Agilent N77*xx* Series

Programming Guide

Agilent Technologies

Notices

Manual Part Number

N7744-90C01

Edition

Fifth edition, September 2011

Agilent Technologies Deutschland GmbH
Herrenberger Str. 130
71034 Böblingen, Germany

Warranty

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

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

Agilent warrants that its software and firmware designated by Agilent for use with an instrument will execute its programming instructions when properly installed on that instrument. Agilent does not warrant that the operation of the instrument, 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.
No other warranty is expressed or implied. Agilent Technologies specifically disclaims the implied warranties of Merchantability and Fitness for a Particular Purpose.

Technology Licenses

The hardware and/or software described in this document are furnished under a license and may be used or copied only in accordance with the terms of such license.

Restricted Rights Legend

If software is for use in the performance of a U.S. Government prime contract or subcontract, Software is delivered and licensed as “Commercial computer software” as defined in DFAR 252.227-7014 (June 1995), or as a “commercial item” as defined in FAR 2.101(a) or as “Restricted computer software” as defined in FAR 52.227-19 (June 1987) or any equivalent agency regulation or contract clause. Use, duplication or disclosure of Software is subject to Agilent Technologies’ standard commercial license terms, and non-DOD Departments and Agencies of the U.S. Government will receive no greater than Restricted Rights as defined in FAR 52.227-19(c)(1-2) (June 1987). U.S. Government users will receive no greater than Limited Rights as defined in FAR 52.227-14 (June 1987) or DFAR 252.227-7015 (b)(2) (November 1995), as applicable in any technical data.

Exclusive Remedies

The remedies provided herein are Buyer's sole and exclusive remedies. Agilent Technologies 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 Agilent Technologies products. For any assistance contact your nearest Agilent Technologies Sales and Service Office.

Certification

Agilent Technologies Inc. certifies that this product met its published specifications at the time of shipment from the factory. Agilent Technologies 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 Institutes’s calibration facility, and to the calibration facilities of other International Standards Organization members.

ISO 9001 Certification

Produced to ISO 9001 international quality system standard as part of our objective of continually increasing customer satisfaction through improved process control.

Safety Notices

CAUTION

A **CAUTION** notice calls attention to an operating procedure, practice, or the like that, if not correctly performed or adhered to, could result in damage to the product or loss of important data. Do not proceed beyond a **CAUTION** notice until the indicated conditions are fully understood and met.

WARNING

A WARNING notice calls attention to an operating procedure, practice, or the like that, if not correctly performed or adhered to, could result in personal injury or death. Do not proceed beyond a WARNING notice until the indicated conditions are fully understood and met.

Warnings and Notices

WARNING

To avoid the possibility of injury or death, you must observe the following precautions before switching on the instrument.
Insert the power cable plug only into a socket outlet provided with a protective earth contact. Do not negate this protective action by the using an extension cord without a protective conductor.

WARNING

Never look directly into the end of a fiber or a connector, unless you are absolutely certain that there is no signal in the fiber.

In this Manual

This manual contains information about SCPI commands which can be used to program the following instruments:

- N7744A and N7745A Optical Multiport Power Meters
- N7751A and N7752A Variable Optical Attenuators and 2-Channel Optical Power Meter
- N7761A, N7762A and N7764A Variable Optical Attenuators
- N7766A and N7768A variable Optical Multimode Attenuators
- N7711A and N7714A Tunable Laser System Source
- N773xA Optical Switch

The N773xA Optical Switches are supported by the standard 816x Plug & Play Instrument Driver version 4.4 and higher.

The N7744A/45A Multiport Optical Power Meter and the N7751A/52A/61A/62A/64A/66A/68A are supported by the standard 816x Plug & Play Instrument Driver version 4.3 and higher.

The N7744A and N7745A Optical Multiport Power Meters can also be programmed via the IVI drivers (IVI-COM, IVI-C), available on
http://www.agilent.com/find/ivi-com

Conventions used in this Manual

- All commands and typed text is written in Courier font, for example `INIT[:IMM]`.
- SCPI commands are written in mixed case: text that you MUST print is written in capitals; text which is helpful but not necessary is written in lower case.
  So, the command `INITiate[:IMMediate]` can be entered either as `init[:imm]`, or as `initiate[:immediate]`.
  Every part of a command can be entered in its long form or short form, which can also be mixed: e.g. `initiate:imm` or `init:immediate`. It does not matter whether you enter text using capitals or lower-case letters.
- SCPI commands often contain parts in square brackets. These arguments may be helpful, but they need not be entered.
  So, the command `INITiate[:IMMediate]` can be entered as `init` or `initiate:imm`.
- [n] denotes a subopcode. These are used as indices, and [n] must be replaced with a number. If you do not specify a number, the default value is 1.

- A SCPI command which can be either a command or a query is appended with the text /?.
  So, `:INPut1:ATTenuation/?` refers to both the command `:INPut1:ATTenuation` and the query `:INPut1:ATTenuation?`.
- Strings are always quoted, unless marked as 'unquoted'.
- Indices are counted from 1
- A list consists of comma separated values
- Blocks have the form: `#n1…1d…d`.
  n is the number of digits of *l…l*.
  l…l gives the length of the data part *d…d*.
  For example, `#212Hello world`, where "#" indicates a block, "2" gives the 2 digit length, "12" indicates 12 bytes, and "Hello world" is the12 data bytes.
- Hexadecimal numbers can be given with a leading "#h", for example, "#hff"
- Float numbers are always 32-bit floats, this is especially important where raw binary data blocks are transmitted.
- Binary data is transmitted as big endian; that is, bytes must be swapped for Intel platforms.

Related Manuals

You can find more information about the instrument covered by this manual in the following manuals:

- *The Agilent N77- Series User's Guide.*

NOTE

Refer to the books listed in "Introduction to Programming" on page 11 for additional information about the General Purpose Interface Bus, GPIB.

Contents

1 Introduction to Programming

This chapter gives general information on how to control your instrument remotely.

Descriptions for the actual commands for the instruments are given in the following chapters. The information in these chapters is specific to all instruments addressed in this manual and assumes that you are already familiar with programming the SCPI interface.

LAN Interface

LAN Connection using the Web Interface

Please refer to the Getting Started Guide.

Direct instrument (socket) connection using LAN:

If you are connecting directly to a LAN instrument, disable DHCP on your instrument. Otherwise your computer will wait for the DHCP to time out before it gets an IP address from the product. This timeout can take around 2 minutes.

When connecting directly to a LAN instrument rather than through a server, do the following:

1. Use a crossover LAN cable. This is different than what you would use to connect to a server.
2. Disable DHCP on your instrument.
3. If using VISA or VISA-COM, add your instrument to the "HOSTS" file (see LAN FAQ)

Test your connection by opening a DOS window, (Run -> "cmd" -> OK from the START menu of many Windows operating systems) type ping and the IP address of the product. Once you have established communication with the product, use this IP address in the program for an example of instrument control using sockets.

Agilent instruments have standardized on using port 5025 for SCPI socket services. Once a connection is made you simply send the SCPI strings to the instrument and read back responses over the socket connection. All strings must be terminated with a newline character. All responses from the instrument will be terminated with a newline character.have established communication with the product, use this IP address in the program for an example of instrument control using sockets.

For further information, see the Agilent Technologies USB/LAN/GPIB Connectivity Guide.

http://cp.literature.agilent.com/litweb/pdf/E2094-90009.pdf

LAN Connection using Telnet

When communicating with a LAN instrument using TELNET, do the following.

- Use a LAN cable and connect to the instrument. If doing direct connection, see the note on connecting directly to an instrument without a router.
- Test the connection. Either:
  - Type the IP address into a web browser. The web page should be displayed. Click on “Advanced information” and find the SCPI Telnet Port. For most Agilent products this is 5024.
  - Test your connection by opening a DOS window, (Run -> "cmd" -> OK from the START menu of many Windows operating systems). Type ping and the IP address of the product.
- Run Telnet. (Run -> "telnet" -> OK from the START menu of many Windows operating systems)
- At the command line, type the following: open XXX.XXX.XXX.XXX PPPP where “XXX.XXX.XXX.XXX” is the IP address and “PPPP” is the port (5024 for most Agilent products).
- Enter the commands for the product. CTRL-C is a device clear.
- When finished, close the connection and exit Telnet. To do this using the Microsoft Telnet Client (as described above), close the window.

USB Interface

To use the USB Interface please install the USB Test and Measurement Device Driver which comes with the Agilent I/O Libs.

For further information, see the Agilent Technologies USB/LAN/GPIB Connectivity Guide.

http://cp.literature.agilent.com/litweb/pdf/E2094-90009.pdf

GPIB Interface

GPIB is the interface used for communication between a controller and an external device, such as the the power meter. The GPIB conforms to IEEE Std 488.1-2003, IEEE Std 488.2-1992 standard MC 1.1 and IEC recommendation 625-1.

If you are not familiar with the GPIB, then refer to the following books:

- The International Institute of Electrical and Electronics Engineers. *IEC/IEEE Standard for Higher Performance Protocol for the Standard Digital Interface for Programmable Instrumentation - Part 1: General (Adoption of IEEE Std 488.1-2003). Print ISBN: 2-8318-7440-8*
- The International Institute of Electrical and Electronics Engineers. *Standard Digital Interface for Programmable Instrumentation - Part 2: Codes, Formats, Protocols and Common Commands (Adoption of (IEEE Std 488.2-1992) Print ISBN: 2-8318-7441-6*

To obtain a copy of either of these last two documents, write to:

The Institute of Electrical and Electronics Engineers, Inc.
3 Park Avenue, 17th Floor
New York, NY 10016-5997
USA.

In addition, the commands not from the IEEE-488.2 standard, are defined according to the Standard Commands for Programmable Instruments (SCPI).

For information about SCPI, and SCPI programming techniques, please refer to:

- The IVI Foundation (formerly known as SCPI Consortium): *Standard Commands for Programmable Instruments*. To obtain a copy of this manual, contact the following address:

  IVI Foundation Corporate Office
  Bob Helsel, Director of Services
  IVI Foundation, PO Box 1016
  Niwot, CO 80544-1016

  Web: http://www.ivifoundation.org

Table 1 shows the interface functional subset that the instruments implement.

Table 1 GPIB Capabilities

Mnemonic	Function
SH1	Complete source handshake capability
AH1	Complete acceptor handshake capability
T6	Basic talker; serial poll; no talk only mode; unaddressed to talk if addressed to listen
L4	Basic listener; no listen only mode; unaddressed to listen if addressed to talk
SR0	No service request capability
RL1	Complete remote/local capability
PP0	No parallel poll capability
DC1	Complete device clear capability
DT0	No device trigger capability
C0	No controller capability.

Setting the GPIB Address

There are two ways to set the GPIB address:

- You can set the GPIB address by using the command ":SYSTem:COMMunicate:GPIB[:SELF]:ADDRess" on page 69.
- The most convenient way to set the GPIB address is using the user interface (the N77*xx* Viewer software), running on a PC. You can run this to control the instrument over USB, LAN or GPIB.
- For the N774*x*A Multiport power meters, you can also set the GPIB address from the web interface. See the *User's Guide* for more information.

The default GPIB address is 20.

NOTE

GPIB address 21 is often applied to the GPIB controller. If so, 21 cannot be used as an instrument address.

Message Queues

The instrument exchanges messages using an input and an output queue. Error messages are kept in a separate error queue.

How the Input Queue Works

The input queue is a FIFO queue (first-in first-out). Incoming bytes are stored in the input queue as follows:

1 Receiving a byte:
   - Clears the output queue.
   - Clears Bit 7 (MSB).
2 No modification is made inside strings or binary blocks. Outside strings and binary blocks, the following modifications are made:
   - Lower-case characters are converted to upper-case.
   - The characters 00_{16} to 09_{16} and $0B_{16}$ to $1F_{16}$ are converted to spaces (20_{16}).
   - Two or more blanks are truncated to one.
3 An EOI (End Or Identify) sent with any character is put into the input queue as the character followed by a line feed (LF, $0A_{16}$). If EOI is sent with a LF, only one LF is put into the input queue.
4 The parser starts if the LF character is received or if the input queue is full.

Clearing the Input Queue

Switching the power off, or sending a Device Interface Clear signal, causes commands that are in the input queue, but have not been executed to be lost.

The Output Queue

The output queue contains responses to query messages. The instrument transmits any data from the output queue when a controller addresses the instrument as a talker.

Each response message ends with a carriage return (CR, $0D_{16}$) and a LF ($0A_{16}$), with EOI=TRUE. If no query is received, or if the query has an error, the output queue remains empty.

The Message Available bit (MAV, bit 4) is set in the Status Byte register whenever there is data in the output queue.

The Error Queue

The error queue is 30 errors long. It is a FIFO queue (first-in first-out). That is, the first error read is the oldest error to have occurred. For example:

1. If no error has occurred, the error queue contains:
   ```
   + 0, "No error"
   ```
2. After a command such as wav:pow, the error queue now contains:
   ```
   + 0, "No error"
   -113, "Undefined header"
   ```
3. If the command is immediately repeated, the error queue now contains:
   ```
   + 0, "No error"
   -113, "Undefined header"
   -113, "Undefined header"
   ```

If more than 29 errors are put into the queue, the message:

```
-350, "Queue overflow"
```

is placed as the last message in the queue.

Programming and Syntax Diagram Conventions

A program message is a message containing commands or queries that you send to the instruments. The following are a few points about program messages:

- You can use either upper-case or lower-case characters.
- You can send several commands in a single message. Each command must be separated from the next one by a semicolon (;).
- A command message is ended by a line feed character (LF) or <CR><LF>.
- You can use any valid number/unit combination.

  In other words, 1500NM, 1.5UM and 1.5E-6M are all equivalent.

  If you do not specify a unit, then the default unit is assumed. The default unit for the commands are given with command description in the next chapter.

Short Form and Long Form

The instrument accepts messages in short or long forms.

For example, the message

```
:STATUS:OPERATION:ENABLE 768
```

is in long form.

The short form of this message is

```
:STAT:OPER:ENAB 768
```

In this manual, the messages are written in a combination of upper and lower case. Upper case characters are used for the short form of the message.

For example, the above command would be written

```
:STATus:OPERation:ENABle
```

The first colon can be left out for the first command or query in your message. That is, the example given above could also be sent as

```
STAT:OPER:ENAB 768
```

Command and Query Syntax

All characters not between angled brackets must be sent exactly as shown.

The characters between angled brackets (<...>) indicate the kind of data that you should send, or that you get in a response. You do not type the angled brackets in the actual message.

Descriptions of these items follow the syntax description. The following types of data are most commonly used:

string	is ascii data. A string is contained between double quotes ("...") or single quotes ('...').
value	is numeric data in integer (12), decimal (34.5) or exponential format (67.8E-9).
wsp	is a white space.

Other kinds of data are described as required.

The characters between square brackets ([...]) show optional information that you can include with the message.

The bar (|) shows an either-or choice of data, for example, *a*|*b* means either *a* or *b*, but not both simultaneously.

Extra spaces are ignored, so spaces can be inserted to improve readability.

Units

Where units are given with a command, usually only the base units are specified. The full sets of units are given in the table below.

Table 2 Units and allowed Mnemonics

Unit	Default	Allowed Mnemonics
meters	M	PM, NM, UM, MM, M
decibel	DB	MDB, DB
second	S	NS, US, MS, S
decibel/1mW	DBM	MDBM, DBM
Hertz	HZ	HZ, KHZ, MHZ, GHZ, THZ
Watt	Watt	PW, NW, UW, MW, Watt
meters per second	M/S	NM/S, UM/S, MM/S, M/S

Data Types

With the commands you give parameters to the instrument and receive response values from the instrument. Unless explicitly specified these data are given in ASCII format. The following types of data are used:

- ***Boolean*** data may only have the values 0 or 1.
- ***Integer*** range is given for each individual command.
- ***Float*** variables may be given in decimal or exponential writing (0.123 or 123E-3).
  All ***Float*** values conform to the 32 bit IEEE Standard, that is, all ***Float*** values are returned as 32-bit real values.
- A ***string*** is contained between double quotes ("...") or single quotes (‘...’). When the instrument returns a string, it is always included in " " and terminated by `<END>`.
- When a ***register*** value is given or returned (for example *ESE), the ***decimal*** values for the single bits are added. For example, a value of nine means that bit 0 and bit 3 are set.
- Larger blocks of data are given as ***Binary Blocks***, preceded by “#<*H*><*Len*><*Block*>”, terminated by `<END>`; <*H*> represents the number of digits, <*Len*> represents the number of bytes, and <*Block*> is the data block. For example, for a ***Binary Block*** with 1 digit and 6 bytes this is: `#16TRACES<END>`.

Slot and Channel Numbers

Many of the N77-series instruments have multiple sections, like the 4-port N7744A power meter. These sections can be addressed by many commands separately and are referred to here as modules for compatibility with the 816x modular instruments.

Each module is identified by a slot number and a channel number. For commands that require you to specify a channel, the slot number is represented by [*n*] in a command and the channel number is represented by [*m*].

The slot number represents the module’s position in the mainframe. These are:

- from 1 to 4 for the Agilent N7744A
- from 1 to 8 for the Agilent N7745A,
- from 1 to 2 for the attenuator in the N7751A or N7761A

- from 1 to 2 and from 3 to 4 respectively for the two attenuators in the N7752A or N7762A or N7766A - each attenuator occupies 2 slots, and all commands are valid for either of these slots.
- from 5 to 6 for the power meters in the N7751A or N7752A
- from 1 to 2, 3 to 4, 5 to 6 and 7 to 8 respectively for the four attenuators in the N7764A or N7768A - each attenuator occupies 2 slots, and all commands are valid for either of these slots.

To get the number of installed Switches in a N773xA Device use the *OPT? Command.

Exampe for one installed Switch:

```
*opt?
<- N7734A-008
```

Exampe for two installed Switches:

```
*opt?
<- N7736A-008,N7736A-008
```

These numbers are displayed on the front panel beside each module slot.

Channel numbers are currently not used and are optional in all commands for the N77xx Series. They are accepted to ensure compatibility with the Agilent 816x instruments.

For example, if you want to query slot 1 with the command, :SENSe[n][:CHANnel[m]]:POWer:WAVelength?, you should send the command:

```
:sens1:pow:wav?
```

NOTE

If you do not specify a slot or channel number, the lowest possible number is used as the default value. This means:

- Slot 1.
- Channel 1.

Common Commands

The IEEE 488.2 standard has a list of reserved commands, called common commands. Some of these commands must be implemented by any instrument using the standard, others are optional.

Your instrument implements all the necessary commands, and some optional ones. This section describes the implemented commands.

Common Command Summary

Table 3 gives a summary of the common commands.

Table 3 Common Command Summary

Command	Parameter	Function	Page
*CLS		Clear Status Command	page 38
*ESE		Standard Event Status Enable Command	page 38
*ESE?		Standard Event Status Enable Query	page 39
*ESR?		Standard Event Status Register Query	page 39
*IDN?		Identification Query	page 39
*OPC		Operation Complete Command	page 40
*OPC?		Operation Complete Query	page 40
*OPT?		Options Query	page 40
*RST		Reset Command	page 41
*STB?		Read Status Byte Query	page 41
*TST?		Self Test Query	page 42
*WAI		Wait Command	page 43

NOTE

These commands are described in more detail in "IEEE-Common Commands" on page 38.

