising,falling edge->line
340   ! Sides2wmem3      4078  Saves rise,fall edge line approx to HP 8990A WMEM3
350   ! Vertmarkers       732  Place vert markers to two pts used in Droop calc
360   ! Build_demo       4226  In computer, synthesize pulse with known droop
370   ! Acquire_data     4064  Acquire pulse w/good timebase, read ->computer
380   ! Linear_regress   2454  Utility pgm for linear curve fit to (x,y) data
390   ! FNY_intersect     836  Calculate 1 y-value where two lines intersect.
400   ! FNVertical        690  Get watts or volts, given waveform data value.
410   ! FNTime            942  Get seconds, given index in   waveform data value.
420   ! FNVert_rawval     724  Get waveform data unscaled value from vertical
430   ! FNTime_index      594  Get position in wave:data ary given time, sec.
440   ! ==============  ====  ========================================
450   !
460   COM /Meas_data/ REAL Preamble(1:10),INTEGER Raw_data(1:1024)
470   !
```

```
480   REAL Chan                    ! HP 8990A channel to acquire data on.
490   REAL Echo_results            ! Enable intermediate results display
500   REAL Tstart,Tstop            ! DeltaTimes from edge for regression
510   !
520   ! *** Initialize the address, and the channel to acquire the pulse on
530   ! *** (if the pulse is acquired).  Also sets the echo-of-results flag,
540   ! *** and presets the HP 8990A to the desired state.
550   GOSUB Mainvar_init               ! Initialize main variables.
560   CALL Instrument_init(@Ppa)       ! Initialize the HP 8990A for I/O, etc
570   !
580   ! *** Acquire a waveform to be characterized, either by measuring it or
590   ! *** by making one up in BASIC.  Then choose over what limits the
600   ! *** the top of the pulse should be fit to a straight line.
610   CALL Get_a_waveform(@Ppa,Chan,Echo_results)  ! Pulse --> WaveMemory 1
620   GOSUB Pick_measlimits            ! Select Tstart & Tstop times for meas
630   !
640   ! *** Next, Measure the droop.  See the documentation in 'Measure_droop'
650   ! *** for details on how this is done.
660   CALL Measure_droop(@Ppa,Droop,Tstart,Tstop,Echo_results)  ! Meas pulse droop
670   Droop=DROUND(Droop,3)                ! Round result to 3 sig digits.
680   DISP "Droop is";Droop;"percent.  Demo Completed."
690   !
700   LOCAL @Ppa                           ! Put the HP 8990A back to local ctl.
710   ASSIGN @Ppa TO *                     ! Close out the 8990A I/O path.
720   PAUSE                                ! Pause, allows look at variables
730   STOP                                 ! End of execution, demo is finished.
740   !
750   !
760   !
770 Mainvar_init:                          ! - - - - - - - - - - - - - - - - - -
780   ! Initialize all vars used in the main part of the program, such as the
790   ! Instrument address, etc.
800   ASSIGN @Ppa TO 707                   ! Assume factory default address.
810   Chan=2                               ! Which channel to acquire data from
820   Echo_results=1                       ! Set 0 to see intermediate results.
830   RETURN
840   !
850 Pick_measlimits:                       ! - - - - - - - - - - - - - - - - - -
860   ! Pick the start and stop times within the pulse where the top of the
870   ! pulse is to be fitted to a linear line and droop determined.
880   ! The vars 'Tstart' and 'Tstop' are set as follows:
890   !   'Tstart' set to delta time after rising edge when the linear fitting
900   !            of the top of the pulse should begin.
910   !   'Tstop' set to delta time before falling edge when the linear fitting
920   !            of the top of the pulse should stop.
930   ! Both times are returned in seconds.
940   !
950   CALL Top_of_display                  ! Repaint display w/title, for text.
960   Tab_in=10                            ! Specifies left margin for prints.
```

```
970   PRINT TAB(Tab_in);"DROOP MEASUREMENT LIMITS..."
980   PRINT
990   PRINT TAB(Tab_in);"  Step 1: Enter the amount of time, in microseconds, from the "
1000  PRINT TAB(Tab_in);"          50% point of the pulse rising edge.  The default is "
1010  PRINT TAB(Tab_in);"          40 microseconds.";
1020   !
1030  Tstart=40                              ! Default start time defined @40us.
1040  INPUT "Enter measurement start time relative to rising edge, in usec (40us def) ",Tstart
1050  PRINT "   --->> Using ";Tstart;"microseconds."
1060  PRINT
1070  Tstart=Tstart*1.E-6                    ! Convert from useconds to seconds.
1080   !
1090  PRINT TAB(Tab_in);"  Step 2: Enter the amount of time, in microseconds, from the "
1100  PRINT TAB(Tab_in);"          50% point of the pulse falling edge.  The default is "
1110  PRINT TAB(Tab_in);"          100 microseconds.";
1120  !
1130  Tstop=100                             ! Default STOP  time defined @40us.
1140  INPUT "Enter measurement stop time relative to falling edge, in usec (100us def) ",Tstop
1150  PRINT "   --->> Using ";Tstop;"microseconds."
1160  Tstop=Tstop*1.E-6                     ! Convert from microseconds to sec.
1170  RETURN
1180  !
1190  END
1200  !
1210  !
1220 Get_a_waveform:SUB Get_a_waveform(@Ppa,Chan,Echo_results)  ! = = = = = =
1230   ! This function obtains a pulse to be characterized.  The signal is
1240   ! either created by a BASIC subpgm, or it is acquired on the HP 8990A on
1250   ! channel 'Chan' and then copied to Waveform Memory 1 on the HP 8990A.
1260   !
1270    DIM Choice$[5]                       ! User selections entered to here.
1280    !
1290    CALL Top_of_display                  ! Initialize display for question.
1300    GOSUB Pick_demo_type                 ! Choice$= acquire sig or do demo.
1310    IF Choice$="D" THEN CALL Build_demo ! Build waveform into /Meas_data/.
1320    IF Choice$="A" THEN                  ! Then Acquire a new waveform.
1330      CALL Acquire_data(@Ppa,Chan)       ! Acquire data into /Meas_data/.
1340    END IF                               ! End of "if acquisition" 'IF'.
1350    !
1360    CALL Waveform2wmem1(@Ppa)            ! Copy the waveform to WFORM MEM 1.
1370    SUBEXIT
1380     !
1390 Pick_demo_type:                         ! - - - - - - - - - - - - - - - - -
1400   ! Ask the user to specify the type of demo they would like to do.  If
1410   ! there is no signal around, the D demo choice will cause a pulse waveform
1420   ! to be synthesized and downloaded to waveform memory 1.
1430    REPEAT
1440      PRINT TAB(20);"Which type of demo would you like?"
1450      PRINT
```