Common Status Information

There are three registers for the status information. Two of these are status-registers and one is an enable-registers. These registers are described in the IEEE Standard 488.1-2003. You can find further descriptions of these registers under *ESE, *ESR?, and *STB?.

Figure 1 shows how the Standard Event Status Enable Mask (SESEM) and the Standard Event Status Register (SESR) determine the Event Status Bit (ESB) of the Status Byte.

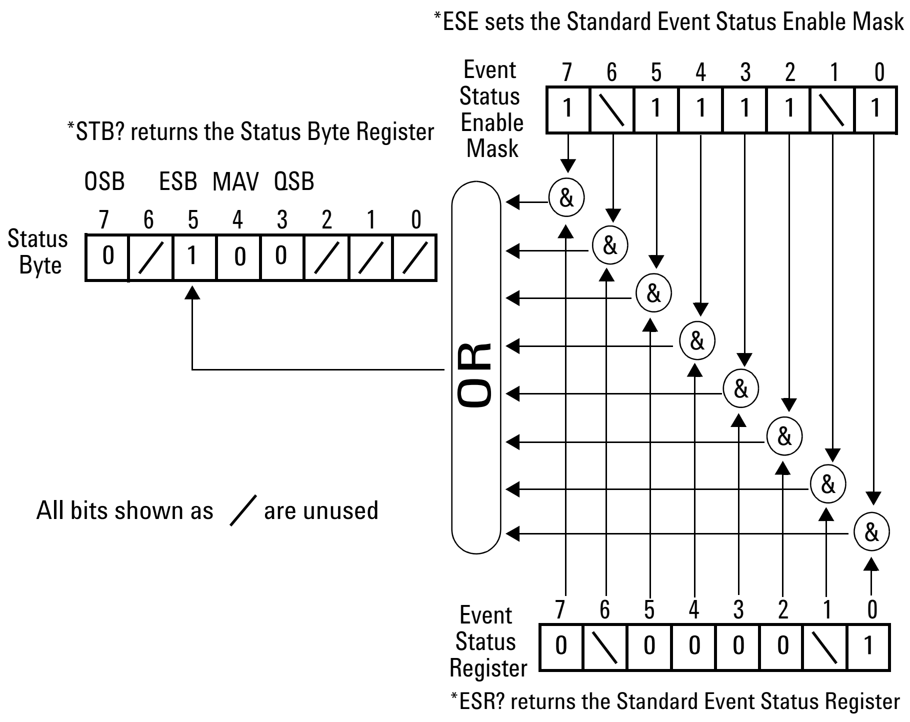


Figure 1 The Event Status Bit

The SESR contains the information about events that are not slot or instrument specific. For details of the function of each bit of the SESR, see “Standard Event Status Register” on page 24.

The SESEM allows you to choose the event that may affect the ESB of the Status Byte. If you set a bit of the SESEM to zero, the corresponding event cannot affect the ESB. The default is for all the bits of the SESEM to be set to 0.

The questionable and operation status systems set the Operational Status Bit (OSB) and the Questionable Status Bit (QSB). These status systems are described in “The Status Model - Multiport Power Meters” on page 44 and “The Status Model - Variable Optical Attenuators and Tunable Laser Sources" on page 57.

NOTE

Unused bits in any of the registers change to 0 when you read them.

Common Status Command Summary

See also “The Status Model - Multiport Power Meters" on page 44, and “The Status Model - Variable Optical Attenuators and Tunable Laser Sources" on page 57.

Annotations

Status Byte Register

- Bit 3, the QSB, is built from the questionable event status register and its enable mask.
- Bit 4, the MAV, is set if the message output queue is not empty.
- Bit 5, the ESB, is built from the SESR and its SESEM.
- Bit 7, the OSB, is built from the operation event status register and its enable mask.
- All other bits are unused, and therefore set to 0.

Standard Event Status Register

- Bit 0 is set if an operation complete event has been received since the last call to *ESR?.
- Bit 1 is always 0 (no service request).
- Bit 2 is set if a query error has been detected.
- Bit 3 is set if a device dependent error has been detected.
- Bit 4 is set if an execution error has been detected.
- Bit 5 is set if a command error has been detected.
- Bit 6 is always 0 (no service request).
- Bit 7 is set for the first call of *ESR? after Power On.

*STB?	returns status byte, value 0 .. +255
*ESE	sets the standard event status enable mask, parameter 0 .. +255
*ESE?	returns SESE, value 0 .. +255
*ESR?	returns the standard event status register, value 0 .. +255
*OPC	parses all program message units in the message queue, and prevents the instrument from executing any further commands until all pending commands are completed.
*OPC?	returns 1 if "operation complete", 0 otherwise.

*CLS	clears the status byte and SESR, and removes any entries from the error queue.
*RST	clears the error queue, loads the default setting, and restarts communication. **NOTE:** *RST does NOT touch the STB or SESR. A running measurement is stopped.
*TST?	initiates an instrument selftest and returns the results as a 32 bit LONG.

Other Common Commands

*OPT?	returns the installed modules and the slots these modules are installed in: For example, *OPT? $\rightarrow$???, , A modules is installed in slot 0. Slot 1 is empty.
*WAI	Included for compatibility.
*IDN?	identifies the instrument; returns the manufacturer, instrument model number, serial number, and firmware revision level.

2
Specific Commands

This chapter lists all the instrument specific commands with a single-line description.

Each of these summaries contains a page reference for more detailed information about the particular command later in this manual.

Specific Command Summary

The commands are ordered in a command tree. Every command belongs to a node in this tree.

The root nodes are also called the subsystems. A subsystem contains all commands belonging to a specific topic. In a subsystem there may be further subnodes.

All the nodes have to be given with a command. For example in the command :init:cont

- INITiate is the subsystem containing all commands and queries concerned with setting up the power meter,
- CONTinuous is the command setting a continuous measurement.

NOTE If a command and a query are both available, the command ends /?. So, :init:cont/? means that init:cont and init:cont? are both available.

Table 1 and Table 2 gives an overview of the command tree. You see the nodes, the subnodes, and the included commands.

Not all commands apply to all instruments. In the instrument column

- *Attenuator* indicates a command that applies to the attenuator channel of N775*x*A/6*x*A variable optical attenuators.
- *Att-PM* indicates a command that applies to the power meter channel of N775*x*A variable optical attenuators.
- *MPPM* indicates a command that applies to the N774*x*A Multiport power meter.
- *PM* indicates a command that applies to the N774*x*A Multiport power meter and the power meter channel of N775*x*A variable optical attenuators.
- *81950A* indicates a command that applies to the LMS Platform compact tunable laser modules.
- *N771XA* indicates a command that applies to the Tunable Laser System Source N7711A, N7714A.
- *N773XA* indicates a command that applies to the Optical Switch N7731A, N7734A, N7736A, N7738A.

Table 1 Specific Command Summary

Command	Description	Instrument	Page
:CONFigure[n][:CHANnel[m]]:OFFSet:WAVelength			
:REFerence	Sets/gets the slot and channel of the external powermeter.	Attenuator	page 112
:STATe	Sets/gets whether the attenuator uses its offset table .	Attenuator	page 111
:TABle?	Queries the complete offset table.	Attenuator	page 113
:TABle:SIZe?	Queries the size of the offset table, or the maximum/minimum size of the table.	Attenuator	page 113
:VALue	Adds or overwrites a value pair (wavelength; offset) to the offset table.	Attenuator	page 111
:VALue:DELete	Deletes an offset value pair (wavelength:offset).	Attenuator	page 114
:VALue:DELete:ALL	Deletes every value pair (wavelength:offset) from the offset table.	Attenuator	page 112
:VALue:OFFSet?	Queries an offset value.	Attenuator	page 113
:VALue:PAIR?	Queries an offset value pair (wavelength:offset).	Attenuator	page 114
:VALue:WAVelength?	Queries a wavelength value from its position, or index, in the offset table.	Attenuator	page 113
:CONFigure:MEASurement:SETTing			
:ACTual?	Get the index of the setting currently being used.	all	page 77
:CANCel	Discard all the changes to the setting since the last save or recall	all	page 78
:ERASe	Erase a setting from memory.	all	page 78
:NUMBer?	Get the number of settings.	all	page 78
:PRESet	Resets the setting values in the working memory	all	page 79
:RECall	Recall a setting from FLASH memory.	all	page 79
:SAVE	Save the current setting to FLASH memory.	all	page 79
:FETCh[n][:CHANnel[m]][:SCALar]			
:POWer[:DC]?	Returns the most recent power value from a sensor.	MPPM Attenuator	page 115
:POWer[:DC]:ALL?	Returns the most recent power values from all the sensors.	MPPM Attenuator	page 115
:POWer[:DC]:ALL:CSV?	Returns a string of the most recent power values from all the sensors in comma separated format.	MPPM Attenuator	page 116
:POWer[:DC]:ALL:CONFig?	Return the slot and channel numbers of available power meter channels.	MPPM Attenuator	page 116
:INITiate[n][:CHANnel[m]]			
[:IMMediate]	Starts a measurement.	PM	page 117
:CONTinuous/?	Starts or Queries a single/continuous measurement.	PM	page 117
:INPut[:CHANnel[m]]			
:ATTenuation:ALL	Sets/gets the current attenuation factor, in dB.	Attenuator	page 100
:WAVelength:ALL	Sets/gets the attenuator operating wavelength.	Attenuator	page 103

Table 1 Specific Command Summary (continued)

Command	Description	Instrument	Page
:INPut[n][:CHANnel[m]]			
:ATTenuation/?	Sets/gets the current attenuation factor, in dB.	Attenuator	page 100
:ATTenuation:SPEed/?	Sets/gets the filter transition speed; in dB/s.	Attenuator	page 102
:OFFSet/?	Sets the offset factor.	Attenuator	page 101
:OFFSet:DISPlay	Sets the offset factor (attenuation = 0).	Attenuator	page 101
:OFFSet:POWermeter	Sets offset to difference between an external power meter and the value measured by the attenuator.	Attenuator	page 101
:WAVelength/?	Sets/gets the attenuator operating wavelength.	Attenuator	page 103
:OUTPut[:CHANnel[m]]			
:POWer:ALL	Sets/gets the output power value (P) for all attenuators.	Attenuator	page 104
[:STATe]:ALL	Sets/gets the state of the shutter for all attenuators.	Attenuator	page 107
:OUTPut[n][:CHANnel[m]]			
:APMode /?	Sets/gets whether the user has amended the power value or the attenuation value.	Attenuator	page 103
:ATIMe /?	Sets/gets the internal power meter's averaging time.	Attenuator	page 108
:CORRection:COLLect:ZERO/?	Zeros the electrical offsets of the attenuator's integrated powermeter. Queries the zero status.	Attenuator Att-PM	page 108
:CORRection:COLLect:ZERO:ALL/?	Zero all available powermeter ports in the instrument. Queries the zeroing status.	Attenuator Att-PM	page 109
:POWer/?	Sets/gets the output power value (P).	Attenuator	page 104
:POWer:CONTRol/?	Sets/gets whether the power control mode is on or off.	Attenuator	page 106
:POWer:OFFSet/?	Sets/gets a power offset (Poffset).	Attenuator	page 105
:POWer:OFFSet:POWermeter	Calculates the power offset by subtracting the power measured by a power meter from the attenuator's measured output power.	Attenuator	page 105
:POWer:Unit/?	Sets/gets whether the power unit used is dBm or Watts.	Attenuator	page 106
[:STATe]/?	Sets/gets the state of the shutter.	Attenuator	page 107
[:STATe]:APOWeron /?	Sets/sets the state of the shutter at power on.	Attenuator	page 107
:READ[n][:CHANnel[m]][:SCALar]			
:POWer[:DC]?	Returns a power value from a sensor.	MPPM Attenuator	page 118
:POWer[:DC]:ALL?	Returns the power values from all the sensors.	MPPM Attenuator	page 117
:POWer[:DC]:ALL:CONFig?	Returns the slot and channel numbers for all the sensors.	MPPM Attenuator	page 118
:ROUTe[n]			
[:CHANnel[m]]/?	Sets or returns the channel route between two ports.	N773xA	page 119
[:CHANnel[m]]:CONFig?	Reads the switch configuration of an instrument.	N773xA	page 119
[:CHANnel[m]]:CONFig:ROUTe?	Reads the allowed switch routes of an instrument.	N773xA	page 120

Table 1 Specific Command Summary (continued)

Command	Description	Instrument	Page
:SENSe[n][:CHANnel[m]]:CORRection			
[:LOSS][:INPut][:MAGNitude]/?	Sets or returns the calibration value for a power meter.	PM	page 120
:COLLECT:ZERO/?	Executes a zero calibration of a sensor. Returns the current zero state of a sensor.	MPPM Attenuator	page 121
:COLLECT:ZERO:ALL/?	Executes a zero calibration of all sensors, or returns the status of the most recent zero calibration of all sensors.	MPPM Attenuator	page 122
:SENSe[n][:CHANnel[m]]:FUNCtion			
:LOOP/?	Sets or returns the number of logging loops.	MPPM	page 129
:PARameter:LOGGing/?	Sets or returns the number of samples and the averaging time, t_{avg}, for logging.	MPPM	page 122
:PARameter:MINMax/?	Sets or returns the minmax mode and the window size.	PM	page 124
:PARameter:STABility/?	Sets or returns the total time, delay time and the averaging time, t_{avg}, for stability.	MPPM	page 125
:RESult?	Returns the data array of the last function.	PM	page 130
:RESult:BUFA?	Returns the data array of the last data acquisition function in Buffer A.	MPPM	page 130
:RESult:BUFB?	Returns the data array of the last data acquisition function in Buffer B.	MPPM	page 131
:RESult:INDex?	Returns the number of logging loops that have already finished.	MPPM	page 129
:STATe/?	Enables/disables the function mode or returns whether the function mode is enabled.	PM	page 131
:SENSe[n][:CHANnel[m]]:POWer			
:ATIMe/?	Sets or returns the average time of a sensor.	PM	page 132
:GAIN:AUTO/?	Sets or returns the auto gain setting of a sensor, which optimizes the dynamic or transient response.	PM	page 134
:RANGe:AUTO/?	Sets or returns the range of a sensor to produce the most dynamic range without overloading.	PM	page 133
:RANGe[:UPPer]/?	Sets or returns the most positive signal entry expected for a sensor.	PM	page 133
:REFerence/?	Sets or returns the reference level of a sensor.	PM	page 135
:REFerence:DISPlay	Sets the reference level for a sensor from the input power level.	PM	page 135
:REFerence:STATe/?	Sets or returns whether sensor results are in relative or absolute units.	PM	page 135
:REFerence:STATe:RATio/?	Sets or returns whether sensor results are displayed relative to a slot or to an absolute reference.	PM	page 136
:UNIT/?	Sets or returns the units used for absolute readings on a sensor.	PM	page 137
:UNIT:ALL:CSV?	Returns the units from all the ports of the instrument.	all	page 137
:WAVelength/?	Sets or returns the wavelength for a sensor.	PM	page 137

Table 1 Specific Command Summary (continued)

Command	Description	Instrument	Page
:SLOT[n][:HEAD]			
:WAVelength:RESPonse?	Returns the wavelength response from port n, in binary data format.	PM, Attenuator	page 83
:WAVelength:RESPonse:CSV?	Returns the wavelength response from port n, in .csv data format.	PM, Attenuator	page 83
:WAVelength:RESPonse:SIZE?	Returns the number of elements in the wavelength response table.	PM, Attenuator	page 83
:STATus:OPERation			
[:EVENt]?	Returns the Operational Status Event Summary Register (OESR).	MPPM Attenuator	page 48
:CONDition?	Returns the Operational Status Condition Summary Register.	MPPM Attenuator	page 49
:ENABle/?	Sets or queries the Operational Status Enable Summary Mask.	MPPM Attenuator	page 49
:NTRansition/?	Sets or queries the Operational negative transition filter.	Attenuator	page 60
:PTRansition/?	Sets or queries the Operational positive transition filter.	Attenuator	page 61
:STATus*n*:OPERation			
[:EVENt]?	Returns the Operational Slot Status Event Register for slot *n*.	MPPM Attenuator	page 51
:CONDition?	Returns the Operational Slot Status Condition Register for slot *n*.	MPPM Attenuator	page 51
:ENABle/?	Sets or queries the Operation Slot Status Enable Mask for slot *n*.	MPPM Attenuator	page 52
:STATus:PRESet	Restore the status configuration to factory defaults.	MPPM Attenuator	page 48
:STATus:QUEStionable			
[:EVENt]?	Returns the Questionable Status Event Summary Register.	MPPM Attenuator	page 52
:CONDition?	Returns the Questionable Status Condition Summary Register.	MPPM Attenuator	page 55
:ENABle/?	Sets or queries the Questionable Status Enable Summary Mask.	MPPM Attenuator	page 55
:NTRansition/?	Sets or queries the Questionable negative transition filter.	Attenuator	page 64
:PTRansition/?	Sets or queries the Questionable positive transition filter.	Attenuator	page 65
:STATus*n*:QUEStionable			
[:EVENt]?	Returns the Questionable Slot Status Event Register for slot *n*.	MPPM Attenuator	page 54
:CONDition?	Returns the Questionable Slot Status Condition Register for slot *n*.	MPPM Attenuator	page 55
:ENABle/?	Sets or queries the Questionable Slot Status Enable Mask for slot *n*.	MPPM Attenuator	page 55

Table 1 Specific Command Summary (continued)

Command	Description	Instrument	Page
:SYSTem			
:DATE/?	The instruments have no internal clock. This command is included for backwards compatibility.	all	page 67
:ERRor?	Returns the contents of the instrument's error queue.	all	page 67
:HELP:HEADers?	Returns a list of GPIB commands.	all	page 67
:LXI:IDN	Starts or stops the LAN LED. This can be used to identify the unit.	all	page 68
:PRESet	Sets all parameters to their default values.	all	page 68
:REBoot	Reboots the instrument	all	page 68
:TIME/?	The instruments have no internal clock. This command is included for backwards compatibility.	all	page 69
:VERSion?	Returns the instrument's SCPI version.	all	page 69
:SYSTem:COMMunicate:ETHernet			
:AUTOip	Enable/Check whether auto IP is enabled or disabled.	Attenuator	page 71
:CANCel	Undo changes to the network parameters.	all	page 76
:DGATeway/?	Set/Get the default gateway.	all	page 75
:DGATeway:CURRent?	Get the currently used default gateway.	all	page 73
:DHCP:ENABle/?	Enable/Check whether DHCP is enabled or disabled.	all	page 71
:DOMainname/?	Set/Get the domain name	all	page 74
:DOMainname:CURRent?	Get the currently used domain name	all	page 72
:HOSTname/?	Set/Get the host name	all	page 73
:HOSTname:CURRent?	Get the current host name.	all	page 72
:IPADdress/?	Set/Get the manually set IP address of the system.	all	page 74
:IPADdress:CURRent?	Get the current IP address of the system.	all	page 72
:MACaddress?	Get the MAC address of the network adapter.	all	page 70
:RESet	Resets all LAN parameters to the factory default.	all	page 75
:RESTart	Restart the system's network interface.	all	page 76
:SAVE	Save the system's network interface parameters.	all	page 76
:SMASk/?	Set/Get the subnet mask.	all	page 74
:SMASk:CURRent?	Get the currently used subnet mask.	all	page 72
:SYSTem:COMMunicate:GPIB			
[:SELF]:ADDRess/?	Sets or returns the GPIB address.	all	page 69
:TRIGger:CONFiguration/?	Sets or returns trigger configuration.	all	page 141
:TRIGger[n][CHANnel[m]]			
:INPut/?	Sets or returns the incoming trigger response .	MPPM Attenuator	page 140
:OUTPut/?	Sets or returns the outgoing trigger response.	MPPM Attenuator	page 141

Table 2 describes the SCPI Commands for use of 81950A compact tunable laser modules and N7711A, N7714A single and 4-port Tunable Laser System Sources.

Table 2 Specific Command Summary for Tunable Laser Modules

Command	Description	Instrument	Page
:OUTPut[n][:CHANnel[m]			
:POWer:UNit/?	Sets or returns the power unit used (dBm or W).	81950A, N7711A, N7714A	page 85
[:STATe]/?	Sets a source's output state or returns its current state.	81950A, N7711A, N7714A	page 85
[:SOURce[n]][:CHANnel[m]]:POWer			
[:LEVel][:IMMediate][:AMPLitude]/?	Sets or returns the laser output power.	81950A, N7711A, N7714A	page 86
:STATe/?	Sets or returns the state of the source output signal.	81950A, N7711A, N7714A	page 87
:UNIT/?	Sets or returns the power unit.	81950A, N7711A, N7714A	page 87
[:SOURce[n]][:CHANnel[m]]:WAVelength			
[:CW|:FIXED]/?	Sets or returns the wavelength of a source.	81950A, N7711A, N7714A	page 88
:TOGRid	Sets the wavelength to the nearest grid point, only changing the channel number.	81950A, N7711A, N7714A	page 89
:AUTO/?	Sets auto mode on or off or returns its current state.	81950A, N7711A, N7714A	page 89

Table 2 Specific Command Summary for Tunable Laser Modules (continued)

Command	Description	Instrument	Page
[:SOURce[n]][:CHANnel[m]]:FREQuency			
/?	Sets or returns the frequency.	81950A, N7711A, N7714A	page 90
:REFerence/?	Sets or returns the reference frequency.	81950A, N7711A, N7714A	page 91
:GRID/?	Sets or returns the grid spacing.	81950A, N7711A, N7714A	page 91
:CHANnel/?	Sets or returns the channel number.	81950A, N7711A, N7714A	page 92
:OFFSet/?	Sets or returns the frequency offset (aka fine tune frequency).	81950A, N7711A, N7714A	page 93
:TOGRid	Sets the frequency to the nearest grid point, only changing the channel number.	81950A, N7711A, N7714A	page 94
:AUTO/?	Sets auto mode on or off or returns its current state.	81950A, N7711A, N7714A	page 94
[:SOURce[n]][:CHANnel[m]]:MODUlation			
:INTernal[:STATe]/?	Enable or disable the SBS suppression or return its current state.		page 95
:INTernal:SBSControl[:LEVel]/?	Sets or returns the SBS suppression frequency.		page 95
:INTernal:RATE?	Internal modulation rate.	N7711A, N7714A	page 96
:INTernal:WAVeform?	Internal modulation waveform.	N7711A, N7714A	page 96
:EXTernal[:STATe]/?	Sets or returns the state of the amplitude modulation.	81950A	page 97
:EXTernal:AM[:LEVel]/?	Sets or returns signaling or trace tone AM level.	81950A	page 97

3 Instrument Setup and Status

This chapter gives descriptions of commands that you can use when setting up your instrument. The commands are split into the following separate subsytems:

- IEEE specific commands that were introduced in "Common Commands" on page 22.
- STATus subsystem commands that relate to the status model.
- SYSTem subsystem commands that control the interfaces and internal data.

IEEE-Common Commands

"Common Commands" on page 22 gave a brief introduction to the IEEE-common commands which can be used with the instruments. This section gives fuller descriptions of each of these commands.

command:	***CLS**
syntax:	*CLS
description:	The CLear Status command *CLS clears the following: Error queue Standard event status register (SESR) Status byte register (STB) After the *CLS command the instrument is left waiting for the next command. The instrument setting is unaltered by the command, although *OPC/*OPC? actions are cancelled.
parameters:	none
response:	none
example:	*CLS

command:	***ESE**		
syntax:	*ESE<wsp><value> 0 ≤ *value* ≤ 255		
description:	The standard Event Status Enable command (*ESE) sets bits in the Standard Event Status Enable Mask (SESEM) that enable the corresponding bits in the standard event status register (SESR). The register is cleared: at power-on, by sending a value of zero. The register is not changed by the *RST and *CLS commands.		
parameters:	The bit value for the register (a *16-bit signed integer* value):		
	Bit	**Mnemonic**	**Decimal Value**
	7 (MSB)	Power On	128
	6	Not Used	0
	5	Command Error	32
	4	Execution Error	16
	3	Device Dependent Error	8
	2	Query Error	4
	1	Not Used	0
	0 (LSB)	Operation Complete	1

response:	none
example:	*ESE 21

command:	***ESE?**
syntax:	*ESE?
description:	The standard Event Status Enable query *ESE? returns the contents of the Standard Event Status Enable Mask (see *ESE for information on this register).
parameters:	none
response:	The bit value for the register (a *16-bit signed integer* value).
example:	*ESE? ⟶ 21<END>

command:	***ESR?**		
syntax:	*ESR?		
description:	The standard Event Status Register query *ESR? returns the contents of the Standard Event Status Register. The register is cleared after being read.		
parameters	none		
response	The bit value for the register (a *16-bit signed integer* value):		
	Bit	**Mnemonic**	**Decimal Value**
	7 (MSB)	Power On	128
	6	Not used	0
	5	Command Error	32
	4	Execution Error	16
	3	Device Dependent Error	8
	2	Query Error	4
	1	Not used	0
	0 (LSB)	Operation Complete	1
example:	*ESR? ⟶ 21<END>		

command:	***IDN?**
syntax:	*IDN?
description:	The IDeNtification query *IDN? gets the instrument identification over the interface.
parameters:	none

response:	The identification terminated by <END>: *For example.*	
	Agilent Technologies	manufacturer
	mmmm	instrument model number (for example N7744A)
	ssssssss	serial number
	rrrrrrrrr	firmware revision level
example:	*IDN? ⟶ Agilent Techologies,mmmm,ssssssss,rrrrrrrrr<END>	

command:	***OPC**
syntax:	*OPC
description:	The instrument parses and executes all program message units in the input queue and sets the operation complete bit in the standard event status register (SESR). This command can be used to avoid filling the input queue before the previous commands have finished executing. The following actions cancel the *OPC command (and put the instrument into Operation Complete, Command Idle State): Power-on the Device Clear Active State is asserted on the interface. *CLS *RST
parameters:	none
response:	none
example:	*OPC

command:	***OPC?**
syntax:	*OPC?
description:	The Operation Complete query *OPC? returns 1 if all operations are completed (0 otherwise). See also SLOT[n]:OPC?
parameters:	none
response:	0 or 1
example:	*OPC? ⟶ 1<END>

command:	***OPT?**
syntax:	*OPT?
description:	The OPTions query *OPT? returns the options installed in your instrument.
parameters:	none

response:	Returns the part number of all installed options, separated by commas. Slots are listed starting with the lowest slot number, that is, slot 1. If any slot is empty, two spaces are inserted instead of the option's part number. See the example below, for an N7752 Optical Attenuator and Power Meter. For compatibility with the 816x platform, every attenuator is followed by an empty slot..
example:	*OPT? → N7752A-002, ,N7752A-002, ,N7752A-001,N7752A-001, , <END>

command:	***RST**
syntax:	*RST
description:	The ReSeT command *RST sets the instrument to the reset setting (standard setting) stored internally. NOTE: the stored settings are deleted. By contrast the `:CONFigure:MEASurement:SETTing:PRESet` command keeps the previously stored settings in nonvolatile RAM and they can be recalled again. Pending *OPC? actions are cancelled. The instrument is placed in the idle state awaiting a command. The *RST command clears the error queue. The *RST command is equivalent to the *CLS command AND the syst:preset command. The following are not changed: GPIB (interface) state Instrument interface address Output queue Service request enable register (SRE) Standard Event Status Enable Mask (SESEM)
parameters:	none
response:	none
example:	*RST

command:	***STB?**
syntax:	*STB?
description:	The STatus Byte query *STB? returns the contents of the Status Byte register.
parameters:	none

response: The bit value for the register (a *16-bit signed integer* value):

Bit	**Mnemonic**	**Decimal Value**
7 (MSB)	Operation Status (OSB)	128
6	Not used	0
5	Event Status Bit (ESB)	32
4	Message Available (MAV)	16
3	Questionable Status (QSB)	8
2	Error/Event Queue	4
1	Not used	0
0	Not used	0

example: *STB? ⟶ 128<END>

command: ***TST?**

syntax: *TST?

description: The self-TeST query *TST? makes the instrument perform a self-test and place the results of the test in the output queue. If the self-test fails, the results are also put in the error queue. We recommend that you read self-test results from the error queue. No further commands are allowed while the test is running. After the self-test the instrument is returned to the setting that was active at the time the self-test query was processed. The self-test does not require operator interaction beyond sending the *TST? query.

parameters: none

response: The sum of the results for the individual tests (a *32-bit signed integer* value, where 0 ≤ *value* ≤ 4294967296):

Bits	**Mnemonic**	**Decimal Value**
31	Selftest failed on Mainframe	A negative value
9-30	Not used	0
8	Selftest failed on Slot 8	256
7	Selftest failed on Slot 7	128
6	Selftest failed on Slot 6	64
5	Selftest failed on Slot 5	32
4	Selftest failed on Slot 4	16
3	Selftest failed on Slot 3	8
2	Selftest failed on Slot 2	4
1	Selftest failed on Slot 1	2

If 16 is returned, slot 4 has failed.
If 18 is returned, slots 1 and 4 have failed.
A value of zero indicates no errors.

example: *TST? ⟶ 0<END>

command:	***WAI**
syntax:	*WAI
description:	Included for compatibility.
parameters:	none
response:	none
example:	*WAI

Status Reporting – The STATus Subsystem

The Status subsystem allows you to return and set details from the Status Model.

The Status Model - Multiport Power Meters

Status Registers

Each node of the status circuitry has three registers:

- A condition register (CONDition), which contains the current status. This register is updated continuously. It is not changed by having its contents read.
- The event register (EVENt), which contains details of any positive transitions in the corresponding condition register, that is, when a bit changes from 0 → 1. The contents of this register are cleared when it is read. The contents of any higher-level registers are affected with regard to the appropriate bit.
- The enable register (ENABle), which enables changes in the event register to affect the next stage of registers.

NOTE The event register is the only kind of register that can affect the next stage of registers.

The structures of the Operational and Questionable Status Systems are similar. Figure 2 describe how the Questionable Status Bit (QSB) and the Operational Status Bit (OSB) of the Status Byte Register are determined.

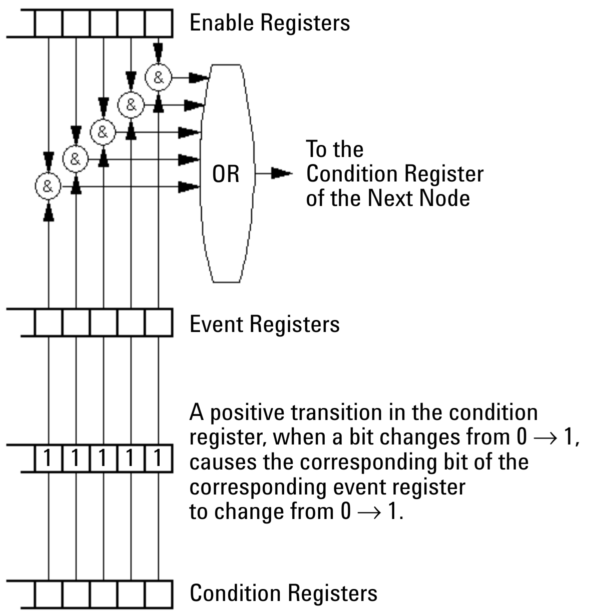

Enable Registers
&
OR
To the
Condition Register
of the Next Node
Event Registers
1 1 1 1 1
A positive transition in the condition register, when a bit changes from 0 → 1, causes the corresponding bit of the corresponding event register to change from 0 → 1.
Condition Registers

Figure 1 The Registers and Filters for a Node of the Agilent N7744A / N7745A Multiport Optical Power Meter

The Operational/Questionable Slot Status Event Register (OSSER/QSSER) contains the status of a particular slot. A bit changes from 0 → 1 when an event occurs, for example, when a laser is switched on. For details of the function of each bit of these registers, see “Operation/Questionable Status Summary Register” on page 47.

The Operational/Questionable Slot Enable Status Mask (OSESM/QSESM) allows you to choose the events for each slot that may affect the Operational/Questionable Status Event Register (see below). If you set a bit of the OSESM/QSESM to zero, the occurence of the corresponding event for this particular slot cannot affect the Operational/Questionable Status Event Register. The default is for all the bits of the OSESM/QSESM to be set to 0.

The Operational/Questionable Status Event Summary Register (OSESR/QSESR) summarizes the status of every slot of your instrument. If, for any slot, any bit of the QSSER goes from 0 → 1 AND the corresponding bit of the QSSEM is 1 at the same time, the QSESR bit representing that slot is set to 1.

The Operational/Questionable Status Enable Summary Mask (OSESM/QSESM) allows you to choose the slots that may affect the OSB/QSB of the Status Byte. If any bit of the QSESR goes from 0 → 1 AND the corresponding bit of the

QSESM is 1 at the same time, the QSB of the Status Byte is set to 1. If you set a bit of the OSESM/QSESM to zero, the corresponding slot cannot affect the OSB/QSB. The default is for all the bits of the OSESM/QSESM to be set to 0.

The Operational/Questionable Status Enable Summary Mask for the Agilent N7744A / N7745A Multiport Optical Power Meter consists of one level. These are described in "Status System" on page 46.

Status System

The status system for the Agilent N7744A / N7745A Multiport Optical Power Meter returns the status of 4 and 8 slots respectively. The Operational/Questionable Status Summary Registers consist of one level and are described by Figure 2. Any commands that require LEVel*1* do not apply to these instruments.

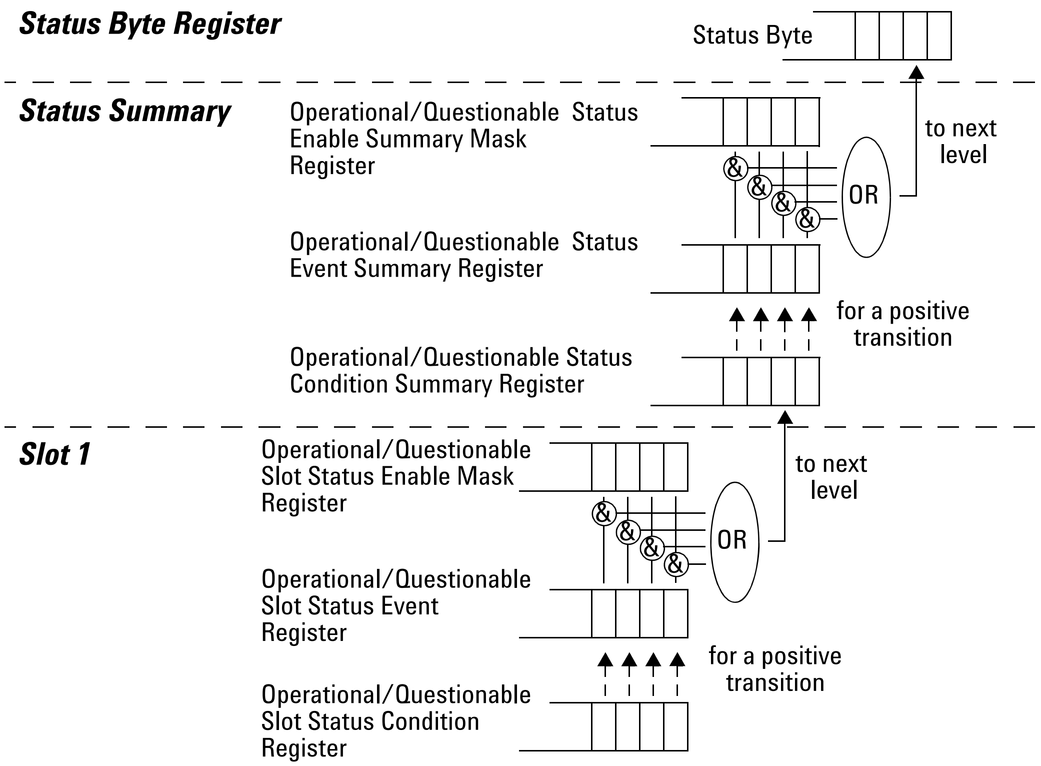

Status Byte Register
Status Byte
Status Summary
Operational/Questionable Status Enable Summary Mask Register
to next level
OR
Operational/Questionable Status Event Summary Register
for a positive transition
Operational/Questionable Status Condition Summary Register
Slot 1
Operational/Questionable Slot Status Enable Mask Register
to next level
OR
Operational/Questionable Slot Status Event Register
for a positive transition
Operational/Questionable Slot Status Condition Register

Figure 2 The Operational/Questionable Status System for the Agilent N7744A / N7745A Multiport Optical Power Meter

Annotations

Operation/Questionable Status Summary

- The Operation/Questionable Status Summary consist of a condition and an event register.
- A "rising" bit in the condition register is copied to the event register.
- A "falling" bit in the condition register has no effect on the event register.
- Reading the condition register is non-destructive.
- Reading the event register is destructive.
- A summary of the event register and its enable mask is set in the status byte.

Operation/Questionable Status Summary Register

- Bits 0 to 4 are built from the OSSER/QSSER and the OSSEM/QSSEM.
- A summary of the event register, the condition register and the enable mask is set in the status byte.

Operation/Questionable Slot Status

- The Operation/Questionable Slot Status consist of a condition and an event register.
- A "rising" bit in the condition register is copied to the event register.
- A "falling" bit in the condition register has no effect on the event register.
- Reading the condition register is non-destructive.
- Reading the event register is destructive.
- A summary of the event register, the condition register and the enable mask is set in the status byte.

Operation Slot Status Register

- Bit 3 is set if Power Meter zeroing.
- All other bits are unused, and therefore set to 0.

Questionable Slot Status Register

- Bit 0 is set if excessive averaging time is set for any Power Meter.
- Bit 1 is set if the last Power Meter zeroing failed.