```
1460      PRINT TAB(20);"  A   ...To measure an acquired signal."
1470      PRINT
1480      PRINT TAB(20);"  D   ...To measure a Demo Signal."
1490      INPUT "Enter the letter of your choice: ",Choice$
1500      Choice$=UPC$(Choice$[1;1])          ! Grab uppercase of 1st character.
1510    UNTIL Choice$="A" OR Choice$="D"
1520    CALL Top_of_display                   ! Clear display of answered ? text
1530    RETURN
1540  SUBEND
1550    !
1560    !
1570 Measure_droop:SUB Measure_droop(@Ppa,Droop,Tstart,Tstop,Echo_results)
1580    ! Measure the droop of the pulse which is in waveform memory one.
1590    !
1600    ! MEASUREMENT ALGORITHM:
1610    !
1620    !   o Determine the linear 50% point of the waveform as the average
1630    !     of the max & min value of the pulse.
1640    !
1650    !   o Fit those points on the top of the pulse, which lie between the
1660    !     following two time intervals, to a line, determining slope and
1670    !     y-intercept:
1680    !         --> 'Tstart' seconds after rising edge crosses 50%
1690    !         --> 'Tstop' seconds before falling edge crosses 50%
1700    !
1710    !   o Compute slope & y-axis intercept of the rising edge to a
1720    !     straight line as well, using the 10% and 50% points on the
1730    !     rising edge as the two points which determine the line.
1740    !
1750    !   o Similarly, compute slope & y-axis intercept for falling edge.
1760    !
1770    !   o Compute Droop as the % difference of:
1780    !         (1) The rising edge line intersection with the top line,
1790    !         (2) The top line's intersection with the falling edge line.
1800    !
1810    ! INTERMEDIATE RESULTS:
1820    !   If the 'Echo_results' flag is non-zero, the linear regressed line
1830    !   and the rising & falling edge lines are all echoed to the HP 8990A
1840    !   display.
1850    !
1860    ! PARAMETER LIST:
1870    !   @Ppa = I/O path specifier to the HP 8990A, already initialized.
1880    !  Droop = Returned percent droop of the pulse, in percent.
1890    ! Tstart = How much time (seconds) after rising edge to start the linear
1900    !          regression of the top of the pulse for the droop meas.
1910    !  Tstop = How much time (seconds) before falling edge to stop linear
1920    !          regression
1930    ! Echo_results = Flag telling if intermediate results are echoed to the
1940    !                HP 8990A display.
```

```
1950   !
1960    REAL Tr,Tf                          ! Pulse's Risetime and Falltime
1970    REAL Slope,Intercept                ! Of top of pulse btwn limits.
1980    REAL Slope_r,Intercept_r            ! Of Risetime w/10% &50% points
1990    REAL Slope_f,Intercept_f            ! Of Falltime w/10% &50% points
2000    REAL Vert_ref                       ! Y value where topline=Riseline
2010    REAL Vert_droop                     ! Y value where topline=Fallline
2020    !
2030    IF FNFind_edges(@Ppa,Tr,Tf,"WMEM1")=-MAXREAL THEN ! Find the pulse edges.
2040      GOTO Demo_all_done                ! If error was encountered, then
2050    END IF                              ! stop the demo here and now.
2060    !
2070    ! *** Do the linear regression of the selected top part of the pulse.
2080    ! *** The slope & intercept are returned in Slope & Intercept.
2090    CALL Measure_slope(@Ppa,Slope,Intercept,Tr+Tstart,Tf-Tstop,Echo_results)
2100    IF Echo_results THEN CALL Line2wmem2(@Ppa,Slope,Intercept,Tr+Tstart,Tf-Tstop,Echo_results)
2110    !
2120    ! *** Line-fit the rising and falling edge to lines.  These lines
2130    ! *** are used to approximate the overall starting and ending of
2140    ! *** the pulse.
2150    CALL Measure_sides(@Ppa,Slope_r,Intercept_r,Slope_f,Intercept_f)
2160    IF Echo_results THEN CALL Sides2wmem3(@Ppa,Slope_r,Intercept_r,Slope_f,Intercept_f)
2170    !
2180    ! *** Compute the intersection of the sloping top of the pulse with
2190    ! *** the lines fit to the rising and falling edge.  These two
2200    ! *** intersections are used to show the starting and drooped ending
2210    ! *** of the top of the pulse.
2220    Vert_ref=FNY_intersect(Slope_r,Intercept_r,Slope,Intercept)
2230    Vert_drooped=FNY_intersect(Slope_f,Intercept_f,Slope,Intercept)
2240    OUTPUT @Ppa;"MEAS:SOUR WMEM1; :MEAS:VBASE?" ! Measure base of pulse
2250    ENTER @Ppa;Vert_base                        !for droop calc.
2260    IF Echo_results THEN CALL Vertmarkers(@Ppa,Vert_ref,Vert_drooped)
2270    !
2280    ! *** Finally, compute the % droop.
2290    Droop=100*(Vert_ref-Vert_drooped)/(Vert_ref-FNVert_rawval(Vert_base))
2300 Demo_all_done:!
2310  SUBEND
2320   !
2330   !
2340 Top_of_display:SUB Top_of_display       ! = = = = = = = = = = = = = = = =
2350    OUTPUT KBD;CHR$(255)&"K";           ! Clear display, (CLEAR SCREEN)
2360    PRINT TABXY(23,2),"P U L S E   D R O O P   D E M O"
2370    PRINT TABXY(1,8)
2380  SUBEND
2390   !
2400   !
2410 Instrument_init:SUB Instrument_init(@Ppa)  ! = = = = = = = = = = = = = = =
2420  ! Initialize the HP-IB and the HP 8990A.  This should always be done at the
2430  ! beginning of any HP-IB program.  The first command sends an INTERFACE
```

```
2440  ! clear to remove any pending data from the bus.  The second command
2450  ! sends a selective device clear to the HP 8990A, which resets the input
2460  ! and output buffers, the HP-IB grammar parser inside the HP 8990A, and
2470  !clears any pending commands.  Finally, the HP 8990A is preset to a know
2480  ! device state by the sending of a reset command.  This is a standard
2490  ! reset sequence which should be used for most HP-IB instruments.
2500    !
2510    ! *** First, the standard reset sequence.
2520    ABORT SC(@Ppa)                      ! Interface Clear, clear byte HShake
2530    CLEAR @Ppa                          ! Selective Device Clear to HP 8990A.
2540    OUTPUT @Ppa;"*CLS"                  ! Clear the status bits in the HP 8990
2550    OUTPUT @Ppa;"*RST"                  ! Reset HP 8990A to know configuration.
2560    !
2570    ! *** After the above 'standard reset sequence,' the following sets up
2580    ! *** the particular features which are relied on for this demo program.
2590    OUTPUT @Ppa;"BLANK WMEM1; BLANK WMEM2" ! Turn off both waveform memories
2600    OUTPUT @Ppa;"BLANK WMEM3; BLANK WMEM4" ! Turn off the other wmem's too.
2610    OUTPUT @Ppa;"WAV:FORM WORD"         ! Set waveform data mode to WORDS.
2620    OUTPUT @Ppa;"DISP:FORM 1"           ! Set for one screen display format.
2630    OUTPUT @Ppa;"DISP:CONN 1"           ! Turn the connect-the-dots mode ON.
2640    OUTPUT @Ppa;"SYST:POW:UNIT WATT"    ! All pwr meas done in linear watts
2650  SUBEND
2660  !
2670  !
2680 Waveform2wmem1:SUB Waveform2wmem1(@Ppa)! = = = = = = = = = = = = = = = =
2690    ! Copies the waveform that is in the COM 'Raw_data(*)' array out to the
2700    ! instrument's waveform memory number 1, and sets the timebase to match
2710    ! the time range in the Preamble(*).  'Ppa' is the I/O path to the HP 8990A.
2720    !
2730    COM /Meas_data/ REAL Preamble(1:10),INTEGER Raw_data(1:1024)
2740    !
2750    DISP "Downloading the waveform to Waveform Memory #1..."
2760    !
2770    ! *** Specify the measurement source as waveform memory number 1, so
2780    ! *** the output of the waveform data goes into the right place.
2790    OUTPUT @Ppa;"WAV:SOUR WMEM1 "         ! Switch source to waveform mem1
2800    !
2810    ! *** First the preamble is downloaded.  The HP 8990A uses the info in this
2820    ! *** array to set y-axis scaling, & number of pts on x-axis.  However,
2830    ! *** values for x-axis are still determined by the current timebase
2840    ! *** setting (see below).
2850    OUTPUT @Ppa;"WAV:PREAMBLE ";Preamble(*)! Download the new preamble first
2860    !
2870    ! *** Next, the raw data for the waveform is downloaded to the waveform
2880    ! *** memory that was chosen above.  Block formatting is used, so the
2890    ! *** number of bytes is determined as twice the number of 16-bit words
2900    ! *** being sent-- 2*Preamble(3).  The array is resized to match the
2910    ! *** current number of points, and then each element is output as a
2920    ! *** 16-bit signed integer using the 'W' format specifier.  The
```

```
2930    ! *** 'SC()' in the device selector prevents re-addressing of the bus
2940    ! *** devices in the middle of the message, which saves a little time.
2950    OUTPUT @Ppa USING "#,K";"WAV:DATA " ! Output cmd part of data block.
2960    OUTPUT SC(@Ppa) USING "#,K,ZZZZZZZZ";"#8",2*Preamble(3)
2970    REDIM Raw_data(1:Preamble(3))       ! Re-size data array for output
2980    OUTPUT SC(@Ppa) USING "W";Raw_data(*)! Send the data out next.
2990    !
3000    ! *** Since all measurements in this demo are done on the waveform
3010    ! *** memory, display of the acquired channel is turned off and the
3020    ! *** waveform memory with the data to be measured is displayed.
3030    OUTPUT @Ppa;"BLANK CHAN1; BLANK CHAN2; BLANK CHAN3; BLANK CHAN4"
3040    OUTPUT @Ppa;"VIEW WMEM1"
3050    !
3060    ! *** Finally, set up the timebase.  As mentioned above, this is
3070    ! *** necessary because all measurements, be they on functions OR
3080    ! *** waveform memory OR channels, rely on the current timebase
3090    ! *** setting-- not on the preamble that may have been saved with
3100    ! *** the waveform!
3110    OUTPUT @Ppa;"TIM:RANG ";Preamble(3)*Preamble(5)! Set timebase range up
3120    OUTPUT @Ppa;"TIM:DEL ";(Preamble(3)-Preamble(7))*Preamble(5)/2+Preamble(6)
3130    OUTPUT @Ppa;"TIM:REF CENT"          ! Set timebase reference to center.
3140  SUBEND
3150  !
3160  !
3170 Find_edges:DEF FNFind_edges(@Ppa,Tr,Tf,Source$) != = = = = = = = = = = = =
3180    ! Use the HP 8990A measurement on the named Source$ to find the times when
3190    ! the pulse risetime & falltime passes thru the half-amplitude point.
3200    ! The start & stop time markers are left on the two times, and the two
3210    ! times are returned in Tr and Tf, in seconds.
3220    !
3230    !   RETURN STATUS VALUE:
3240    !   -MAXREAL ...The full 'on' part of the pulse with rising & falling
3250    !               edge couldn't be found on the specified source.
3260    !        0     ...Indicates that the edges were found.
3270    !
3280    DISP "Finding edges of pulse on HP 8990A..."
3290    OUTPUT @Ppa;"MEAS:SOUR "&Source$    ! Measurements done on this source
3300    !
3310    ! *** First find amplitude half-way between the top & bottom of the
3320    ! *** pulse.  This is done by using VTOP & VBASE, which do histogram
3330    ! *** determined evaluations to derive values for the statistical top
3340    ! *** and bottom of the pulse.
3350    OUTPUT @Ppa;"MEAS:VTOP?;VBASE?"     ! Query amplitude of top & bottom.
3360    ENTER @Ppa;Vtop,Vbase               ! Get histogram top & base back
3370    Vmid=(Vtop+Vbase)/2                 ! Compute midpoint amplitude of sig.
3380    !
```