- Bit 2 is set if temperature is out of range.
- Bit 5 is set if the slot is out of specifications.
- All other bits are unused, and therefore set to 0.

command:	**:STATus:PRESet**
syntax:	:STATus:PRESet
affects:	N774xA multiport power meters. See “The Status Model - Variable Optical Attenuators and Tunable Laser Sources" on page 57 for information on the equivalent command for the N775xA and N776xA attenuators.
description:	Presets all bits in all the enable masks for both the OPERation and QUEStionable status systems to 0, that is, OSSEM, QSSEM, OSESM, and QSESM.
parameters:	none
response:	none
example:	stat:pres

command:	**:STATus:OPERation[:EVENt][:LEVel]?**
syntax:	:STATus:OPERation[:EVENt][:LEVel]?
affects:	N774xA multiport power meters. See “The Status Model - Variable Optical Attenuators and Tunable Laser Sources" on page 57 for information on the equivalent query for the N775xA and N776xA attenuators.
description:	Returns the Operational Status Event Summary Register (OSESR).
parameters:	none

response:	The sum of the results for the slots (a *16-bit signed integer* value, where 0 ≤ *value* ≤ 32767):		
	Bits	**Mnemonics**	**Decimal Value**
	15	Not used	0
	14	Not used	16384
	13	Not used	8192
	12	Not used	4096
	11	Not used	2048
	10	Not used	1024
	9	Not used	512
	8	Slot 8 Summary	256
	7	Slot 7 Summary	128
	6	Slot 6 Summary	64
	5	Slot 5 Summary	32
	4	Slot 4 Summary	16
	3	Slot 3 Summary	8
	2	Slot 2 Summary	4
	1	Slot 1 Summary	2
	0	Not used	1
example:	stat:oper? ⟶ +0<END>		

command:	**:STATus:OPERation:CONDition[:LEVel]?**
syntax:	:STATus:OPERation:CONDition[:LEVel]?
affects:	N774xA multiport power meters. See "The Status Model - Variable Optical Attenuators and Tunable Laser Sources" on page 57 for information on the equivalent query for the N775xA and N776xA attenuators.
description:	Reads the Operational Status Condition Summary Register.
parameters:	none

response:	The sum of the results for the individual slots (a *16-bit signed integer* value, where 0 ≤ *value* ≤ 32767):		
	Bits	**Mnemonics**	**Decimal Value**
	15	Not used	0
	14	Not used	16384
	13	Not used	8192
	12	Not used	4096
	11	Not used	2048
	10	Not used	1024
	9	Not used	512
	8	Slot 8 Summary	256
	7	Slot 7 Summary	128
	6	Slot 6 Summary	64
	5	Slot 5 Summary	32
	4	Slot 4 Summary	16
	3	Slot 3 Summary	8
	2	Slot 2 Summary	4
	1	Slot 1 Summary	2
	0	Not used	1
example:	stat:oper:cond? ⟶ +0<END>		

command:	**:STATus:OPERation:ENABle[:LEVel]**
syntax:	:STATus:OPERation:ENABle[:LEVel]<wsp><value>
affects:	N774xA multiport power meters. See "The Status Model - Variable Optical Attenuators and Tunable Laser Sources" on page 57 for information on the equivalent command for the N775xA and N776xA attenuators.
description:	Sets the bits in the Operational Status Enable Summary Mask (OSESM) that enable the contents of the OSESR to affect the Status Byte (STB). Setting a bit in this register to 1 enables the corresponding bit in the OSESR to affect bit 7 of the Status Byte.
parameters:	The bit value for the OSESM as a *16-bit signed integer* value (0 .. +32767) The default value is 0.
response:	none
example:	stat:oper:enab 128

command:	**:STATus:OPERation:ENABle[:LEVel]?**
syntax:	:STATus:OPERation:ENABle[:LEVel]?
affects:	N774xA multiport power meters. See "The Status Model - Variable Optical Attenuators and Tunable Laser Sources" on page 57 for information on the equivalent query for the N775xA and N776xA attenuators.

description:	Returns the OSESM for the OSESR
parameters:	none
response:	The bit value for the operation enable mask as a *16-bit signed integer* value (0 .. +32767)
example:	stat:oper:enab? ⟶ +128<END>

command:	**:STATus*n*:OPERation[:EVENt]?**
syntax:	:STATus*n*:OPERation[:EVENt]?
affects:	N774xA multiport power meters. See "The Status Model - Variable Optical Attenuators and Tunable Laser Sources" on page 57 for information on the equivalent query for the N775xA and N776xA attenuators.
description:	Returns the Operational Slot Status Event Register (OSSER) of slot *n*.
parameters:	none
response:	The results for the individual slot events (a *16-bit signed integer* value, where 0 ≤ *value* ≤ 32767):
example:	stat1:oper? ⟶ +0<END>

Bit	Mnemonic	Decimal Value
8-15	Not used	0
7	Not used	128
6	Not used	64
5	Not used	32
4	Not used	16
3	Slot *n*: Zeroing ongoing	8
2	Not used	0
1	Not used	2
0	Not used	1

command:	**:STATus*n*:OPERation:CONDition?**
syntax:	:STATus*n*:OPERation:CONDition?
affects:	N774xA multiport power meters. See "The Status Model - Variable Optical Attenuators and Tunable Laser Sources" on page 57 for information on the equivalent query for the N775xA and N776xA attenuators.
description:	Returns the Operational Slot Status Condition Register of slot *n*.
parameters:	none

response:	The results for the individual slot events (a *16-bit signed integer* value, where 0 ≤ *value* ≤ 32767): **Bit** / **Mnemonic** / **Decimal Value** 8-15 / Not used / 0 7 / Not used / 128 6 / Not used / 64 5 / Not used / 32 4 / Not used / 16 3 / Slot *n*: Zeroing ongoing / 8 2 / Not used / 0 1 / Not used / 2 0 / Not used / 1
example:	stat1:oper:cond? ⟶ +0<END>

command:	**:STATus*n*:OPERation:ENABle**
syntax:	:STATus*n*:OPERation:ENABle<wsp><value>
affects:	N774xA multiport power meters. See "The Status Model - Variable Optical Attenuators and Tunable Laser Sources" on page 57 for information on the equivalent command for the N775xA and N776xA attenuators.
description:	Sets the bits in the Operation Slot Status Enable Mask (OSSEM) for slot *n* that enable the contents of the Operation Slot Status Event Register (OSSER) for slot *n* to affect the OSESR. Setting a bit in this register to 1 enables the corresponding bit in the OSSER for slot *n* to affect bit *n* of the OSESR.
parameters:	The bit value for the OSSEM as a *16-bit signed integer* value (0 .. +32767)
response:	none
example:	stat:oper:enab 128

command:	**:STATus*n*:OPERation:ENABle?**
syntax:	:STATus*n*:OPERation:ENABle?
affects:	N774xA multiport power meters. See "The Status Model - Variable Optical Attenuators and Tunable Laser Sources" on page 57 for information on the equivalent query for the N775xA and N776xA attenuators.
description:	Returns the OSSEM of slot *n*
parameters:	none
response:	The bit value for the OSSEM as a *16-bit signed integer* value (0 .. +32767)
example:	stat:oper:enab? ⟶ +128<END>

command:	**:STATus:QUEStionable[:EVENt][:LEVel]?**
syntax:	:STATus:QUEStionable[:EVENt][:LEVel]?
affects:	N774xA multiport power meters. See "The Status Model - Variable Optical Attenuators and Tunable Laser Sources" on page 57 for information on the equivalent query for the N775xA and N776xA attenuators.

description:	Returns the Questionable Status Event Summary Register (QSESR).		
parameters:	none		
response:	The sum of the results for the QSESR as a *16-bit signed integer* value (0 .. +32767)		
	Bits	**Mnemonics**	**Decimal Value**
	15	Not used	0
	14	Not used	16384
	13	Not used	8192
	12	Not used	4096
	11	Not used	2048
	10	Not used	1024
	9	Not used	512
	8	Slot 8 Summary	256
	7	Slot 7 Summary	128
	6	Slot 6 Summary	64
	5	Slot 5 Summary	32
	4	Slot 4 Summary	16
	3	Slot 3 Summary	8
	2	Slot 2 Summary	4
	1	Slot 1 Summary	2
	0	Not used	1
example:	stat:ques? ⟶ +0<END>		

command:	**:STATus:QUEStionable:CONDition[:LEVel]?**		
syntax:	:STATus:QUEStionable:CONDition[:LEVel]?		
affects:	N774xA multiport power meters. See “The Status Model - Variable Optical Attenuators and Tunable Laser Sources" on page 57 for information on the equivalent query for the N775xA and N776xA attenuators.		
description:	Returns the Questionable Status Condition Summary Register.		
parameters:	none		
response:	The sum of the results for the Questionable Status Condition Summary Register as a *16-bit signed integer* value (0 .. +32767)		
	Bits	**Mnemonics**	**Decimal Value**
	15	Not used	0
	14	Not used	16384
	13	Not used	8192
	12	Not used	4096
	11	Not used	2048
	10	Not used	1024
	9	Not used	512

	8	Slot 8 Summary	256
	7	Slot 7 Summary	128
	6	Slot 6 Summary	64
	5	Slot 5 Summary	32
	4	Slot 4 Summary	16
	3	Slot 3 Summary	8
	2	Slot 2 Summary	4
	1	Slot 1 Summary	2
	0	Not used	1
example:	stat:ques:cond? ⟶ +0<END>		

command:	**:STATus:QUEStionable:ENABle[:LEVel]**
syntax:	:STATus:QUEStionable:ENABle[:LEVel*0*]<wsp><value>
affects:	N774xA multiport power meters. See "The Status Model - Variable Optical Attenuators and Tunable Laser Sources" on page 57 for information on the equivalent command for the N775xA and N776xA attenuators.
description:	Sets the bits in the Questionable Status Enable Summary Mask (QSESM) that enable the contents of the QSESR to affect the Status Byte (STB). Setting a bit in this register to 1 enables the corresponding bit in the QSESR to affect bit 3 of the Status Byte.
parameters:	The bit value for the questionable enable mask as a *16-bit signed integer* value (0 .. +32767) The default value is 0.
response:	none
example:	stat:ques:enab 128

command:	**:STATus:QUEStionable:ENABle[:LEVel]?**
syntax:	:STATus:QUEStionable:ENABle[:LEVel*0*]?
affects:	N774xA multiport power meters. See "The Status Model - Variable Optical Attenuators and Tunable Laser Sources" on page 57 for information on the equivalent query for the N775xA and N776xA attenuators.
description:	Returns the QSESM for the event register
parameters:	none
response:	The bit value for the QSEM as a *16-bit signed integer* value (0 .. +32767)
example:	stat:ques:enab? ⟶ +128<END>

command:	**:STATus*n*:QUEStionable[:EVENt]?**
syntax:	:STATus*n*:QUEStionable[:EVENt]?

affects:	N774xA multiport power meters. See “The Status Model - Variable Optical Attenuators and Tunable Laser Sources" on page 57 for information on the equivalent query for the N775xA and N776xA attenuators.
description:	Returns the Questionable Slot Status Event Register (OSSER) of slot *n*.
parameters:	none
response:	The results for the individual slot events (a *16-bit signed integer* value, where 0 ≤ *value* ≤ 32767): **Bit** / **Mnemonic** / **Decimal Value** 2 - 15 / Not Used / 1 / Slot *n*: Zeroing failed / 2 0 / Not Used / 1
example:	stat1:ques? ⟶ +0<END>

command:	**:STATus*n*:QUEStionable:CONDition?**
syntax:	:STATus*n*:QUEStionable:CONDition?
affects:	N774xA multiport power meters. See “The Status Model - Variable Optical Attenuators and Tunable Laser Sources" on page 57 for information on the equivalent query for the N775xA and N776xA attenuators.
description:	Returns the Questionable Slot Status Condition Register for slot *n*.
parameters:	none
response:	The results for the individual slot events (a *16-bit signed integer* value, where 0 ≤ *value* ≤ 32767): **Bit** / **Mnemonic** / **Decimal Value** 2 - 15 / Not Used / 1 / Slot *n*: Zeroing failed / 2 0 / Not Used / 1
example:	stat1:ques:cond? ⟶ +0<END>

command:	**:STATus*n*:QUEStionable:ENABle**
syntax:	:STATus*n*:QUEStionable:ENABle<wsp><value>
affects:	N774xA multiport power meters. See “The Status Model - Variable Optical Attenuators and Tunable Laser Sources" on page 57 for information on the equivalent command for the N775xA and N776xA attenuators.
description:	Sets the bits in the Questionable Slot Status Enable Mask (QSSEM) for slot *n* that enable the contents of the Questionable Slot Status Register (QSSR) for slot *n* to affect the QSESR. Setting a bit in this register to 1 enables the corresponding bit in the QSSER for slot *n* to affect bit *n* of the QSESR.
parameters:	The bit value for the QSSEM as a *16-bit signed integer* value (0 .. +32767)
response:	none
example:	stat:ques:enab 128

command:	**:STATus*n*:QUEStionable:ENABle?**
syntax:	:STATus*n*:QUEStionable:ENABle?
affects:	N774xA multiport power meters. See "The Status Model - Variable Optical Attenuators and Tunable Laser Sources" on page 57 for information on the equivalent query for the N775xA and N776xA attenuators.
description:	Returns the QSSEM for slot *n*
parameters:	none
response:	The bit value for the QSSEM as a *16-bit signed integer* value (0 .. +32767)
example:	stat:ques:enab? ⟶ +128<END>

The Status Model - Variable Optical Attenuators and Tunable Laser Sources

Status Model

- Condition registers show the actual state.
- Event registers keep there values until read. Reading them clears their contents.
- Bits in the registers can report a single state/event or a summary.
- Bit 15 is never used.

NOTE The status bit can be read with “*STB?”

The Status Registers

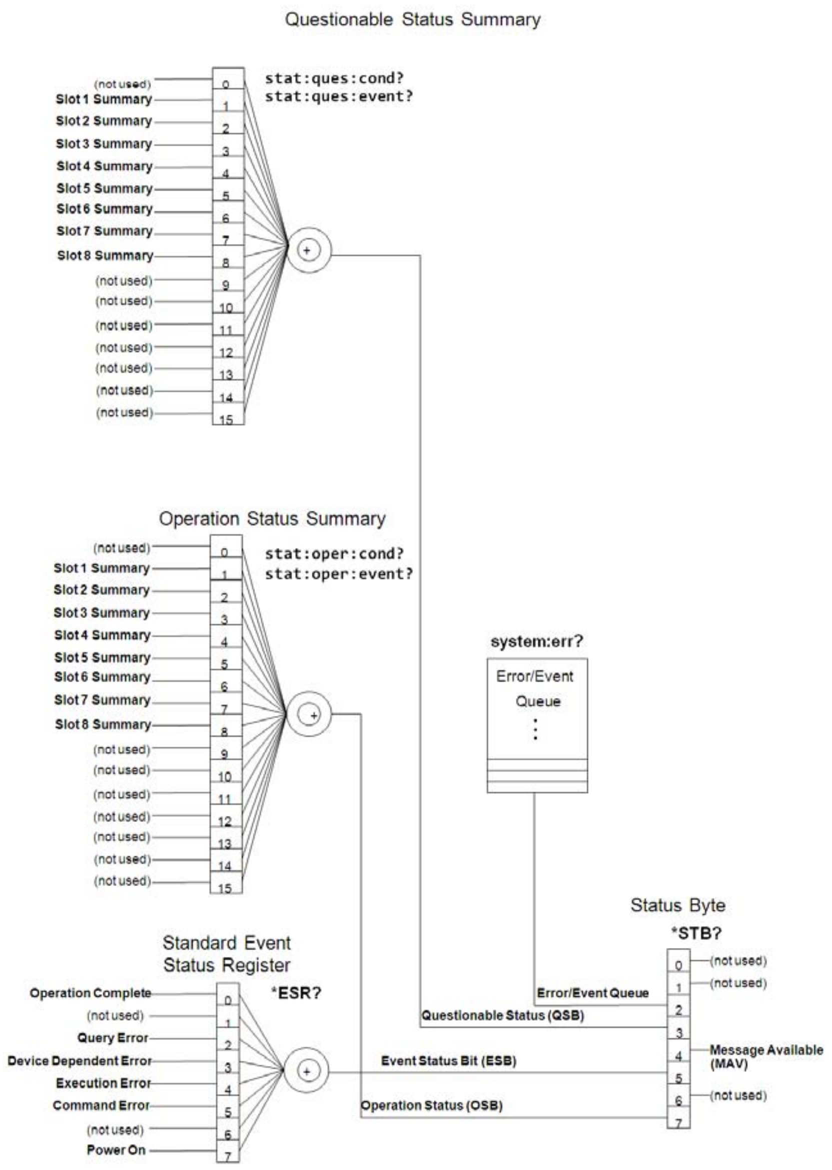


Figure 3 Status Registers for the N775*x* and the N776*x* Variable Optical Attenuators and N771x TLS

Some Comments on Status Registers

- Bit 2 in the Status Byte is set if the Error Queue is not empty.
- Power On bit (bit 7 in the SESR)
    - is not set before bootup is finished.

- can only be read in VXI-11 and USB connection, because other connections are created dynamically and their SESR doesn't exist at boot time.

- Bit names in parentheses are defined in the SCPI standard but are not used by the GenericInstrument.
- Alarm and Error register are not available in this Recovery Image but the drawing was made with Visio and Agilent decided not to use Visio any more.

command:	**:STATus:PRESet**
syntax:	:STATus:PRESet
affects:	N775xA and N776xA attenuators and N771x Tunable Laser Source. See "The Status Model - Multiport Power Meters" on page 44 for information on the equivalent command for the N774xA multiport power meters.
description:	Performs the same as *CLS
parameters:	none
response:	none
example:	stat:pres

command:	**:STATus:OPERation[:EVENt]?**
syntax:	:STATus:OPERation[:EVENt][:LEVel]?
affects:	N775xA and N776xA attenuators and N771x Tunable Laser Source. See "The Status Model - Multiport Power Meters" on page 44 for information on the equivalent command for the N774xA multiport power meters.
description:	Returns the Operational Status Event Summary Register (OSESR).
parameters:	none
response:	The sum of the results for the slots (a *16-bit signed integer* value, where 0 ≤ *value* ≤ 32767) (see Figure 3 on page 58)
example:	stat:oper? ⟶ +0<END>

command:	**:STATus:OPERation:CONDition?**
syntax:	:STATus:OPERation:CONDition[:LEVel]?
affects:	N775xA and N776xA attenuators and N771x Tunable Laser Source. See "The Status Model - Multiport Power Meters" on page 44 for information on the equivalent command for the N774xA multiport power meters.
description:	Reads the Operational Status Condition Summary Register.
parameters:	none
response:	The sum of the results for the individual slots (a *16-bit signed integer* value, where 0 ≤ *value* ≤ 32767):
example:	stat:oper:cond? ⟶ +0<END>

command:	**:STATus:OPERation:ENABle**
syntax:	:STATus:OPERation:ENABle[:LEVel]<wsp><value>
affects:	N775xA and N776xA attenuators and N771x Tunable Laser Source. See "The Status Model - Multiport Power Meters" on page 44 for information on the equivalent command for the N774xA multiport power meters.
description:	Sets the bits in the Operational Status Enable Summary Mask (OSESM) that enable the contents of the OSESR to affect the Status Byte (STB). Setting a bit in this register to 1 enables the corresponding bit in the OSESR to affect bit 7 of the Status Byte.
parameters:	The bit value for the OSESM as a *16-bit signed integer* value (0 .. +32767) (see Figure 3 on page 58) The default value is 0.
response:	none
example:	stat:oper:enab 128

command:	**:STATus:OPERation:ENABle?**
syntax:	:STATus:OPERation:ENABle[:LEVel]?
affects:	N775xA and N776xA attenuators and N771x Tunable Laser Source. See "The Status Model - Multiport Power Meters" on page 44 for information on the equivalent command for the N774xA multiport power meters.
description:	Returns the OSESM for the OSESR
parameters:	none
response:	The bit value for the operation enable mask as a *16-bit signed integer* value (0 .. +32767)
example:	stat:oper:enab? → +128<END>

command:	**:STATus:OPERation:NTRansition**
syntax:	:STATus:OPERation:NTRansition<wsp><value>
affects:	N775xA and N776xA attenuators and N771x Tunable Laser Source.
description:	Set the negative transition filter. Setting a bit in the negative transition filter means that when the corresponding bit of the OPERation condition register transitions from 1 to 0, it causes a 1 to be written in the corresponding bit of the OPERation event register.
parameters:	The bitmask is a *16-bit signed integer* value (0 .. +32767) (see Figure 3 on page 58) The default value is 0.
response:	none
example:	stat:oper:ntr 128

command:	**:STATus:OPERation:NTRansition?**
syntax:	:STATus:OPERation:NTRansition?
affects:	N775xA and N776xA attenuators and N771x Tunable Laser Source.
description:	Returns the negative transition filter bitmask.
parameters:	none

response:	The bit value for the operation negative transition filter, as a *16-bit signed integer* value (0 .. +32767)
example:	stat:oper:ntr? ⟶ +128<END>

command:	**:STATus:OPERation:PTRansition**
syntax:	:STATus:OPERation:PTRansition<wsp><value>
affects:	N775xA and N776xA attenuators and N771x Tunable Laser Source.
description:	Set the positive transition filter. Setting a bit in the positive transition filter means that when the corresponding bit of the OPERation condition register transitions from 0 to 1, it causes a 1 to be written in the corresponding bit of the OPERation event register.
parameters:	The bitmask is a *16-bit signed integer* value (0 .. +32767) (see Figure 3 on page 58) The default value is +32767.
response:	none
example:	stat:oper:ptr 128

command:	**:STATus:OPERation:PTRansition?**
syntax:	:STATus:OPERation:PTRansition?
affects:	N775xA and N776xA attenuators and N771x Tunable Laser Source.
description:	Returns the positive transition filter bitmask.
parameters:	none
response:	The bit value for the operation positive transition filter, as a *16-bit signed integer* value (0 .. +32767)
example:	stat:oper:ntr? ⟶ +128<END>