```
3390    ! *** Next, locate the first rising edge of the pulse, and the
3400    ! *** following falling edge.  If the first falling edge happens to
3410    ! *** come first, then the left side of the display begins with the
3420    ! *** pulse already in it's high state, and another
3430    ! *** Time-at-vertical-value measurement is done to find the NEXT
3440    ! *** falling edge.  This makes sure that Tr and Tf are both referring
3450    ! *** to the same on-part of the pulse.
3460    OUTPUT @Ppa;"MEAS:TVER? ";Vmid;", +1"! Ask for time of 1st rising edge.
3470    ENTER @Ppa;Tfirst_rising            !  and read it back from the HP 8990A.
3480    !
3490    OUTPUT @Ppa;"MEAS:TVER? ";Vmid;", -1"! Ask for pulse's 1st falling edge
3500    ENTER @Ppa;Tfirst_falling           !  and read it back from the HP 8990A.
3510    !
3520    IF (Tfirst_falling<Tfirst_rising) THEN ! Display starts @left on pulse.
3530      OUTPUT @Ppa;"MEAS:TVER? ";Vmid;", -2"! Get falling edge after rising.
3540      ENTER @Ppa;Tfirst_falling         !  and read it back from the HP 8990A.
3550    END IF
3560    !
3570    ! *** It is possible that there was no pulse on the display.  If that
3580    ! *** was the case, one or both of the rising & falling edge
3590    ! *** measurements that were done above would have returned the
3600    ! *** 'not-available' values for the rising & falling edge.  If that
3610    ! *** happens, return a -MAXREAL to show that no pulse was found.  IF
3620    ! *** IT WAS FOUND OK, then return the time where the middle of the
3630    ! *** pulse happened, which is assumed to = avg of the times of
3640    ! *** occurrence for the rising & falling edge.  Also leave the start and
3650    ! *** stop time marker on each edge to show which edges were chosen.
3660    IF (Tfirst_falling=9.99999E+37 OR Tfirst_rising=9.99999E+37) THEN
3670      PRINT "Sorry.  Could not find one complete 'on' pulse on screen."
3680      PAUSE                             ! Wait for operator to read the message.
3690      RETURN -MAXREAL                   ! & then return the error ret code.
3700    ELSE                                ! Else both times are valid times
3710      Tr=Tfirst_rising                  ! Report rising edge time back and
3720      Tf=Tfirst_falling                 !  also the falling edge too.
3730      OUTPUT @Ppa;"MEAS:TSTA ";Tr       ! Move start marker to rising edge.
3740      OUTPUT @Ppa;"MEAS:TSTO ";Tf       ! Move stop marker to falling edge.
3750      RETURN 0                          ! Return an 'ok it passed' status
3760    END IF
3770  FNEND
3780    !
3790 Measure_slope:SUB Measure_slope(@Ppa,Slope,Intercept,Tstart,Tstop,Echo_results)
3800  ! Do a linear regression curve fit on the top of the pulse in Raw_data(*)
3810  ! from the 'Tstart' to 'Tstop' times.  The slope and y-axis intercept of
3820  ! the resulting datafit to a line is returned as 'Slope' and 'Intercept'.
3830  ! The stop marker on the HP 8990A is moved to each examined point if the
3840  ! Echo_results flag is non-zero.
```

```
3850    !
3860    COM /Meas_data/ REAL Preamble(1:10),INTEGER Raw_data(1:1024)
3870    !
3880    DISP "Doing linear regression on computer copy of the pulse..."
3890    I1=FNTime_index(Tstart)             ! Find 'bucket' matching rising edge
3900    I2=FNTime_index(Tstop)              ! Find 'bucket' matching falling edge
3910    !
3920    ! *** The linear regression code contains running sums of various
3930    ! *** combinations of each (x,y) point.  This statement initializes
3940    ! *** those running sums to zero to begin the regression.
3950    CALL Linear_regress(-1,Slope,Intercept,0,0,0) ! Init the curvefiter code
3960    !
3970    ! *** Do the regression for the points in the waveform data array
3980    ! *** which correspond to the Tstart and Tstop times.  The regression
3990    ! *** is biased with an artificial slope, which is removed afterwards.
4000    ! *** The slope is to allow the coefficient of fit number to be safely
4010    ! *** computed without 'blowing up'.
4020    FOR X=I1 TO I2                      ! For each element in meas range,
4030      Y=Raw_data(X)                     ! Look up y-value for this x value
4040      Yfit=Y+X*32767/Preamble(3)        ! Add in an artificial slope.
4050      IF Echo_results THEN              ! ..show where at in regression on HP 8990A.
4060        OUTPUT @Ppa;"MEAS:TSTOP ";FNTime(X)! Move marker to pt being evaluated
4070      END IF
4080      CALL Linear_regress(R2,Slope,Intercept,N,X,Yfit) ! Add next pt into fit
4090      !
4100    NEXT X
4110    Slope=Slope-32767/Preamble(3)         ! Remove the artificial slope.
4120  SUBEND
4130  !
4140  !
4150 Line2wmem2:SUB Line2wmem2(@Ppa,Slope,Intercept,Tr,Tf,Echo_results) ! = = =
4160  ! Save the line indicated by the Slope and Intercept into the waveform
4170  ! memory register 2 on the HP 8990A.
4180    COM /Meas_data/ REAL Preamble(1:10),INTEGER Raw_data(1:1024)
4190    INTEGER Raw_line(1:1024)            ! The output line is built in here
4200    !
4210    DISP "Drawing the linear fit line on top of pulse on HP 8990A Waveform Memory #2..."
4220    !
4230    ! *** Each point along the line, for each time 'bucket', is written to
4240    ! *** the raw waveform array.  All unwritten points are set to -1 so
4250    ! *** the HP 8990A won't try to plot them.  The lines are extended by 10%
4260    ! *** so they slightly overshoot either side.
4270    MAT Raw_line= (-1)                  ! Init all points to no data in them.
4280    Ir=FNTime_index(Tr)                 ! Get index in Raw_line wrt Trising
4290    If=FNTime_index(Tf)                 ! Get index in Raw_line wrt Tfalling
4300    Ir=MAX(1,Ir-.1*Preamble(3))         ! Move left index down 10% of x rng.
```