If bit 5 is set for the instrument, for the individual slots, bits 6 and 7 of the OPERation:EVENt and OPERation:CONDition registers work together as follows:

Offset type	Bit 6	Bit 7	Decimal value
Exact value	0	0	0
Extrapolate below	1	0	64
Extrapolate above	0	1	128
Interpolated	1	1	192

command:	**:STATus*n*:OPERation[:EVENt]?**
syntax:	:STATus*n*:OPERation[:EVENt]?
affects:	N775xA and N776xA attenuators and N771x Tunable Laser Source. See “The Status Model - Multiport Power Meters" on page 44 for information on the equivalent command for the N774xA multiport power meters.
description:	Returns the Operational Slot Status Event Register (OSSER) of slot *n*.

parameters: none

response: The results for the individual slot events (a *16-bit signed integer* value, where 0 ≤ *value* ≤ 32767):

Bit	Mnemonic	Decimal Value
8-15	Not used	0
7	Slot *n*: Offset type bit 1	128
6	Slot *n*: Offset type bit 0	64
5	Slot *n*: Offset enabled or disabled	32
4	Slot *n*: Shutter open	16
3	Slot *n*: Zeroing ongoing	8
2	Not used	4
1	Not used	2
0	Not used	1

example: stat1:oper? → +0<END>

command: **:STATus*n*:OPERation:CONDition?**

syntax: :STATus*n*:OPERation:CONDition?

affects: N775xA and N776xA attenuators and N771x Tunable Laser Source.
See "The Status Model - Multiport Power Meters" on page 44 for information on the equivalent command for the N774xA multiport power meters.

description: Returns the Operational Slot Status Condition Register of slot *n*.

parameters: none

response: The results for the individual slot events (a *16-bit signed integer* value, where 0 ≤ *value* ≤ 32767):

Bit	Mnemonic	Decimal Value
8-15	Not used	0
7	Slot *n*: Offset type bit 1	128
6	Slot *n*: Offset type bit 0	64
5	Slot *n*: Offset enabled or disabled	32
4	Slot *n*: Shutter open	16
3	Slot *n*: Zeroing ongoing	8
2	Not used	4
1	Not used	2
0	Not used	1

example: stat1:oper:cond? → +0<END>

command: **:STATus*n*:OPERation:ENABle**

syntax: :STATus*n*:OPERation:ENABle<wsp><value>

affects: N775xA and N776xA attenuators and N771x Tunable Laser Source.
See "The Status Model - Multiport Power Meters" on page 44 for information on the equivalent command for the N774xA multiport power meters.

description: Sets the bits in the Operation Slot Status Enable Mask (OSSEM) for slot *n* that enable the contents of the Operation Slot Status Event Register (OSSER) for slot *n* to affect the OSESR.
Setting a bit in this register to 1 enables the corresponding bit in the OSSER for slot *n* to affect bit *n* of the OSESR.

parameters: The bit value for the OSSEM as a *16-bit signed integer* value (0 .. +32767)

response:	none
example:	stat:oper:enab 128

command:	**:STATus*n*:OPERation:ENABle?**
syntax:	:STATus*n*:OPERation:ENABle?
affects:	N775xA and N776xA attenuators and N771x Tunable Laser Source. See "The Status Model - Multiport Power Meters" on page 44 for information on the equivalent command for the N774xA multiport power meters.
description:	Returns the OSSEM of slot *n*
parameters:	none
response:	The bit value for the OSSEM as a *16-bit signed integer* value (0 .. +32767)
example:	stat:oper:enab? ⟶ +128<END>

command:	**:STATus:QUEStionable[:EVENt]?**
syntax:	:STATus:QUEStionable[:EVENt][:LEVel]?
affects:	N775xA and N776xA attenuators and N771x Tunable Laser Source. See "The Status Model - Multiport Power Meters" on page 44 for information on the equivalent command for the N774xA multiport power meters.
description:	Returns the Questionable Status Event Summary Register (QSESR).
parameters:	none
response:	The sum of the results for the QSESR as a *16-bit signed integer* value (0 .. +32767) (see Figure 3 on page 58)
example:	stat:ques? ⟶ +0<END>

command:	**:STATus:QUEStionable:CONDition?**
syntax:	:STATus:QUEStionable:CONDition[:LEVel]?
affects:	N775xA and N776xA attenuators and N771x Tunable Laser Source. See "The Status Model - Multiport Power Meters" on page 44 for information on the equivalent command for the N774xA multiport power meters.
description:	Returns the Questionable Status Condition Summary Register.
parameters:	none
response:	The sum of the results for the Questionable Status Condition Summary Register as a *16-bit signed integer* value (0 .. +32767) (see Figure 3 on page 58)
example:	stat:ques:cond? ⟶ +0<END>

command:	**:STATus:QUEStionable:ENABle[:LEVel]**
syntax:	:STATus*n*:QUEStionable:ENABle[:LEVel*0*]<wsp><value>
affects:	N775xA and N776xA attenuators and N771x Tunable Laser Source. See "The Status Model - Multiport Power Meters" on page 44 for information on the equivalent command for the N774xA multiport power meters.

description:	Sets the bits in the Questionable Status Enable Summary Mask (QSESM) that enable the contents of the QSESR to affect the Status Byte (STB). Setting a bit in this register to 1 enables the corresponding bit in the QSESR to affect bit 3 of the Status Byte.
parameters:	The bit value for the questionable enable mask as a *16-bit signed integer* value (0 .. +32767) (see Figure 3 on page 58) The default value is 0.
response:	none
example:	stat:ques:enab 128

command:	**:STATus:QUEStionable:ENABle[:LEVel]?**
syntax:	:STATus:QUEStionable:ENABle[:LEVel*0*]?
affects:	N775xA and N776xA attenuators and N771x Tunable Laser Source. See "The Status Model - Multiport Power Meters" on page 44 for information on the equivalent command for the N774xA multiport power meters.
description:	Returns the QSESM for the event register
parameters:	none
response:	The bit value for the QSEM as a *16-bit signed integer* value (0 .. +32767) (see Figure 3 on page 58)
example:	stat:ques:enab? → +128<END>

command:	**:STATus:QUEStionable:NTRansition**
syntax:	:STATus:QUEStionable:NTRansition<wsp><value>
affects:	N775xA and N776xA attenuators and N771x Tunable Laser Source.
description:	Set the negative transition filter. Setting a bit in the negative transition filter means that when the corresponding bit of the QUEStionable condition register transitions from 1 to 0, it causes a 1 to be written in the corresponding bit of the QUEStionable event register.
parameters:	The bitmask is a *16-bit signed integer* value (0 .. +32767) (see Figure 3 on page 58) The default value is 0.
response:	none
example:	stat:ques:ntr 128

command:	**:STATus:QUEStionable:NTRansition?**
syntax:	:STATus:QUEStionable:NTRansition?
affects:	N775xA and N776xA attenuators and N771x Tunable Laser Source.
description:	Returns the negative transition filter bitmask.
parameters:	none
response:	The bit value for the operation negative transition filter, as a *16-bit signed integer* value (0 .. +32767)
example:	stat:ques:ntr? → +128<END>

command:	**:STATus:QUEStionable:PTRansition**
syntax:	:STATus:QUEStionable:PTRansition<wsp><value>
affects:	N775xA and N776xA attenuators and N771x Tunable Laser Source.
description:	Set the positive transition filter. Setting a bit in the positive transition filter means that when the corresponding bit of the QUEStionable condition register transitions from 0 to 1, it causes a 1 to be written in the corresponding bit of the QUEStionable event register.
parameters:	The bitmask is a *16-bit signed integer* value (0 .. +32767) (see Figure 3 on page 58) The default value is +32767.
response:	none
example:	stat:ques:ptr 128

command:	**:STATus:QUEStionable:PTRansition?**
syntax:	:STATus:QUEStionable:PTRansition?
affects:	N775xA and N776xA attenuators and N771x Tunable Laser Source.
description:	Returns the positive transition filter bitmask.
parameters:	none
response:	The bit value for the operation positive transition filter, as a *16-bit signed integer* value (0 .. +32767)
example:	stat:ques:ntr? $\longrightarrow$ +128<END>

command:	**:STATus*n*:QUEStionable[:EVENt]?**
syntax:	:STATus*n*:QUEStionable[:EVENt]?
affects:	N775xA and N776xA attenuators and N771x Tunable Laser Source. See "The Status Model - Multiport Power Meters" on page 44 for information on the equivalent command for the N774xA multiport power meters.
description:	Returns the Questionable Slot Status Event Register (OSSER) of slot *n*.
parameters:	none
response:	The results for the individual slot events (a *16-bit signed integer* value, where 0 ≤ *value* ≤ 32767): **Bit** / **Mnemonic** / **Decimal Value** 2 - 15 / Not Used / 1 / Slot *n*: Zeroing failed / 2 0 / Slot *n*: Excessive Value / 1
example:	stat1:ques? $\longrightarrow$ +0<END>

command:	**:STATus*n*:QUEStionable:CONDition?**
syntax:	:STATus*n*:QUEStionable:CONDition?
affects:	N775xA and N776xA attenuators and N771x Tunable Laser Source. See "The Status Model - Multiport Power Meters" on page 44 for information on the equivalent command for the N774xA multiport power meters.
description:	Returns the Questionable Slot Status Condition Register for slot *n*.

parameters:	none
response:	The results for the individual slot events (a *16-bit signed integer* value, where 0 ≤ *value* ≤ 32767): **Bit** / **Mnemonic** / **Decimal Value** 2 - 15 / Not Used / 1 / Slot *n*: Zeroing failed / 2 0 / Slot *n*: Excessive Value / 1 Every *n*th bit is the summary of slot *n*.
example:	stat1:ques:cond? ⟶ +0<END>

command:	**:STATus*n*:QUEStionable:ENABle**
syntax:	:STATus*n*:QUEStionable:ENABle<wsp><value>
affects:	N775xA and N776xA attenuators and N771x Tunable Laser Source. See “The Status Model - Multiport Power Meters" on page 44 for information on the equivalent command for the N774xA multiport power meters.
description:	Sets the bits in the Questionable Slot Status Enable Mask (QSSEM) for slot *n* that enable the contents of the Questionable Slot Status Register (QSSR) for slot *n* to affect the QSESR. Setting a bit in this register to 1 enables the corresponding bit in the QSSER for slot *n* to affect bit *n* of the QSESR.
parameters:	The bit value for the QSSEM as a *16-bit signed integer* value (0 .. +32767)
response:	none
example:	stat:ques:enab 128

command:	**:STATus*n*:QUEStionable:ENABle?**
syntax:	:STATus*n*:QUEStionable:ENABle?
affects:	N775xA and N776xA attenuators and N771x Tunable Laser Source. See “The Status Model - Multiport Power Meters" on page 44 for information on the equivalent command for the N774xA multiport power meters.
description:	Returns the QSSEM for slot *n*
parameters:	none
response:	The bit value for the QSSEM as a *16-bit signed integer* value (0 .. +32767)
example:	stat:ques:enab? ⟶ +128<END>

Interface/Instrument Behaviour Settings – The SYSTem Subsystem

The SYSTem subsystem lets you control the instrument's serial interface. You can also control some internal data (like date, time, and so on).

command:	**:SYSTem:DATE**
syntax:	:SYSTem:DATE<wsp><year>,<month>,<day>
affects:	All instruments
description:	The instrument has no internal clock. This command is included for backwards compatibility.

command:	**:SYSTem:DATE?**
syntax:	:SYSTem:DATE?
affects:	All instruments
description:	The instrument has no internal clock. This query is included for backwards compatibility.
parameters:	none
response:	The date in the format year, month, day (*16-bit signed integer* values)
example:	syst:date? → +0000,+0,+00<END>

command:	**:SYSTem:ERRor?**
syntax:	:SYSTem:ERRor?
affects:	All instruments
description:	Returns the next error from the error queue (see "The Error Queue" on page 17). Each error has the error code and a *short* description of the error, separated by a comma, for example 0, "No error". Error codes are numbers in the range -32768 and +32767. Negative error numbers are defined by the SCPI standard. Positive error numbers are device dependent.
parameters:	none
response:	The number of the latest error, and its meaning.
example:	syst:err? → -113,"Undefined header"<END>

command:	**:SYSTem:HELP:HEADers?**
syntax:	:SYSTem:HELP:HEADers?
affects:	All instruments
description:	Returns a list of all SCPI commands.
parameters:	none
response:	Returns a list of all SCPI commands
example:	syst:help:head? → *Returns a list of all SCPI commands*

command:	**:SYSTem:LXI:IDN**
syntax:	:SYSTem:LXI:IDN<wsp><boolean>
affects:	All instruments
description:	On: The LAN LED on the front panel of the instrument flashes for identification. Off: The LAN LED on the front panel of the instrument shows the current LAN state.
parameters:	boolean (0 | 1 | off | on)
response:	none
example:	SYST:LXI:IDN ON

command:	**:SYSTem:PRESet**
syntax:	:SYSTem:PRESet
affects:	All instruments
description:	Sets the insrument to its standard settings. **NOTE:** the stored settings are deleted. By contrast the `:CONFigure:MEASurement:SETTing:PRESet` command keeps the previously stored settings in nonvolatile RAM and they can be recalled again. The following are not affected by this command: the GPIB, USB and LAN (interface) state, the interface addresses, the output and error queues, the Service Request Enable register (SRE), the Status Byte (STB), the Standard Event Status Enable Mask (SESEM), and the Standard Event Status Register (SESR).
parameters:	none
response:	none
example:	SYST:PRES

command:	**:SYSTem:REBoot**
syntax:	:SYSTem:REBoot
affects:	All instruments
description:	Reboots the instrument
parameters:	none
response:	none
example:	SYST:REB

command:	**:SYSTem:TIME**
syntax:	:SYSTem:TIME<wsp><hour>,<minute>,<second>
affects:	All instruments
description:	The instrument has no internal clock. This command is included for backwards compatibility.

command:	**:SYSTem:TIME?**
syntax:	:SYSTem:TIME?
affects:	All instruments
description:	The instrument has no internal clock. This query is included for backwards compatibility.
parameters:	none
response:	The time in the format hour, minute, second. Hours are counted 0...23 (*16-bit signed integer* values).
example:	syst:time? ⟶ +00,+00,+00<END>

command:	**:SYSTem:VERSion?**
syntax:	:SYSTem:VERSion?
affects:	All instruments
description:	Returns the SCPI revision to which the instrument complies.
parameters:	none
response:	The revision year and number.
example:	syst:vers? ⟶ 1990.0<END>

command:	**:SYSTem:COMMunicate:GPIB[:SELF]:ADDRess**	
syntax:	:SYSTem:COMMunicate:GPIB[:SELF]:ADDRess<wsp><GPIB Address>	
affects:	All instruments	
description:	Sets the GPIB address.	
parameters:	The GPIB Address	Values allowed 0-30 21 is often reseververd by the GPIB Controller.
response:	none	
example:	SYST:COMM:GPIB:ADDR 20	

command:	**:SYSTem:COMMunicate:GPIB[:SELF]:ADDRess?**
syntax:	:SYSTem:COMMunicate:GPIB[:SELF]:ADDRess?
affects:	All instruments
description:	Returns the GPIB address.
parameters:	none

response:	The GPIB Address
example:	SYST:COMM:GPIB:ADDR? → +20<END>

:SYSTem:COMMunicate:ETHernet subtree

The optical power meter supports USB and LAN interfaces.The :SYSTem:COMMunicate:ETHernet subtree is only necessary for setting up the LAN (ETHernet).

When first delivered, DHCP is enabled.
If you do not want to use DHCP, or if it is not supported by your network, configure the network settings first over USB. Although later changes can be made using the LAN interface, we recommend always changing ethernet parameters via USB connection, otherwise you may lose your connection.

The default host name is of the format

A-*PPPPPP*-*SSSSSS*

where, *PPPPPP* is the product number (for example, N7745A), and *SSSSSS* is the last six digits of the serial number. For example, A-N7745A-123456.

Some notes on DHCP/AutoIP/DNS

If DHCP is enabled but no DHCP server is found, the power meter tries to use AutoIP as a fallback. AutoIP can take some time (depending on timeout settings).

Depending on the availabe network capabilities the power meter tries to tell the DNS server its host name, or read the host name it has been assigned.

MAC address

The Media Access Control (MAC) number is a unique number associated with each instrument's network adapter.

command:	**:SYSTem:COMMunicate:ETHernet:MACaddress?**
syntax:	:SYSTem:COMMunicate:ETHernet:MACaddress?
affects:	All instruments.
description:	Get the MAC address of the network adapter.
parameters:	none
response:	response string (hex value without a prefix or separators).
example:	:syst:comm:eth:mac? → “0007E014AE08”<END>

Automatically set Ethernet parameters

If DHCP/AutoIP is enabled, the optical power meter may use other parameters than specified explicitly, that is, it will use the parameters provided by the DHCP server. It tries to use its configured hostname (which may fail, depending on the network setup).

There will be an error if you try to query these parameters if the network is not connected, or before they have been set by the DHCP server.

command:	**:SYSTem:COMMunicate:ETHernet:DHCP:ENABle**
syntax:	:SYSTem:COMMunicate:ETHernet:DHCP:ENABle
affects:	All instruments.
description:	Enable or disable DHCP
parameters:	boolean (0 | 1 | off | on)
response:	none
example:	:syst:comm:eth:dhcp:enab on

command:	**:SYSTem:COMMunicate:ETHernet:DHCP:ENABle?**
syntax:	:SYSTem:COMMunicate:ETHernet:DHCP:ENABle?
affects:	All instruments.
description:	Check whether DHCP is enabled or disabled.
parameters:	none
response:	boolean (0 | 1)
example:	:syst:comm:eth:dhcp:enab? $\longrightarrow$ 1<END>

command:	**:SYSTem:COMMunicate:ETHernet:AUTOip:ENABle**
syntax:	:SYSTem:COMMunicate:ETHernet:AUTOip:ENABle
affects:	All instruments. N7744A and N7745A require FW Version 1.16 or higher. For N774*x*A Multiport power meters, you can also control automatic IP addressing in the web interface. Please consult the *User's Guide*.
description:	Enable or disable whether IP addresses can be created automatically by the instrument. Automatic IP addressing is only used if DHCP is enabled, but the instrument cannot find a DHCP server.
parameters:	boolean (0 | 1 | off | on)
response:	none
example:	:syst:comm:eth:auto:enab on

command:	**:SYSTem:COMMunicate:ETHernet:AUTOip:ENABle?**
syntax:	:SYSTem:COMMunicate:ETHernet:AUTOip:ENABle?

affects:	All instruments. N7744A and N7745A require FW Version 1.16 or higher.
description:	Check whether Automatic IP addressing is enabled or disabled. For N774*x*A Multiport power meters, you can also check whether automatic IP addressing is enabled in the web interface. Please consult the *User's Guide*.
parameters:	none
response:	boolean (0 | 1)
example:	:syst:comm:eth:auto:enab? → 1<END>

command:	**:SYSTem:COMMunicate:ETHernet:IPADdress:CURRent?**
syntax:	:SYSTem:COMMunicate:ETHernet:IPADdress:CURRent?
affects:	All instruments.
description:	Get the current IP address of the system.
parameters:	none
response:	string
example:	:syst:comm:eth:ipad:curr? → "192.132.13.2"<END>

command:	**:SYSTem:COMMunicate:ETHernet:SMASk:CURRent?**
syntax:	:SYSTem:COMMunicate:ETHernet:SMASk:CURRent?
affects:	All instruments.
description:	Get the currently used subnet mask.
parameters:	none
response:	string
example:	example :syst:comm:eth:smas:curr? → "255.255.255.0"<END>

command:	**:SYSTem:COMMunicate:ETHernet:HOSTname:CURRent?**
syntax:	:SYSTem:COMMunicate:ETHernet:HOSTname:CURRent?
affects:	All instruments.
description:	Get the current host name. The default host name is A-*P..P*-*S..S*; where A is forAgilent, *P..P* is the Product Number, and *S..S* is as many of the last digits of the Serial Number to get a 15 character hostname. For example: A-N7745A-012345
parameters:	none
response:	string
example:	:syst:comm:eth:host:curr? → "A-N7745A-123456"<END>

command:	**:SYSTem:COMMunicate:ETHernet:DOMainname:CURRent?**
syntax:	:SYSTem:COMMunicate:ETHernet:DOMainname:CURRent?
affects:	All instruments.

description:	Get the currently used domain name
parameters:	none
response:	string
example:	:syst:comm:eth:dom:curr? $\rightarrow$ ".companyname.com"<END>

command:	**:SYSTem:COMMunicate:ETHernet:DGATeway:CURRent?**
syntax:	SYSTem:COMMunicate:ETHernet:DGATeway:CURRent?
affects:	All instruments.
description:	Get the currently used default gateway.
parameters:	none
response:	string (maximum 79 characters)
example:	:syst:comm:eth:dgat:curr? $\rightarrow$ "192.168.101.11"<END>

Explicitly set Ethernet parameters

You must reboot the instrument or send a SYST:COMM:ETH:RESTart command before any alterations to the Ethernet parameters become effective.

If you query one of the alterable parameters, you always get the most recently set value, even if you have not yet activated it.

To undo any changes before they become active, send SYST:COMM:ETH:CANCel.

command:	**:SYSTem:COMMunicate:ETHernet:HOSTname**
syntax:	:SYSTem:COMMunicate:ETHernet:HOSTname
affects:	All instruments.
description:	Set the host name
parameters:	string (maximum 79 characters, though not all characters can be used)
response:	none
example:	:syst:comm:eth:host "powermeter1"

command:	**:SYSTem:COMMunicate:ETHernet:HOSTname?**
syntax:	:SYSTem:COMMunicate:ETHernet:HOSTname?
affects:	All instruments.
description:	Get the host name
parameters:	none
response:	string
example:	:syst:comm:eth:host? $\rightarrow$ "powermeter1"<END>

command:	**SYSTem:COMMunicate:ETHernet:DOMainname**
syntax:	:SYSTem:COMMunicate:ETHernet:DOMainname
affects:	All instruments.
description:	Set the domain name (used if DHCP is disabled)
parameters:	string (maximum 79 characters, though not all characters can be used)
response:	none
example:	:syst:comm:eth:dom ".companyname.com"

command:	**:SYSTem:COMMunicate:ETHernet:DOMainname?**
syntax:	:SYSTem:COMMunicate:ETHernet:DOMainname?
affects:	All instruments.
description:	Get the domain name
parameters:	none
response:	string
example:	:syst:comm:eth:dom? $\rightarrow$ ".companyname.com"<END>

command:	**:SYSTem:COMMunicate:ETHernet:IPADdress**
syntax:	:SYSTem:COMMunicate:ETHernet:IPADdress
affects:	All instruments.
description:	Set the IP address of the system manually (used if DHCP is disabled).
parameters:	string (maximum 79 characters, though only integers and the period, ".", can be used)
response:	none
example:	:syst:comm:eth:ipad "192.132.13.2"

command:	**:SYSTem:COMMunicate:ETHernet:IPADdress?**
syntax:	:SYSTem:COMMunicate:ETHernet:IPADdress?
affects:	All instruments.
description:	Get the manually set IP address of the system.
parameters:	none
response:	string
example:	:syst:comm:eth:ipad? $\rightarrow$ "192.132.13.2"<END>

command:	**:SYSTem:COMMunicate:ETHernet:SMASk**
syntax:	:SYSTem:COMMunicate:ETHernet:SMASk
affects:	All instruments.
description:	Set the subnet mask.
parameters:	string (maximum 79 characters, though only integers and the period, ".", can be used)

response:	none
example:	:syst:comm:eth:smas “255.255.255.0“

command:	**:SYSTem:COMMunicate:ETHernet:SMASk?**
syntax:	:SYSTem:COMMunicate:ETHernet:SMASk?
affects:	All instruments.
description:	Get the subnet mask.
parameters:	none
response:	string
example:	:syst:comm:eth:smas? $\longrightarrow$ “255.255.255.0”<END>

command:	**:SYSTem:COMMunicate:ETHernet:DGATeway**
syntax:	:SYSTem:COMMunicate:ETHernet:DGATeway
affects:	All instruments.
description:	Set the default gateway.
parameters:	string (maximum 79 characters, though only integers and the period, “.”, can be used)
response:	none
example:	:syst:comm:eth:dgat “192.168.101.11“

command:	**:SYSTem:COMMunicate:ETHernet:DGATeway?**
syntax:	SYSTem:COMMunicate:ETHernet:DGATeway?
affects:	All instruments.
description:	Get the default gateway.
parameters:	none
response:	string
example:	:syst:comm:eth:dgat? $\longrightarrow$ “192.168.101.11“<END>

Changing the Ethernet parameters

In most cases, instead of using :SYSTem:COMMunicate:ETHernet:RESTart, it is better to save the new parameters (:SYSTem:COMMunicate:SAVE) and reboot the instrument (:SYSTem:REBoot).

command:	**:SYSTem:COMMunicate:ETHernet:RESet**
syntax:	:SYSTem:COMMunicate:ETHernet:RESet
affects:	All instruments

description:	Resets all the LAN parameters to the factory default. • DHCP On • AutoIP On • Hostname is a concatenation of product number and serial number. • The password for the web based LAN configuration interface is reset to 'agilent'. This command is also triggered when the reset button on the front panel is pressed longer than 3 seconds.
parameters:	none
response:	none
example:	:syst:comm:eth:res

command:	**:SYSTem:COMMunicate:ETHernet:RESTart**
syntax:	:SYSTem:COMMunicate:ETHernet:RESTart
affects:	All instruments.
description:	Restart the system's network interface with the new parameters. This command only works if the instrument has a working network connection at the time the command is issued. If you are connected by USB, use :SYSTem:COMMunicate:ETHernet:SAVE followed by :SYSTem:REBoot.
parameters:	none
response:	none
example:	:syst:comm:eth:rest

command:	**:SYSTem:COMMunicate:ETHernet:SAVE**
syntax:	:SYSTem:COMMunicate:ETHernet:SAVE
affects:	All instruments.
description:	Save the system's network interface parameters.
parameters:	none
response:	none
example:	:syst:comm:eth:save

command:	**:SYSTem:COMMunicate:ETHernet:CANCel**
affects:	All instruments.
syntax:	:SYSTem:COMMunicate:ETHernet:CANCel
description:	Undo all changes to the network parameters that have been made since the last save, reboot or ":syst:comm:eth:restart" command.
parameters:	none
response:	none
example:	:syst:comm:eth:canc

Handling Measurement Settings - The :CONFigure:MEASurement:SETTing subtree

The instrument can store a number of settings in FLASH memory. The number of memory places can be queried with CONF:MEAS:SETT:NUMBer?.

- For the Multiport Power Meters, a measurement setting consists of all parameters which can be set with the :SENSe:* commands.
- For the variable optical attenuators, a measurement setting consists of all parameters of the following commands:
    - INPut:CHANnel:ATTenuation
    - INPut:CHANnel:ATTenuation:SPEed
    - INPut:CHANnel:OFFSet
    - INPut:CHANnel:WAVelength
    - OUTput:CHANnel:ATIMe
    - OUTput:CHANnel:POWer:OFFSet
    - OUTPut:CHANnel:POWer:UNit
    - OUTPut:CHANnel:POWer:CONTRol
    - OUTPut:CHANnel:POWer
    - OUTPut:CHANnel[:STATe]
    - OUTPut:CHANnel[:STATe]:APOWeron
    - The table of wavelength dependent offsets is automatically stored in the non-volatile RAM when the offset table state has been set to ON. (That is, :CONFigure:OFFSet:WAVelength:STATe 1)

After each parameter change, you can use :SYSTem:ERRor? to check if it is OK.

command:	**:CONFigure:MEASurement:SETTing:ACTual?**
syntax:	:CONFigure:MEASurement:SETTing:ACTual?
affects:	All instruments.
description:	Get the index of the setting currently being used.
parameters:	none

response:	int A value >0 is returned if the setting has been stored in FLASH memory (using :CONFigure:MEASurement:SETTing:SAVE), or has been recalled from FLASH memory (using :CONFigure:MEASurement:SETTing:RECall), and has not been changed since. 0 is returned if the setting has not yet been stored. 0 is returned if the FLASH setting has been deleted (using:CONFigure:MEASurement:SETTing:ERASe) since the last recall or store. -1 is returned if the setting was changed but has not been saved yet.
example:	:conf:meas:sett:act? → +2<END>

command:	**:CONFigure:MEASurement:SETTing:NUMBer?**
syntax:	:CONFigure:MEASurement:SETTing:NUMBer?
affects:	All instruments.
description:	Get the number of settings. In addition to the settings spaces in FLASH memory, the working memory can hold a setting.
parameters:	none
response:	int
example:	:conf:meas:sett:numb? → +1<END>

command:	**:CONFigure:MEASurement:SETTing:PRESet**
syntax:	:CONFigure:MEASurement:SETTing:PRESet
affects:	All instruments.
description:	Resets the setting values in the working memory. In contrast to the *RST and System:Preset commands, the previous stored settings remain in nonvolatile RAM and can be recalled again.
parameters:	none (N774x Multiport Power Meters). String (user name, optional, on the N775x / N776x attenuators and N771x Tunable Laser Source)
response:	none
example:	:conf:meas:sett:pres

command:	**:CONFigure:MEASurement:SETTing:CANCel**
syntax:	:CONFigure:MEASurement:SETTing:CANCel
affects:	All instruments.
description:	Discard all the changes to the setting since the last save or recall
parameters:	none
response:	none
example:	:conf:meas:sett:canc

command:	**:CONFigure:MEASurement:SETTing:RECall**
syntax:	:CONFigure:MEASurement:SETTing:RECall
affects:	All instruments.
description:	Recall a setting from FLASH memory.
parameters:	int (setting index), string (user name, optional, on the N775*x* and NZZ6*x* attenuators)
response:	none
example:	:conf:meas:sett:rec 1

command:	**:CONFigure:MEASurement:SETTing:SAVE**
syntax:	:CONFigure:MEASurement:SETTing:SAVE
affects:	All instruments.
description:	Save the current setting to FLASH memory.
parameters:	int (setting index), string (user name, optional, on the N775*x* and NZZ6*x* attenuators)
response:	none
example:	:conf:meas:sett:sav 1

command:	**:CONFigure:MEASurement:SETTing:ERASe**
syntax:	:CONFigure:MEASurement:SETTing:ERASe
affects:	All instruments.
description:	Erase a setting from memory.
parameters:	int (setting index)
response:	none
example:	:conf:meas:sett:eras 1

4
Measurement Operations & Settings

This chapter gives descriptions of commands that you can use when you are setting up or performing measurements. The commands are split up into the following subsystems:

- Root layer commands that take power measurements, configures triggering, and return information about the instrument and it's slots
- SENSe subsystem commands that control Power Sensors.
- TRIGger subsystem commands that control triggering.

Root Layer Command

The commands in the Slot subsystem allow you to query a particular port n, identified here as slot for compatibility with the 816x platform.

command:	**:SLOT[n]:OPC?**
syntax:	:SLOT[n]:OPC?
affects:	N775xA and N776xA attenuators and N771xA Tunable Laser Source.
description:	The Operation Complete query returns 1 if all operations of the selected Slot are completed (0 otherwise). In contrast the standard *OPC? Command returns 1 when all operations of all slots are completed.
parameters:	none
response:	A string
example:	slot1:opc? ⟶ 1

command:	**:SLOT[n][:HEAD[m]]:OPTions?**
syntax:	:SLOT[n][:HEAD[m]]:OPTions?
affects:	All instruments
description:	Returns information about the slot options.
parameters:	none
response:	A string
example:	slot1:head:opt? ⟶ NO CONNECTOR OPTION, NO INSTRUMENT OPTIONS<END>

command:	**:SLOT[n][:HEAD[n]]:IDN?**
syntax:	:SLOT[n][:HEAD[n]]:IDN?
affects:	Agilent N774xA Multiport Optical Power Meter and Agilent N775xA/6xA Multichannel Attenuators
description:	Returns information about the slot.
parameters:	none
response:	Agilent Technologies: mmmm: ssssssss: rrrrrrrrr: manufacturer instrument model number (for example N7752A) serial number firmware revision
example:	slot1:head:idn? ⟶ Agilent Technologies, N7752A-002, DE49300071, V1.0<END>

command:	**:SLOT[n][:HEAD[m]]:WAVelength:RESPonse?**
syntax:	:SLOT[n][:HEAD[m]]:WAVelength:RESPonse?
affects:	Agilent N774xA Multiport Optical Power Meter and Agilent N775xA/6xA Multichannel Attenuators
description:	Returns the wavelength response from a wavelength calibrated slot in binary format.
parameters:	none
response:	Wavelength Response table as a binary block. One 8 byte long wavelength calibration value pair consisting of a 4 byte long float for wavelength and a 4 byte long float for the scalar calibration factor. For more information on binary block formats see "Data Types" on page 20
example:	slot1:head1:wav:resp? ⟶ #536570........

command:	**:SLOT[n][:HEAD[m]]:WAVelength:RESPonse:CSV?**
syntax:	:SLOT[n][:HEAD[m]]:WAVelength:RESPonse:CSV?
affects:	Agilent N774xA Multiport Optical Power Meter and Agilent N775xA/6xA Multichannel Attenuators
description:	Returns the wavelength response from the attenuator in CSV format.
parameters:	none
response:	Wavelength Response table as a string The string is a comma separated value (CSV) list and can be written to a file and be processed with a spreadsheet program. List format: λ1, c1\n λ2, c2\n .... λn, cn\n "," separates wavelength and response factor "\n" (= ASCII code 10) separates value pairs
example:	slot1:head1:wav:resp:csv? ⟶ 1200e-6,2.019\n 1210e-6, 1.956\n...

command:	**:SLOT[n][:HEAD[m]]:WAVelength:RESPonse:SIZE?**
syntax:	:SLOT[n][:HEAD[m]]:WAVelength:RESPonse:SIZE?
affects:	Agilent N774xA Multiport Optical Power Meter and Agilent N775xA/6xA Multichannel Attenuators
description:	Returns the number of elements in the wavelength response table.
parameters:	none
response:	Number of elements in the wavelength table as an integer value
example:	slot2:head1:wav:resp:size? ⟶ 50<END>

Signal Generation - The SOURce subtree

This chapter describes the SCPI Commands for use of 81950A compact tunable laser modules and N7711A, N7714A single and 4-port Tunable Laser System Sources.

Summary

- This chapter refers to both the 81950A module and the N7711A/N7714A instruments. The N771xA command set is derived from the 8163A/B, 8164A/B, and 8166A/B command set.
- Please refer to the *Agilent 8163A/B, 8164A/B, 8166A/B Programming Guide* for an introduction to programming instruments and for information about programming and syntax conventions. If you are using an 8163B or 8164B with LAN support please also refer to the *Agilent 8163B Lightwave Multimeter, Agilent 8164B Lightwave Measurement System LAN Interface Addendum to the User and Programming Guides*.
- An "X" in the "New" column in the table below means that the command was not implemented in previous instruments and is not yet described in another manual.
- Source specific commands not listed in the table below are not supported.
- The "[:CHANnel[m]]" part means the channel (i.e. sub-module) of a module in a modular instrument like the 8163A/B, 8164A/B, 8166A/B platform. The 81950A module the N7711A and N7714A instruments accept only ":CHANnel1" which is also the default, so this optional part can be omitted.

Detailed Description

The OUTPut sub tree

Command	**:OUTPut[n][:CHANNel[m]:POWer:UNit**
Syntax	:OUTPut[n][:CHANNel[m]]:POWer:UNit<wsp>DBM|0|Watt|1
Description	Sets whether the power unit used is dBm or W.
Parameters	DBM or 0: Sets the power unit to dBm Watt or 1: Sets the power unit to W
Response	none
Example	outp1:pow:un w
Affects	81950A compact tunable laser modules and N7711A, N7714A
Notes	This is the same as [:SOURce[n]][:CHANNel[m]:POWer:UNIT

Command	**:OUTPut[n][:CHANNel[m]:POWer:UNit?**
Syntax	:OUTPut[n][:CHANNel[m]]:POWer:UNit?
Description	Queries whether the power unit used is dBm or W.
Parameters	none
Response	0: the power unit is dBm +1: the power unit is W
Example	outp1:pow:un? ⟶ +1<END>
Affects	81950A compact tunable laser modules and N7711A, N7714A
Notes	This is the same as [:SOURce[n]][:CHANNel[m]:POWer:UNIT?

Command	**:OUTPut[n][:CHANNel[m][:STATe]**
Syntax	:OUTPut[n][:CHANNel[m][:STATe]<wsp>OFF|0|ON|1
Description	Switches the laser on or off.
Parameters	OFF or 0: Disable the output ON or 1: Enable the output
Response	none
Example	outp1 on
Affects	81950A compact tunable laser modules and N7711A, N7714A
Notes	This is the same as [:SOURce[n]][:CHANNel[m]:POWer:STATe

Command	**:OUTPut[n][:CHANNel[m][:STATe]?**
Syntax	:OUTPut[n][:CHANNel[m][:STATe]?
Description	Queries the laser state.
Parameters	none
Response	0: Disabled 1: Enabled
Example	outp1? ⟶ 1
Affects	81950A compact tunable laser modules and N7711A, N7714A
Notes	This is the same as [:SOURce[n]][:CHANnel[m]:POWer:STATe?