```
4310    If=MIN(Preamble(3),If+.1*Preamble(3))! & right index up by 10% of x rng.
4320    FOR I=Ir TO If                       ! For each pt of line that's in rng,
4330      Raw_line(I)=MIN(32767,MAX(0,Slope*I+Intercept))  ! Draw to that point.
4340    NEXT I
4350    !
4360    ! *** Output the scaling information for this configuration.  This
4370    ! *** simply re-uses the previously determined scaling info.
4380    OUTPUT @Ppa;"WAV:SOUR WMEM2 "        ! Switch source to waveform mem2
4390    OUTPUT @Ppa;"WAV:PREAMBLE ";Preamble(*)! Download the new preamble first
4400    !
4410    ! *** Output the raw waveform data, which contains the line that is
4420    ! *** being plotted.  Block formatting is used, so the
4430    ! *** number of bytes is determined as twice the number of 16-bit words
4440    ! *** being sent-- 2*Preamble(3).  The array is resized to match the
4450    ! *** current number of points, and then each element is output as a
4460    ! *** 16-bit signed integer using the 'W' format specifier.  The
4470    ! *** 'SC()' in the device selector prevents re-addressing of the bus
4480    ! *** devices in the middle of the message, which saves a little time.
4490    OUTPUT @Ppa USING "#,K";"WAV:DATA " ! Output cmd part of data block.
4500    OUTPUT SC(@Ppa) USING "#,K,ZZZZZZZZ";"#8",2*Preamble(3)! Block header
4510    REDIM Raw_line(1:Preamble(3))        ! Re-size data array for output
4520    OUTPUT SC(@Ppa) USING "W";Raw_line(*)! Send the 16-bit data out next.
4530    OUTPUT @Ppa;"VIEW WMEM2"             ! Display waveform memory num 2
4540    !
4550  SUBEND
4560    !
4570    !
4580 Measure_sides:SUB Measure_sides(@Ppa,Slope_r,Intercept_r,Slope_f,Intercept_f)
4590    ! Measure the slope and intercept of the rising and falling edges of
4600    ! the pulse.  The 10% and 50% amplitude points on the rising and
4610    !falling edges are assumed.
4620    !
4630    COM /Meas_data/ REAL Preamble(1:10),INTEGER Raw_data(1:1024)
4640    DISP "Measuring slopes of rising & falling edge using the HP 8990A..."
4650    !
4660    ! *** Switch to waveform memory 1, and determine the amplitude's
4670    ! *** 50% point, and the 10% point, in vertical units of volts or watts.
4680    OUTPUT @Ppa;"MEAS:SOUR WMEM1"        ! Switch back to measuring memory1.
4690    OUTPUT @Ppa;"MEAS:VMAX?; :MEAS:VMIN?"! Measure max & min of the pulse
4700    ENTER @Ppa;Vmax,Vmin                 ! Read max & min vertical values.
4710    V_50=Vmin+.5*(Vmax-Vmin)             ! Compute the 50% point of max value
4720    V_10=Vmin+.1*(Vmax-Vmin)             ! The ampl where pulse is @10%
4730    !
4740    ! *** Determine the times when the 10% and 50% power points are
4750    ! *** traversed in the rising and falling edges of the pulse.
4760    OUTPUT @Ppa;"MEAS:TVER? ";V_50;", -1"! Find first falling edge on screen
```

```
4770   ENTER @Ppa;Tf_50
4780   OUTPUT @Ppa;"MEAS:TVER? ";V_10;", -1"   ! Get time of ampl 10%, 1st Tr
4790   ENTER @Ppa;Tf_10
4800   !
4810   OUTPUT @Ppa;"MEAS:TVER? ";V_50;", +1"! And find first rising edge on dsp
4820   ENTER @Ppa;Tr_50
4830   OUTPUT @Ppa;"MEAS:TVER? ";V_10;", +1"! 1st rising @time where ampl=10%.
4840   ENTER @Ppa;Tr_10
4850   !
4860   IF Tf_10<Tr_10 THEN      !Pulse began it's on state off screen to left.
4870     OUTPUT @Ppa;"MEAS:TVER? ";V_50;", -2"! Find 2nd falling edge on screen
4880     ENTER @Ppa;Tf_50
4890     OUTPUT @Ppa;"MEAS:TVER? ";V_10;", -2"! Time @ampl=10%, 2ndfalling edge
4900     ENTER @Ppa;Tf_10
4910   END IF
4920   !
4930   ! *** Convert the real-unit values for the above points into 'raw
4940   ! *** values' for time index and y-axis values in wave:data.
4950   V_10=FNVert_rawval(V_10)
4960   V_50=FNVert_rawval(V_50)
4970   Tr_10=FNTime_index(Tr_10)
4980   Tr_50=FNTime_index(Tr_50)
4990   Tf_10=FNTime_index(Tf_10)
5000   Tf_50=FNTime_index(Tf_50)
5010   !
5020   ! *** Using the 10 & 50 percent points on each of the rising and
5030   ! *** falling edges, the slope and intercept for the rising & falling
5040   ! *** edges can be computed.  Note-- if the risetime or falltime is
5050   ! *** very steep, the line is vertical.  The slope is set to
5060   ! *** MAXREAL, and the intercept is set to the X-VALUE, where the slope
5070   ! *** is infinite.
5080   IF Tr_50Tr_10 THEN                ! Then the risetime is nonzero.
5090     Slope_r=(V_50-V_10)/(Tr_50-Tr_10) ! Slope of rising edge.
5100     Intercept_r=V_50-Slope_r*Tr_50    ! Intercept of rising edge.
5110   ELSE                                ! ELSE the edge is vertical line.
5120     Slope_r=MAXREAL                   ! Infinite slope if line vertical.
5130     Intercept_r=Tr_10                 ! Remember x-axis crossing index.
5140   END IF
5150   !
5160   IF Tf_50Tf_10 THEN                ! Then falltime is nonzero value.
5170     Slope_f=(V_50-V_10)/(Tf_50-Tf_10) ! Slope of falling edge.
5180     Intercept_f=V_50-Slope_f*Tf_50    ! Intercept of falling edge.
5190   ELSE                                ! ELSE the edge is vertical line.
5200     Slope_f=MAXREAL                   ! Infinite slope if line vertical.
5210     Intercept_f=Tf_10                 ! Remember x-axis crossing index.
5220   END IF
```

```
5230    !
5240  SUBEND
5250    !
5260    !
5270 Sides2wmem3:SUB Sides2wmem3(@Ppa,Slope_r,Intercept_r,Slope_f,Intercept_f)
5280    ! Write both the risetime and falltime fit lines, whose slope and
5290    ! y-axis intercepts are passed in, out to waveform memory 3 of the
5300    ! HP 8990A.  If the slope is infinite (ie, it's equal to MAXREAL), then
5310    ! the 'intercept' value is used as the x-value where that slope
5320    ! is infinite.  For non-infinite slopes, slope & intercept are as
5330    ! expected for any y=mx+b line.  Finally, scaling info is derived
5340    ! from the already defined Preamble(*).  The lines are also drawn
5350    ! to the computer's display.
5360    COM /Meas_data/ REAL Preamble(1:10),INTEGER Raw_data(1:1024)
5370    !
5380    INTEGER Raw_line(1:1024)            ! The output line is built in here
5390    !
5400    DISP "Drawing slope lines for rising & falling edges to HP 8990A Waveform Memory #3..."
5410    !
5420    ! *** Initialize the raw data output write array so that all points
5430    ! *** which are not written to are not drawn on the HP 8990A display.
5440    MAT Raw_line= (-1)                  ! Init all points to no data in them.
5450    !
5460    ! *** Draw the rising edge into the raw data array.  Handle the case
5470    ! *** where the timebase was too small to resolve the risetime, which
5480    ! *** means the slope is infinite (aka MAXREAL).
5490    IF Slope_r<>MAXREAL THEN            ! If the slope is not infinite,
5500      X=(0-Intercept_r)/Slope_r         ! x-index where edge would hit y=0.
5510      Y=0                               ! Start by drawing line @bottom
5520      Xstart=X                          ! Remember the starting index value
5530      WHILE (Y<=32767)
5540        Raw_line(X)=Y                   ! Fill in the next part of edge
5550        X=X+1                           ! Increm to next pt to update.
5560        Y=Y+Slope_r                     ! Incr y by slope, deltax = 1.
5570      END WHILE
5580      Raw_line(X)=32767                 ! Draw the last of pulse, to top.
5590    ELSE                                ! ELSE the slope is infinite @T_rise
5600      Raw_line(MAX(1,Intercept_r-1))=0  ! Start by positioning @bottom.
5610      Raw_line(Intercept_r)=32767       ! And extend next point to top.
5620    END IF
5630    !
5640    ! *** Similar to the above, now draw the falling edge's line
5650    ! *** approximation into the waveform raw data array.
5660    IF Slope_f<>MAXREAL THEN            ! If the slope is not infinite,
5670      X=(32767-Intercept_f)/Slope_f     ! x-index where edge would hit y=top
5680      Y=32767
```