The SOURce:CHANnel:POWer sub tree

Command	**:SOURce[n]][:CHANnel[m]]:POWer[:LEVel][:IMMediate][:AMPLitude]**
Syntax	[:SOURce[n]][:CHANnel[m]]:POWer[:LEVel][:IMMediate][:AMPLitude]<wsp> <value>[MW|Watt|DBM]|MIN|MAX|DEF
Description	Sets the power of the laser output.
Parameters	Any value in the specified range (see appropriate User's Guide) Also allowed: MIN - minimum programmable value MAX - maximum programmable value DEF - default value
Response	none
Example	sour1:pow 19mW
Affects	81950A compact tunable laser modules and N7711A, N7714A
Notes	

Command	**:SOURce[n]][:CHANnel[m]]:POWer[:LEVel][:IMMediate][:AMPLitude]?**
Syntax	[:SOURce[n]][:CHANnel[m]]:POWer[:LEVel][:IMMediate][:AMPLitude]?<wsp> [MIN|MAX|DEF]
Description	Returns the power of the laser output in the unit selected with the unit command.
Parameters	Optional: MIN - minimum programmable value MAX - maximum programmable value DEF - default value
Response	The current power value or the minimum, maximum, or default power value.

Example	sour1:pow? ⟶ +2.00000000E-002
Affects	81950A compact tunable laser modules and N7711A, N7714A
Notes	

Command	**[:SOURce[n]][:CHANnel[m]]:POWer:STATe**
Syntax	[:SOURce[n]][:CHANnel[m]:POWer:STATe<wsp>OFF|0|ON|1
Description	Switches the laser on or off.
Parameters	OFF or 0: Disable the output ON or 1: Enable the output
Response	none
Example	sour1:pow:stat on
Affects	81950A compact tunable laser modules and N7711A, N7714A
Notes	This is the same as :OUTPut[n][:CHANnel[m]:STATe

Command	**[:SOURce[n]][:CHANnel[m]]:POWer:STATe?**
Syntax	[:SOURce[n]][:CHANnel[m]]:POWer:STATe?
Description	Queries the laser state.
Parameters	none
Response	0: Disabled 1: Enabled
Example	sour1:pow:stat? ⟶ 1
Affects	81950A compact tunable laser modules and N7711A, N7714A
Notes	This is the same as :OUTPut[n][:CHANnel[m]:STATe?

Command	**[:SOURce[n]][:CHANnel[m]]:POWer:UNIT**
Syntax	[:SOURce[n]][:CHANnel[m]]:POWer:UNIT<wsp>DBM|0|Watt|1
Description	Sets whether the power unit used is dBm or W.
Parameters	DBM or 0: Sets the power unit to dBm Watt or 1: Sets the power unit to W
Response	none
Example	sour1:pow:unit w
Affects	81950A compact tunable laser modules and N7711A, N7714A
Notes	This is the same as :OUTPut[n][:CHANnel[m]:POWer:UNIT

Command	**[:SOURce[n]][:CHANnel[m]]:POWer:UNIT?**
Syntax	[:SOURce[n]][:CHANnel[m]]:POWer:UNIT?
Description	Queries whether the power unit used is dBm or W.
Parameters	none
Response	0: the power unit is dBm +1: the power unit is W
Example	outp1:pow:un? → +1<END>
Affects	81950A compact tunable laser modules and N7711A, N7714A
Notes	This is the same as :OUTPut[n][:CHANnel[m]:POWer:UNIT?

The SOURce:CHANnel:WAVelength sub tree

Command	**[:SOURce[n]][:CHANnel[m]]:WAVelength[:CW|:FIXED]**
Syntax	[:SOURce[n]][:CHANnel[m]]:WAVelength[:CW|:FIXED]<wsp> <value>[PM|NM|UM|MM|M]|MIN|MAX|DEF
Description	Sets the laser wavelength

Parameters	Any value in the specified range (see appropriate User's Guide)
	Also allowed:
	    MIN - minimum programmable value
	    MAX - maximum programmable value
	    DEF - default value

Response	none
Example	sour1:wav 1600nm
Affects	81950A compact tunable laser modules and N7711A, N7714A
Notes	Only available in "Auto Mode". "CW" and "FIXED" have no specific meaning, these mnemonics were just added for backward compatibility.

Command	**[:SOURce[n]][:CHANnel[m]]:WAVelength[:CW|:FIXED]?**
Syntax	[:SOURce[n]][:CHANnel[m]]:WAVelength[:CW|:FIXED]? <wsp>[MIN|MAX|DEF]
Description	Returns the wavelength value in meters.

Parameters	Optional:
	    MIN - minimum programmable value
	    MAX - maximum programmable value
	    DEF - default value

Response	The current wavelength value or the minimum, maximum, or default wavelength value.
Example	sour1:wav? → +1.60000000E-006
Affects	81950A compact tunable laser modules and N7711A, N7714A
Notes	"CW" and "FIXED" have no specific meaning, these mnemonics were just added for backward compatibility

Command	**[:SOURce[n]][:CHANnel[m]]:WAVelength:TOGRid**
Syntax	[:SOURce[n]][:CHANnel[m]]:WAVelength:TOGRid<wsp><value>[PM|NM|UM|MM|M]
Description	Set the wavelength to the nearest grid point only changing the channel number.
Parameters	The wavelength value.
Response	none
Example	sour4:wav:togr 1.6um
Affects	81950A compact tunable laser modules and N7711A, N7714A
Notes	Only available in "Grid Mode" The frequency offset is neither changed nor taken into account when calculating the nearest channel.

Command	**[:SOURce[n]][:CHANnel[m]]:WAVelength:AUTO**
Syntax	[:SOURce[n]][:CHANnel[m]]:WAVelength:AUTO<wsp>OFF|0|ON|1
Description	Switch between "Auto Mode" and "Grid Mode",
Parameters	OFF or 0: "Grid Mode" (= Auto Mode off) ON or 1: "Auto Mode"
Response	none
Example	sour1:wav:auto on
Affects	81950A compact tunable laser modules and N7711A, N7714A
Notes	This command is equivalent to [:SOURce[n][:CHANnel[m]]:FREQuency:AUTO

Command	**[:SOURce[n]][:CHANnel[m]]:WAVelength:AUTO?**
Syntax	[:SOURce[n]][:CHANnel[m]]:WAVelength:AUTO?
Description	Returns the current operation mode.
Parameters	none

Response	0: "Grid Mode" (= Auto Mode off) 1: "Auto Mode"
Example	sour1:wav:auto? → 0
Affects	81950A compact tunable laser modules and N7711A, N7714A
Notes	This query is equivalent to [:SOURce[n][:CHANnel[m]]:FREQuency:AUTO?

The SOURce:CHANnel:FREQuency sub tree

Command	**[:SOURce[n]][:CHANnel[m]]:FREQuency**
Syntax	[:SOURce[n]][:CHANnel[m]]:FREQuency<wsp> <value>[HZ|KHZ|MHZ|GHZ|THZ]|MIN|MAX|DEF
Description	Sets the laser frequency.
Parameters	Any value in the specified range (see appropriate User's Guide) Also allowed: MIN - minimum programmable value MAX - maximum programmable value DEF - default value
Response	None
Example	sour1:freq 188THz
Affects	81950A compact tunable laser modules and N7711A, N7714A
Notes	Only available in "Auto Mode".

Command	**[:SOURce[n]][:CHANnel[m]]:FREQuency?**
Syntax	[:SOURce[n]][:CHANnel[m]]:FREQuency?<wsp>[MIN|MAX|DEF]
Description	Returns the frequency value in Hz.

Parameters	Optional:
	    MIN - minimum programmable value
	    MAX - maximum programmable value
	    DEF - default value

Response	The current frequency value or the minimum, maximum, or default frequency value.
Example	sour1:freq? → +1.88000000E+014
Affects	81950A compact tunable laser modules and N7711A, N7714A
Notes	

Command	**[:SOURce[n]][:CHANnel[m]]:FREQuency:REFerence**
Syntax	[:SOURce[n]][:CHANnel[m]]:FREQuency:REFerence<wsp> <value>[HZ|KHZ|MHZ|GHZ|THZ]|MIN|MAX|DEF
Description	Set the reference frequency for the frequency grid used by the instrument.

Parameters	Any value in the specified range (see appropriate User's Guide)
	Also allowed:
	    MIN - minimum programmable value
	    MAX - maximum programmable value
	    DEF - default value

Response	None
Example	sour1:freq:ref 193.1THz
Affects	81950A compact tunable laser modules and N7711A, N7714A
Notes	Only available in "Grid Mode"

Command	**[:SOURce[n]][:CHANnel[m]]:FREQuency:REFerence?**
Syntax	[:SOURce[n]][:CHANnel[m]]:FREQuency:REFerence?<wsp>[MIN|MAX|DEF]
Description	Returns the reference frequency for the frequency grid used by the instrument.

Parameters	Optional:
	    MIN - minimum programmable value
	    MAX - maximum programmable value
	    DEF - default value

Response	The current reference frequency value or the minimum, maximum, or default reference frequency value in Hz.
Example	sour1:freq:ref? $\longrightarrow$ +1.93100000E+014
Affects	81950A compact tunable laser modules and N7711A, N7714A
Notes	

Command	**[:SOURce[n]][:CHANnel[m]]:FREQuency:GRID**
Syntax	[:SOURce[n]][:CHANnel[m]]:FREQuency:GRID<wsp> <value>[HZ|KHZ|MHZ|GHZ|THZ]|MIN|MAX|DEF
Description	Set the grid spacing for the frequency grid used by the instrument.

Parameters	Any value in the specified range (see appropriate User's Guide)
	Also allowed:
	    MIN - minimum programmable value
	    MAX - maximum programmable value
	    DEF - default value

Response	None
Example	sour1:freq:grid 50e9

Affects	81950A compact tunable laser modules and N7711A, N7714A
Notes	Only available in "Grid Mode"

Command	**[:SOURce[n]][:CHANnel[m]]:FREQuency:GRID?**
Syntax	[:SOURce[n]][:CHANnel[m]]:FREQuency:GRID?<wsp>[MIN|MAX|DEF]
Description	Returns the grid spacing for the frequency grid used by the instrument.
Parameters	Optional: MIN - minimum programmable value MAX - maximum programmable value DEF - default value
Response	The current grid spacing value or the minimum, maximum, or default grid spacing value in Hz.
Example	sour1:freq:grid? → +5.00000000E+010
Affects	81950A compact tunable laser modules and N7711A, N7714A
Notes	

Command	**[:SOURce[n]][:CHANnel[m]]:FREQuency:CHANnel**
Syntax	[:SOURce[n]][:CHANnel[m]]:FREQuency:CHANnel<wsp> <value>|MIN|MAX|DEF
Description	Sets the laser's grid channel.
Parameters	Any value resulting in a valid frequency. Also allowed: MIN - minimum programmable value MAX - maximum programmable value DEF - default value
Response	None
Example	sour1:freq:chan -20
Affects	81950A compact tunable laser modules and N7711A, N7714A
Notes	Only available in "Grid Mode". The minimum, maximum, and default value depend on the current reference frequency and grid spacing values.

Command	**[:SOURce[n]][:CHANnel[m]]:FREQuency:CHANnel?**
Syntax	[:SOURce[n]][:CHANnel[m]]:FREQuency:CHANnel?<wsp>[MIN|MAX|DEF]
Description	Returns the laser's grid channel.

Parameters	Optional: MIN - minimum programmable value MAX - maximum programmable value DEF - default value
Response	The current grid channel value or the minimum, maximum, or default grid channel value.
Example	sour1:freq:chan? → -20
Affects	81950A compact tunable laser modules and N7711A, N7714A
Notes	The minimum, maximum, and default value depend on the current reference frequency and grid spacing values.

Command	**[:SOURce[n]][:CHANnel[m]]:FREQuency:OFFSet**
Syntax	[:SOURce[n]][:CHANnel[m]]:FREQuency:OFFset<wsp> <value>[HZ|KHZ|MHZ|GHZ|THZ]|MIN|MAX|DEF
Description	Set the frequency offset used by the instrument.
Parameters	Any value in the specified range (see appropriate User's Guide) Also allowed: MIN - minimum programmable value MAX - maximum programmable value DEF - default value
Response	None
Example	sour1:freq:offs 0.1e9
Affects	81950A compact tunable laser modules and N7711A, N7714A
Notes	Only available in "Grid Mode"

Command	**[:SOURce[n]][:CHANnel[m]]:FREQuency:OFFSet?**
Syntax	[:SOURce[n]][:CHANnel[m]]:FREQuency:OFFSet?<wsp>[MIN|MAX|DEF]
Description	Returns the frequency offset used by the instrument in Hz.
Parameters	Optional: MIN - minimum programmable value MAX - maximum programmable value DEF - default value
Response	The current frequency offset value or the minimum, maximum, or default frequency offset value in Hz.
Example	sour1:freq:offs? → +1.00000000E+008
Affects	81950A compact tunable laser modules and N7711A, N7714A
Notes	

Command	**[:SOURce[n]][:CHANnel[m]]:FREQuency:TOGRid**
Syntax	[:SOURce[n]][:CHANnel[m]]:FREQuency:TOGRid <wsp> <value>[HZ|KHZ|MHZ|GHZ|THZ]
Description	Set the frequency to the nearest grid point only changing the channel number.
Parameters	The frequency value.
Response	None
Example	sour4:freq:togr 188.5thz
Affects	81950A compact tunable laser modules and N7711A, N7714A
Notes	Only available in "Grid Mode" The frequency offset is neither changed nor taken into account when calculating the nearest channel.

Command	**[:SOURce[n]][:CHANnel[m]]:FREQuency:AUTO**
Syntax	[:SOURce[n]][:CHANnel[m]]:FREQuency:AUTO<wsp>OFF|0|ON|1
Description	Switch between "Auto Mode" and "Grid Mode",
Parameters	OFF or 0: "Grid Mode" (= Auto Mode off) ON or 1: "Auto Mode"
Response	None
Example	sour1:freq:auto on
Affects	81950A compact tunable laser modules and N7711A, N7714A
Notes	This command is equivalent to [:SOURce[n][:CHANnel[m]]:WAVelength:AUTO

Command	**[:SOURce[n]][:CHANnel[m]]:FREQuency:AUTO?**
Syntax	[:SOURce[n]][:CHANnel[m]]:FREQuency:AUTO?
Description	Returns the current operation mode.
Parameters	none
Response	0: "Grid Mode" (= Auto Mode off) 1: "Auto Mode"
Example	sour1:freq:auto? ⟶ 0
Affects	81950A compact tunable laser modules and N7711A, N7714A
Notes	This query is equivalent to [:SOURce[n][:CHANnel[m]]:WAVelength:AUTO?

The SOURce:CHANnel:MODUlation sub tree

Command	**[:SOURce[n]][:CHANnel[m]]:MODUlation:INTernal[:STATe]**
Syntax	[:SOURce[n]][:CHANnel[m]]:MODUlation:INTernal[:STATe]<wsp>OFF|0|ON|1
Description	Switches the internal modulation (SBS suppression) on or off.
Parameters	OFF or 0: Disable the modulation ON or 1: Enable the modulation
Response	None
Example	sour1:modu:int on
Affects	81950A compact tunable laser modules and N7711A, N7714A
Notes	SBSControl must be set to an appropriate value, too.

Command	**[:SOURce[n]][:CHANnel[m]]:MODUlation:INTernal[:STATe]?**
Syntax	[:SOURce[n]][:CHANnel[m]]:MODUlation:INTernal[:STATe]?
Description	Returns the internal modulation state.
Parameters	None
Response	0: off 1: on
Example	sour1:modu:int? $\longrightarrow$ 0
Affects	81950A compact tunable laser modules and N7711A, N7714A
Notes	

Command	**[:SOURce[n]][:CHANnel[m]]:MODUlation:INTernal:SBSControl[:LEVel]**
Syntax	[:SOURce[n]][:CHANnel[m]]:MODUlation:INTernal:SBSControl[:LEVel] <wsp> <value>[HZ|KHZ|MHZ|GHZ|THZ]|MIN|MAX|DEF
Description	Sets the internal frequency modulation frequency.
Parameters	Any value in the specified range (see appropriate User's Guide) Also allowed: MIN - minimum programmable value MAX - maximum programmable value DEF - default value
Response	None
Example	sour1:modu:int:sbsc 0.2GHz
Affects	81950A compact tunable laser modules and N7711A, N7714A
Notes	

Command	**[:SOURce[n]][:CHANnel[m]]:MODUlation:INTernal:SBSControl[:LEVel]?**
Syntax	[:SOURce[n]][:CHANnel[m]]:MODUlation:INTernal:SBSControl[:LEVel]?<wsp> [MIN|MAX|DEF]
Description	Returns the internal frequency modulation frequency in Hz.
Parameters	Optional: MIN - minimum programmable value MAX - maximum programmable value DEF - default value
Response	The current modulation frequency value or the minimum, maximum, or default modulation frequency value in Hz.
Example	sour1:modu:int:sbsc? → +2.00000000E+008
Affects	81950A compact tunable laser modules and N7711A, N7714A
Notes	

Command	**[:SOURce[n]][:CHANnel[m]]:MODUlation:INTernal:RATE?**
Syntax	[:SOURce[n]][:CHANnel[m]]:MODUlation:INTernal:RATE?
Description	Returns the internal modulation rate in Hz.
Parameters	None
Response	The internal modulation rate in Hz.
Example	sour:modu:int:rate? → +2.00000000E+004
Affects	N7711A, N7714A only
Notes	There is no corresponding command, because the hardware does not allow setting this value.

Command	**[:SOURce[n]][:CHANnel[m]]:MODUlation:INTernal:WAVeform?**
Syntax	[:SOURce[n]][:CHANnel[m]]:MODUlation:INTernal:WAVeform?
Description	Returns the internal modulation waveform.
Parameters	None
Response	SIN: sinusoidal TRI: triangular
Example	sour:modu:int:wav? → SIN
Affects	N7711A, N7714A only
Notes	There is no corresponding command, because the hardware does not allow setting this value.