```
5690      Xstart=X                          ! Remember the starting x-index value
5700      WHILE (Y>=0)
5710        Raw_line(X)=Y                   ! Fill in the next part of edge
5720        X=X+1                           ! Increm to next pt to update.
5730        Y=Y+Slope_f                     ! Increm y by slope, since deltax=1
5740      END WHILE
5750      Raw_line(X)=0                     ! Bring rest of line to bottom.
5760    ELSE
5770      Raw_line(MAX(1,Intercept_f-1))=32767  ! Start by positioning at top
5780      Raw_line(Intercept_f)=0           ! And extend next point to bottom.
5790    END IF
5800    !
5810    ! *** Output the scaling information and data to waveform memory 3.
5820    OUTPUT @Ppa;"WAV:SOUR WMEM3 "       ! Switch source to waveform mem3
5830    OUTPUT @Ppa;"WAV:PREAMBLE ";Preamble(*)! Download the new preamble first
5840    !
5850    ! *** Output the raw waveform data, which contains both lines that are
5860    ! *** to be plotted.  Block formatting is used, so the
5870    ! *** number of bytes is determined as twice the number of 16-bit words
5880    ! *** being sent-- 2*Preamble(3).  The array is resized to match the
5890    ! *** current number of points and then each element is output as a
5900    ! *** 16-bit signed integer using the 'W' format specifier.  The
5910    ! *** 'SC()' in the device selector prevents re-addressing of the bus
5920    ! *** devices in the middle of the message, which saves a little time.
5930    OUTPUT @Ppa USING "#,K";"WAV:DATA " ! Output cmd part of data block.
5940    OUTPUT SC(@Ppa) USING "#,K,ZZZZZZZZ";"#8",2*Preamble(3)! Block header
5950    REDIM Raw_line(1:Preamble(3))       ! Re-size data array for output
5960    OUTPUT SC(@Ppa) USING "W";Raw_line(*)! Send the data out next.
5970    OUTPUT @Ppa;"VIEW WMEM3"            ! Enable the waveform memory num 3
5980    !
5990  SUBEND
6000    !
6010    !
6020 Vertmarkers:SUB Vertmarkers(@Ppa,V_1,V_2)  ! = = = = = = = = = = = = = = =
6030    ! Move the two vertical markers to the new specified amplitudes.  The
6040    ! passed in values correspond to the y-scaling in WAVE:DATA datablocks.
6050    !The scaling is determined by the Preamble(*) array.
6060    COM /Meas_data/ REAL Preamble(1:10),INTEGER Raw_data(1:1024)
6070    !
6080    OUTPUT @Ppa;"MEAS:VSTOP ";FNVertical(V_1);"; :MEAS:VSTART ";FNVertical(V_2)
6090    !
6100  SUBEND
6110    !
6120    !
6130 Build_demo:SUB Build_demo              ! * ** * = = = = = = = = = = = = = =
6140    ! Build up the Data_array(*) with a droopy pulse which can be measured
```

```
6150  ! during the demo.  The Waveform preamble is built up with values that
6160  ! correspond to a real hw state.  The data array is filled
6170  ! per the step specs in the preamble.  The data is built with a Rising
6180  ! Top, and Falling edge.
6190    COM /Meas_data/ REAL Preamble(1:10),INTEGER Raw_data(1:1024)
6200    !
6210    DISP "Building a sample waveform in BASIC..."
6220    !
6230    Rmin=9000                          ! Raw value of bottom of pulse
6240    Rmax=23000                         ! Raw value for top of the pulse
6250    Ir=50                              ! Time index where up pulse happens
6260    If=320                             ! Time index where pulse goes down.
6270    Tr=1.0E-5                          ! Risetime assumed at 10 useconds.
6280    Tf=5.0E-5                          ! Falltime is set at 50 microseconds
6290    Droop=-.3                          ! Assume a 30% droop for simulation
6300  !
6310  ! *** Next, Initialize the preamble array for this example.  The values
6320  ! *** assigned to Preamble(*) are:
6330  !    2= Word format                     2E-6= Xincr     1.2207E-3= Yincr
6340  !    1= Normal (not avg or envelope)  -100E-6= Xorigin           0= Yorig
6350  !  500= Num pts per waveform               0= Xreference    16384= Yref.
6360  !    1= Num samples per point
6370    DATA 2,1,500,1,  2E-6, -100E-6, 0,   1.2207E-3, 0, 16384
6380    READ Preamble(*)
6390    !
6400    ! *** Finally, set up the raw data waveform buffer for a pulse with
6410    ! *** known droop.  Initialize the array to the low-value for the
6420    ! *** pulse.
6430    MAT Raw_data= (Rmin)               ! Initialize baseline of the pulse.
6440    !
6450    ! *** Construct the rising edge first.
6460    IF Tr=0 THEN Tr=MINREAL            ! If zero was entered, make nonzero
6470    IF Tf=0 THEN Tf=MINREAL            ! If zero was entered, make nonzero
6480    Idelta_r=Tr/Preamble(5)            ! DeltaIndex covering rising edge
6490    R90=Rmin+.9*(Rmax-Rmin)            ! Raw data value for 90% of peak.
6500    R10=Rmin+.1*(Rmax-Rmin)            ! Raw data value for 10% of peak.
6510    Slope_r=(R90-R10)/Idelta_r         ! Slope of rising edge, raw/index.
6520    R=Rmin+Slope_r                     ! First pt of rising edge from base
6530    I=Ir-.5*INT(Idelta_r+.5)+1         ! Index of that 1st pt in raw ary.
6540    WHILE (R<Rmax)                     ! Keep building till @top of pulse
6550      Raw_data(I)=R                    ! Save nxt part, pulse's rising edge
6560      I=I+1                            ! & incr to save nxt pt @next Time.
6570      R=R+Slope_r                      ! Next point will be above previous
6580    END WHILE                          ! Repeat till edge built up to top.
6590    !
6600    ! *** Now draw the top of the pulse.  The top is ramped down by the
```

```
6610    ! *** indicated slope so we have something to measure.  The pulse is
6620    ! *** ramped down until it gets to the array index where the falling
6630    ! *** edge is supposed to go.
6640    Idelta_f=Tf/Preamble(5)             ! x-axis width of falling pulse.
6650    Slope_t=Droop*(Rmax-Rmin)/(If-Idelta_f/2-I)! Slope of top of the pulse
6660    R=Rmax                              ! First point drawn exactly @top.
6670    WHILE (I<If-Idelta_f/2)             ! Go up to just before falling edge
6680      Raw_data(I)=R                     ! Init next pt on waveform for ramp
6690      Raw_data(I)=Raw_data(I)+1000*SIN(2*PI*5*(I-If)/(If-Ir))
6700      I=I+1                             ! Incr to write to next time index
6710      R=R+Slope_t                       ! Decrem top by amount of droop.
6720    END WHILE                           ! Keep going till @start of fall edg
6730    !
6740    ! *** Finally, construct the falling edge of the pulse.  The value of
6750    ! *** the pulse is decremented until it gets back to the Rmin baseline.
6760    R=R-Slope_t                         ! Back out the last unused topvalue
6770    Slope_f=(Rmin-R)/(Idelta_f)         ! Slope of falling edge, raw/index.
6780    WHILE (R>Rmin)                      ! Keep going till back @bottom.
6790      Raw_data(I)=R                     ! Next data point on falling edge.
6800      R=R+Slope_f                       ! Precompute next falling edge pt.
6810      I=I+1                             ! Increm to next time index.
6820    END WHILE
6830  !
6840    DISP
6850  SUBEND
6860  !
6870  !
6880  ! = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = =
6890 Acquire_data:SUB Acquire_data(@Ppa,Channel_num)
6900  ! Acquire the pulse to be measured using the HP 8990A.  The algorithm
6910  ! used is as follows:
6920  !   o Set up the hw to the right state(s) by executing an HP 8990A autoscale
6930  !   o Acquire the data for measurement over HP-IB with a DIGITIZE.
6940  !   o Fetch the data into the computer for processing.
6950  !       (a) The Preamble is read, to tell x,y axis scaling, num pts,...
6960  !       (b) The raw waveform data record is read into the computer.
6970  !       (c) The waveform record is scaled to actual powers.
6980  !
6990    COM /Meas_data/ REAL Preamble(1:10),INTEGER Raw_data(1:1024)
7000    DISP "Acquiring data on channel";TRIM$(VAL$(Channel_num))&"..."
7010    !
7020    ! *** Autoscale the hardware on the HP 8990A to quickly set up and acquire
7030    ! *** a signal.  Then do a DIGITIZE so it's safe to do measurements.
7040    ! *** ALWAYS do a digitize before doing any HP 8990A measurements over HP-IB!
7050    Chanspec$="CHAN"&VAL$(Channel_num)  ! Form CHANNEL specifier variable.
7060    OUTPUT @Ppa;"AUTOSCALE"             ! Autoscale the signal to set hw up
```

```
7070    OUTPUT @Ppa;"DISP:FORM 1"           ! Set for 1 screen display format.
7080    OUTPUT @Ppa;"DIGITIZE "&Chanspec$   ! Acquire waveform for measuring.
7090    !
7100    ! *** Next, reset the time scale so the on part of the pulse is put
7110    ! *** on the screen in reasonable resolution.  This is done by using
7120    ! *** the HP 8990A to measure the times rising & falling edges, and then
7130    ! *** specifying timebase settings which will place that pulse over
7140    ! *** most of the display.  Since the HP 8990A setup is changed with this
7150    ! *** change of the timebase, another DIGITIZE is REQUIRED and is done
7160    ! *** before any other measurements can be done.
7170    IF FNFind_edges(@Ppa,Tr,Tf,Chanspec$)-MAXREAL THEN ! Find pulse edges.
7180      T_range=1.5*(Tf-Tr)               ! Go for 25% overrange on each side
7190      OUTPUT @Ppa;"TIM:DEL ";(Tr+Tf)/2  ! Center the pulse in the window.
7200      OUTPUT @Ppa;"TIM:RANG ";T_range   ! And set it in the instrument
7210      OUTPUT @Ppa;"TIM:REF CENT"        ! Set so center 'DEL' above centers.
7220      OUTPUT @Ppa;"DIGITIZE "&Chanspec$ ! Re-acquire again w/new centering.
7230    END IF
7240    !
7250    ! *** With the newly acquired waveform now acquired in the channel buffer,
7260    ! *** the scaling information in the preamble and raw waveform data
7270    ! *** block, are read into the computer.
7280    OUTPUT @Ppa;"WAVEFORM:SOURCE "&Chanspec$  ! Specify where to read from.
7290    OUTPUT @Ppa;"WAVEFORM:PREAMBLE?"          ! Have HP 8990A output preamble.
7300    ENTER @Ppa;Preamble(*)              ! Read waveform preamble back in.
7310    !
7320    ! *** The waveform block header is used to determine the size of the
7330    ! *** waveform data array.  It contains the number of BYTES in the
7340    ! *** data block for the unscaled waveform data.  Since WORD format is
7350    ! *** used (specified in the subpgm Instrument_int), there are 16 bits
7360    ! *** (=2 bytes) for each data element.  Therefore, the number of data
7370    ! *** elements is twice the number of bytes indicated in the waveform
7380    ! *** header, which is used to set the size of the waveform data
7390    ! *** array.  Other notes-- the SC() construct is used to prevent
7400    ! *** re-addressing of the 8990 during the block read, which saves a
7410    ! *** minor amount of time.  The waveform data is read in 2 bytes at a
7420    ! *** time by using the W image specifier.
7430    OUTPUT @Ppa;"WAVEFORM:DATA?"        ! Tell HP 8990A to output it's data now
7440    ENTER @Ppa USING "#,10A";Header$    ! Read waveform header: '#0000.. "
7450    Num_words=VAL(Header$[3])/2         ! Number of 16-bit words in block.
7460    REDIM Raw_data(1:Num_words)         ! Resize for num of samples in wave
7470    ENTER SC(@Ppa) USING "W";Raw_data(*)  ! Read the waveform data.
7480  SUBEND
7490  !
7500  !
7510 Linear_regress:SUB Linear_regress(R2,Slope,Intercept,N,X,Y)
7520  ! This can be called in one of three ways:
```

```
7530  !  R2=-1:  Initializes the linear curve fit summations to startup (zero).
7540  !  R2=-2:  Forget the (x,y) value that was passed in before & is being
7550  !          passed in again.  This removes a data pt from the curve fit.
7560  !  R2>0:   The x,y pt is added to the summation, and an R2 is returned
7570  !          to the caller (as is N, Slope, & Intercept) as the coef of fit.
7580  ! This routine came from the HP25C Applications Programs manual, page 88.
7590  !
7600     COM /Linear_regess/ S_xy,S_x,S_y,S_xx,S_yy
7610     IF R2-1 THEN GOTO Not_initialize  ! ..assume this is initialization.
7620     N=0                               ! Init num of pts now in fit to 0.
7630     S_xy=0                            ! Init summation of xi * yi to 0
7640     S_x=0                             ! Init summation of xi to zero.
7650     S_y=0                             ! Init summation of yi to zero.
7660     S_xx=0                            ! Init summation of xi squared ->0
7670     S_yy=0                            ! Init summation of yi squared ->0
7680     SUBEXIT
7690  Not_initialize:  !
7700     !
7710     IF R2-2 THEN GOTO Not_remove      ! ..ELSE add nxt point into fit.
7720     S_x=S_x-X                         ! Remove last summation of xi
7730     S_y=S_y-Y                         ! Remove last summation of yi
7740     S_xy=S_xy-X*Y                     ! Remove last summation of xi * yi
7750     S_xx=S_xx-X*X                     ! Remove last summation of xi * xi
7760     S_yy=S_yy-Y*Y                     ! Remove last summation of yi * yi
7770     N=N-1                             ! Decrem num of fitted pts by one.
7780     SUBEXIT
7790  Not_remove:  !
7800        !
7810     S_x=S_x+X                         ! Add this xi into summation.
7820     S_y=S_y+Y                         ! Add this yi into yi summation.
7830     S_xy=S_xy+X*Y                     ! Add this pt into xi*yi summation.
7840     S_xx=S_xx+X*X                     ! Add this pt into xi*xi summation.
7850     S_yy=S_yy+Y*Y                     ! Add this pt into yi*yi summation.
7860     N=N+1                             ! Incr num of pts in curve fit by 1
7870     IF N<5 THEN SUBEXIT               ! Do no computations for 1st 5 pts.
7880     Numerator=S_xy-S_x*S_y/N          ! Compute numerator of slope term
7890     Denominator=S_xx-S_x*S_x/N        ! and also denominator of slope
7900     Slope=Numerator/Denominator       ! Then compute the actual slope.
7910     Intercept=S_y/N-Slope*S_x/N       ! Compute y-axis intercept of fit.
7920     R2=Slope*Numerator/(S_yy-S_y*S_y/N)
7930  SUBEND
7940  !
7950  !
7960 Y_intersect:DEF FNY_intersect(Slope_1,Intercept_1,Slope_2,Intercept_2)
7970  ! Two lines whose slope and intercept have been previously determined are
7980  ! passed in as their Slope and y-axis intercept values.  The two
```

```
7990  ! equations are solved to return the y-value where both of these lines
8000  ! cross each other.  The equation for the returned value was derived
8010  ! by solving y=slope_1*x + intercept_1 and y=slope_2*x + Intercept_2
8020  ! simultaneously for y.
8030  !
8040    IF Slope_1=MAXREAL THEN RETURN Slope_2*Intercept_1+Intercept_2
8050    IF Slope_2=MAXREAL THEN RETURN Slope_1*Intercept_2+Intercept_1
8060    RETURN (Intercept_1*Slope_2-Intercept_2*Slope_1)/(Slope_2-Slope_1)
8070  FNEND
8080  !
8090  !
8100 Vertical:DEF FNVertical(Raw_value)         ! = = = = = = = = = = = = = = =
8110  ! Use the y-axis scaling factors in the Preamble to return the vertical
8120  ! power or voltage which corresponds to the passed in Raw_value.  The
8130  ! Raw_value is used to compute the actual units by:
8140  !
8150  !      Pwr or volts = (Raw_value - Yref) * YIncr  + Yorigin
8160  !
8170    COM /Meas_data/ REAL Preamble(1:10),INTEGER Raw_data(1:1024)
8180    RETURN ((Raw_value-Preamble(10))*Preamble(8)+Preamble(9))
8190  FNEND
8200  !
8210  !
8220 Time:DEF FNTime(Time_index)                ! = = = = = = = = = = = = = = =
8230  ! The time relative to the trigger for each of the waveform data record
8240  ! samples is determined from the position of the number within that
8250  ! record and the x-axis scaling numbers in the preamble.  The time
8260  ! for a given index in this program's waveform record array is:
8270  !
8280  !    Time = ( -1+Index - Xref) * Xincrement  + Xorigin
8290  !
8300  ! Note that 'Index' is offset by one, since the Waveform Data array in
8310  ! this program goes from ONE to N, not from ZERO to N-1.
8320  !
8330    COM /Meas_data/ REAL Preamble(1:10),INTEGER Raw_data(1:1024)
8340    RETURN ((-1+Time_index-Preamble(7))*Preamble(5)+Preamble(6))
8350  FNEND
8360  !
8370 Vert_rawval:DEF FNVert_rawval(Vertical)   ! = = = = = = = = = = = = = = =
8380  ! Work back through the preamble info to find the vertical raw value that
8390  ! in the raw data record corresponds to the real voltage or power.
8400  ! This is the inverse of the FNVertical() function.  The returned raw
8410  ! value is clipped to prevent any rounding from returning an out of range
8420  ! result.
8430    COM /Meas_data/ REAL Preamble(1:10),INTEGER Raw_data(1:1024)
8440    RETURN MAX(MIN((Vertical-Preamble(9))/Preamble(8)+Preamble(10),32767),0)
```

```
8450  FNEND
8460     !
8470 Time_index:DEF FNTime_index(Seconds)     ! = = = = = = = = = = = = = = = =
8480  ! Work back thru the preamble to find which element in the raw_data() ary
8490  ! corresponds to the named time, which is in seconds.  This is the inverse
8500  ! of the FNTime() function.
8510    COM /Meas_data/ REAL Preamble(1:10),INTEGER Raw_data(1:1024)
8520    RETURN INT((Seconds-Preamble(6))/Preamble(5)+Preamble(7)+1)
8530  !
8540  FNEND
8550  !
8560  !
```

Index

D

K

L

M

Q

R

S

T

U

V

W

X

Y

Z

Manufacturing
Part No.
08990-90002