Command	**[:SOURce[n]][:CHANnel[m]]:MODUlation:EXTernal[:STATe]**
Syntax	[:SOURce[n]][:CHANnel[m]]:MODUlation:EXTernal[:STATe]<wsp>OFF|0|ON|1
Description	Sets the state of the external amplitude modulation.
Parameters	OFF or 0: off ON or 1: on
Response	None
Example	sour1:modu:ext on
Affects	81950A only
Notes	

Command	**[:SOURce[n]][:CHANnel[m]]:MODUlation:EXTernal[:STATe]?**
Syntax	[:SOURce[n]][:CHANnel[m]]:MODUlation:EXTernal[:STATe]?
Description	Returns the state of the external amplitude modulation.
Parameters	None
Response	0: off 1: on
Example	sour1:modu:ext? → 1
Affects	81950A only
Notes	

Command	**[:SOURce[n]][:CHANnel[m]]:MODUlation:EXTernal:AM[:LEVel]**
Syntax	[:SOURce[n]][:CHANnel[m]]:MODUlation:EXTernal:AM[:LEVel] <wsp> <value>|MIN|MAX|DEF
Description	Sets the amplitude modulation level in %.
Parameters	Any value in the specified range (see appropriate User's Guide) Also allowed: MIN - minimum programmable value MAX - maximum programmable value DEF - default value
Response	None
Example	sour1:modu:ext:am 2.0
Affects	81950A only
Notes	

Command	**[:SOURce[n]][:CHANnel[m]]:MODUlation:EXTernal:AM[:LEVel]?**
Syntax	[:SOURce[n]][:CHANnel[m]]:MODUlation:EXTernal:AM[:LEVel]?<wsp> [MIN|MAX|DEF]
Description	Returns the amplitude modulation level in %.
Parameters	Optional: MIN - minimum programmable value MAX - maximum programmable value DEF - default value
Response	The current amplitude modulation level or the minimum, maximum, or default amplitude modulation level.
Example	sour1:modu:ext:am? → +2.00000000E+000
Affects	81950A only
Notes	

Error Messages

Error messages returned by the "SYSTem:ERRor?" query:

- *-221, "Not allowed while laser is on"* - you tried to set grid spacing or reference frequency without switching off the laser.
- *-221, "Not allowed while frequency auto mode is on"* - you tried to issue a command which is only supported in grid mode.
- *-221, "Not allowed while frequency auto mode is off"* - you tried to issue a command which is only supported in auto mode.
- *-221, "Settings conflict"* - returned if the laser module reports an invalid configuration

Status Bits

If the frequency offset doesn't equal 0, bit 12 in the module's questionable status register is set (= value 4096).

Note: This only applies to *Grid Mode*; in *Auto Mode* the frequency offset status bit is not set.

The questionable condition and event registers can be queried using

- :STATus[m]:QUEStionable:CONDition?
- :STATus[m]:QUEStionable:EVENt?

Please refer to the Agilent 8163A/B, 8164A/B, 8166A/B *Programming Guide* for detailed information about the status model.

Differences among 81950A, N7711A and N7714A

Some commands are not available on both platforms

81950A Only

- [:SOURce[n]][:CHANnel[m]]:MODUlation:EXTernal[:STATe]/?
- [:SOURce[n]][:CHANnel[m]]:MODUlation:EXTernal:AM[:LEVel]/?

N771A, N7714A Only

- [:SOURce[n]][:CHANnel[m]]:MODUlation:INTernal:RATE?
- [:SOURce[n]][:CHANnel[m]]:MODUlation:INTernal:WAVeform?

Signal Conditioning - The INPut, OUTput and CONFiguration:CHANnel subtrees

The commands in this section allow you to control attenuators.

The INPut subtree

command:	**:INPut[*n*][:CHANnel[*m*]]:ATTenuation**
syntax:	:INPut[*n*][:CHANnel[*m*]]:ATTenuation<wsp><value>[dB] | MIN | DEF | MAX
affects:	N775xA and N776xA attenuators
description:	Sets the attenuation factor (α) for the slot. The attenuation factor is used, together with an offset factor (α_{Offset}) to set the filter attenuation (α_{filter}). $\alpha_{(new)}$ (dB) = $\alpha_{filter\ (new)}$ (dB) + α_{Offset} (dB) Set the attenuation factor by sending a value (the default units are dB), or by sending MIN, DEF, or MAX.
parameters:	<value>[dB] The attenuation in dB. MIN | DEF The values where α_{filter} = 0dB MAX The value where α_{filter} is at its greatest.
response:	none
example:	INP1:ATT 14dB

command:	**:INPut[*n*][:CHANnel[*m*]]:ATTenuation?**
syntax:	:INPut[*n*][:CHANnel[*m*]]:ATTenuation?<wsp> MIN | DEF | MAX
affects:	N775xA and N776xA attenuators
description:	Returns the current attenuation factor (α), in dB. α (dB) = α_{filter} (dB) + α_{Offset} (dB)
parameters:	MIN | DEF | MAX Returns the minimum, default, or maximum value of the attenuation factor possible.
response:	4 byte Intel floating point; attenuation in dB.
example:	INP1:ATT? $\longrightarrow$ 14<END>

command:	**:INPut[:CHANnel[*m*]]:ATTenuation:ALL**
syntax:	:INPut[:CHANnel[*m*]]:ATTenuation:ALL<wsp><value>[dB]
affects:	N775xA and N776xA attenuators
description:	Sets the attenuation factor (as described above) for all attenuators that support it. If any of the attenuators do not support the attenuation factor, an error is returned (-224, illegal parameter value)
parameters:	<value>[dB] The attenuation in dB.

response:	none
example:	INP:ATT:ALL 12

command:	**:INPut[*n*][:CHANnel[*m*]]:OFFSet**
syntax:	:INPut[*n*][:CHANnel[*m*]]:OFFSet<wsp><value>[dB] | MIN | DEF | MAX
affects:	N775xA and N776xA attenuators
description:	Sets the offset factor (α_{Offset}) for the slot. This factor does not affect the filter attenuation (α_{filter}). It is used to offset the attenuation factor values. This offset factor is used, with the attenuation factor, to set the attenuation of the filter. In this way it is possible to compensate for external losses. $\alpha_{(new)}$ (dB) = α_{filter} (dB) + $\alpha_{Offset\ (new)}$ (dB) Set the offset factor by sending a value (the default units are dB), or by sending MIN, DEF, or MAX.
parameters:	<value>[dB] The offset factor (α_{Offset}) in dB. MIN Sets the minimum value for α_{Offset} = - 200dB. DEF Sets the default value for α_{Offset} = 0dB. MAX Sets the maximum value for α_{Offset} = + 200dB.
response:	none
example:	INP1:OFFS 2dB

command:	**:INPut[*n*][:CHANnel[*m*]]:OFFSet?**
syntax:	:INPut[*n*][:CHANnel[*m*]]:OFFSet?<wsp>MIN | DEF | MAX
affects:	N775xA and N776xA attenuators
description:	Returns the current value of the offset factor (α_{Offset}), in dB.
parameters:	MIN | DEF | MAX Returns the minimum, default, or maximum value of the offset factor.
response:	4 byte Intel floating point; offset in dB.
example:	INP1:OFFS? $\longrightarrow$ 2<END>

command:	**:INPut[*n*][:CHANnel[*m*]]:OFFSet:DISPlay**
syntax:	:INPut[*n*][:CHANnel[*m*]]:OFFSet:DISPlay
affects:	N775xA and N776xA attenuators
description:	Sets the offset factor (α_{Offset}) such that the attenuation factor is zero. $\alpha_{Offset\ (new)}$ (dB) = $\alpha_{Offset\ (old)}$ (dB) - $\alpha_{(old)}$ (dB) = - α_{filter} (dB)
parameters:	none
response:	none
example:	INP1:OFFS:DISP

command:	**:INPut[*n*][:CHANnel[*m*]]:OFFSet:POWermeter**
syntax:	:INPut[*n*][:CHANnel[*m*]]:OFFSet:POWermeter<wsp><slot>,<channel>

affects:	N775xA and N776xA attenuators.
description:	Sets the offset factor (α_{Offset}) to the difference between a power value measured by another powermeter (P_{ext}) and the power value measured by the attenuator's monitor diode (P_{att}). α_{Offset} (dB) = P_{att} *(dBm)* - P_{ext} (dBm)
parameters:	<slot> Slot number of the external powermeter. <channel> Channel number of the external powermeter.
response:	none
example:	INP1:OFFS:POW 4,2

command:	**:INPut[*n*][:CHANnel[*m*]]:ATTenuation:SPEed**
syntax:	:INPut[*n*][:CHANnel[*m*]]:ATTenuation:SPEed<wsp><value> | MIN | MAX | DEF
affects:	N775xA and N776xA attenuators.
description:	Sets the filter transition speed; the speed at which the slot moves from one attenuation to another (in dB/s). **NOTE:** The Optical Multimode Attenuators N7766A to N7768A allow controlled speeds from 0.1 dB/s to 80 dB/s and a full speed of approximately 1000 dB/s. All speed inputs higher than 80 dB/s are set to 1000dB/s.
parameters:	<value> The filter transition speed in dB/s. MIN | MAX | DEF Sets the filter transition speed to the instrument limits, or the instrument default.
response:	none
example:	INP1:ATT:SPE 2

command:	**:INPut[*n*][:CHANnel[*m*]]:ATTenuation:SPEed?**
syntax:	:INPut[*n*][:CHANnel[*m*]]:ATTenuation:SPEed?<wsp> MIN | MAX | DEF
affects:	N775xA and N776xA attenuators.
description:	Without the optional parameter, queries the transition speed of the filter. **NOTE:** The Optical Multimode Attenuators N7766A to N7768A allow controlled speeds from 0.1 dB/s to 80 dB/s and a full speed of approximately 1000 dB/s. All speed inputs higher than 80 dB/s are set to 1000 dB/s.
parameters:	MIN | MAX | DEF Queries the transition speed limits, or the instrument default.
response:	4 byte Intel floating point; transition speed in dB/s.
example:	INP1:ATT:SPE? $\longrightarrow$ 2<END>

command:	**:INPut[*n*][:CHANnel[*m*]]:WAVelength**
syntax:	:INPut[*n*][:CHANnel[*m*]]:WAVelength<wsp><value>[PM | NM | UM| MM | M] | MIN | MAX | DEF
affects:	N775xA and N776xA attenuators.

description:	Sets the attenuator's operating wavelength. This value is used to compensate for the wavelength dependence of the filter, and to calculate a wavelegth dependent offset from the user offset table (if enabled).
parameters:	<value> The wavelength in meters (if you do not specify a unit). MIN | MAX | DEF Sets the wavelength to the instrument limits, or the instrument default.
response:	none
example:	INP1:WAV +1.55000000E-006

command:	**:INPut[*n*][:CHANnel[*m*]]:WAVelength?**
syntax:	:INPut[*n*][:CHANnel[*m*]]:WAVelength?<wsp>MIN | MAX | DEF
affects:	N775xA and N776xA attenuators.
description:	Without the optional parameter, queries the operating wavelength of the attenuator.
parameters:	MIN | MAX | DEF Queries the operating wavelength limits, or the instrument default.
response:	4 byte Intel floating point; wavelength in m.
example:	INP1:WAV → +1.55000000E-006<END>

command:	**:INPut[:CHANnel[*m*]]:WAVelength:ALL**
syntax:	:INPut[:CHANnel[*m*]]:WAVelength:ALL<wsp><value>[PM | NM | UM| MM | M]
affects:	N775xA and N776xA attenuators.
description:	Sets the attenuator's operating wavelength (as described above) for all attenuators.
parameters:	<value> The wavelength in meters (if you do not specify a unit).
response:	none
example:	INP:WAV:ALL +1.55000000E-006

The OUTput subtree

command:	**:OUTPut*[n]*[:CHANnel[*m*]]:APMode**
syntax:	:OUTPut*[n]*[:CHANnel[*m*]]:APMode<wsp><OFF(0) | ON(1)>
affects:	N775xA and N776xA attenuators.
description:	The use of this command is optional and has no effect on operation. Included for compatibility with Agilent 8156A mainframe.
parameters:	OFF or 0 ON or 1
response:	none
example:	OUTP1:APMode OFF

command:	**:OUTPut*[n]*[:CHANnel[*m*]]:APMode?**
syntax:	:OUTPut*[n]*[:CHANnel[*m*]]:APMode?
affects:	N775xA and N776xA attenuators.

description:	Queries whether the user has amended the power value or the attenuation value. This use of this command is optional. Included for compatibility with Agilent 8156A mainframe.
parameters:	none
response:	*boolean* 0 User has amended the attenuation value. 1 User has amended the power value.
example:	OUTP1:APMode? ⟶ 0<END>

command:	**:OUTPut*[n]*[:CHANnel[*m*]]:POWer**
syntax:	:OUTPut*[n]*[:CHANnel[*m*]]:POWer<wsp><value>[PW | NW | UW | MW | W | DBM] | MIN | MAX | DEF
affects:	N775xA and N776xA attenuators.
description:	Sets the output power value (*P*). If your attenuator does not include the power control feature, the new filter attenuation ($\alpha_{\text{filter (new)}}$) is calculated from the reference power (P_{ref}) : $P_{\text{set(new)}}$ *(dBm)* = P_{ref} *(dBm)* - $\alpha_{\text{filter (new)}}$ (dB) - P_{Offset} (dB) If your attenuator includes the power control feature, the filter attenuation is changed until the set power (measured by the slot's internal power meter) has been reached. $P_{\text{set(new)}}$ *(dBm)* = $P_{\text{att(new)}}$ *(dBm)* - P_{offset} *(dB)* If the set power cannot be achieved ExP (indicating 'Excessive Power') is displayed, and the appropriate GPIB status bit is set. The status of these bits can be queried using ":STATus:OPERation:CONDition[:LEVel]?" on page 49. NOTE: The powercontrol uses the configured attenuation speed limit to reach the target power for the first time. After that it uses the maximum possible attenuation speed. This ensures the best possible PDL and input power compensation.
parameters:	<value> Desired output power (current unit is used). MIN | MAX | DEF Sets the output power to the instrument limits, or the instrument default.
response:	none
example:	OUTP1:POW 12

command:	**:OUTPut*[n]*[:CHANnel[*m*]]:POWer?**
syntax:	:OUTPut*[n]*[:CHANnel[*m*]]:POWer<wsp>MIN | MAX | DEF
affects:	N775xA and N776xA attenuators.
description:	Without the optional parameter, queries the output power value.
parameters:	MIN | MAX | DEF Queries the output power instrument limits, or the instrument default.
response:	4 byte Intel floating point; output power in current power unit.
example:	OUTP1:POW? ⟶ 12<END>

command:	**:OUTPut[:CHANnel[*m*]]:POWer:ALL**
syntax:	:OUTPut[:CHANnel[*m*]]:POWer:ALL<wsp><value>[PW | NW | UW | MW | W | DBM]
affects:	N775xA and N776xA attenuators.

description:	Sets the output power value (as described above) for all the attenuators that support it. If any attenuator cannot support the value, an error is returned (-224, illegal parameter value)
parameters:	<value> Desired output power (current unit is used).
response:	none
example:	OUTP:POW:ALL 12

command:	**:OUTPut*[n]*[:CHANnel[*m*]]:POWer:OFFSet**
syntax:	:OUTPut*[n]*[:CHANnel[*m*]]:POWer:OFFSet<wsp><value>[DB] | MIN | MAX | DEF
affects:	N775xA and N776xA attenuators.
description:	Sets a power offset (P_{offset}). This factor is used to offset the power value. It does not affect the filter, nor does it change the power output at the attenuator. $P_{set(new)}$ *(dBm)* = P_{att}*(dBm)* - $P_{offset(new)}$ *(dB)* If the wavelength offset table is enabled, the corresponding λ offset is added to this offset.
parameters:	<value> The power offset required, in dB MIN | MAX | DEF Queries the instrument limits, or the default.
response:	none
example:	OUTP1:POW:OFFS 2

command:	**:OUTPut*[n]*[:CHANnel[*m*]]:POWer:OFFSet?**
syntax:	:OUTPut*[n]*[:CHANnel[*m*]]:POWer:OFFSet? <wsp>MIN | MAX| DEF
affects:	N775xA and N776xA attenuators.
description:	Without the optional parameter, queries the power offset value.
parameters:	MIN | MAX | REF Queries the power offset limits, or the instrument default.
response:	4 byte Intel floating point; power offset in current power units.
example:	OUTP1:POW:OFFS? ⟶ 2<END>

command:	**:OUTPut*[n]*[:CHANnel[*m*]]:POWer:OFFSet:POWermeter**
syntax:	:OUTPut*[n]*[:CHANnel[*m*]]:POWer:OFFSet:POWermeter<wsp><slot>,<channel>
affects:	N775xA and N776xA attenuators.
description:	Calculates the power offset by subtracting the power value measured by another powermeter from the power value measured by the attenuator's integrated powermeter, and stores it as P_{offset} . $P_{offset(new)}$ (dBm) = P_{att} (dBm) - P_{ext} (dBm) + $P_{offset(\lambda)}$ (dB)
parameters:	<slot> Slot number of the external powermeter. <channel> Channel number of the external powermeter.
response:	none
example:	OUTP1:POW:OFFS:POW 4,4

command:	**:OUTPut*[n]*[:CHANnel[*m*]]:POWer:CONTRol**
syntax:	:OUTPut*[n]*[:CHANnel[*m*]]:POWer:CONTRol<wsp>OFF(0) | ON(1)
affects:	N775xA and N776xA attenuators.
description:	Sets whether the power control mode is on or off. If power control is enabled, the attenuator automatically compensates for changes to input power.
parameters:	OFF or 0 Output power follows changes to input power. ON or 1 The filter position automatically adjusts to compensate for changes to input power, so maintaining the output power set by the user.
response:	none
example:	OUTP1:POW:CONTR ON

command:	**:OUTPut*[n]*[:CHANnel[*m*]]:POWer:CONTRol?**
syntax:	:OUTPut*[n]*[:CHANnel[*m*]]:POWer:CONTRol?
affects:	N775xA and N776xA attenuators.
description:	Queries whether the power control mode is on or off.
parameters:	none
response:	*boolean* 0 The power control mode is off 1 The power control mode is on.
example:	OUTP1:POW:CONTR? $\longrightarrow$ 0<END>

command:	**:OUTPut*[n]*[:CHANnel[*m*]]:POWer:UNit**
syntax:	:OUTPut*[n]*[:CHANnel[*m*]]:POWer:UNit<wsp>DBM(0) | WATT(1)
affects:	N775xA and N776xA attenuators.
description:	Sets whether the power unit used is dBm or Watts. This setting affects P_{set} , P_{ref} (if available), and P_{act}
parameters:	DBM (or 0) Sets the power unit to dBm WATT (or 1) Sets the power unit to W
response:	none
example:	OUTP1:POW:UN DBM

command:	**:OUTPut*[n]*[:CHANnel[*m*]]:POWer:UNit?**
syntax:	:OUTPut*[n]*[:CHANnel[*m*]]:POWer:UNit?
affects:	N775xA and N776xA attenuators.
description:	Queries whether the power unit is dBm or W
parameters:	none
response:	*boolean* 0 The power unit is dBm 1 The power unit is W.
example:	OUTP1:POW:UN? $\longrightarrow$ 0<END>

command:	**:OUTPut*[n]*[:CHANNel[*m*]][:STATe]**
syntax:	:OUTPut*[n]*[:CHANNel[*m*]][:STATe]<wsp>OFF(0) | ON(1)
affects:	N775xA and N776xA attenuators.
description:	Sets the state of the shutter. NOTE: the Shutter is closed at maximum speed, and opened at the configurated attenuation speed.
parameters:	OFF or 0 The “shutter” is closed. ON or 1 The “shutter” is open.
response:	none
example:	OUTP1:STAT OFF

command:	**:OUTPut*[n]*[:CHANNel[*m*]][:STATe]?**
syntax:	:OUTPut*[n]*[:CHANNel[*m*]][:STATe]?
affects:	N775xA and N776xA attenuators.
description:	Queries the state of the shutter.
parameters:	none
response:	*boolean* 0 The “shutter” is closed. 1 The “shutter” is open.
example:	OUTP1:STAT? ⟶ 0<END>

command:	**:OUTPut[:CHANNel[*m*]][:STATe]:ALL**
syntax:	:OUTPut[:CHANNel[*m*]][:STATe]<wsp>OFF(0) | ON(1)
affects:	N775xA and N776xA attenuators.
description:	Sets the state of the shutter for all attenuators. NOTE: the Shutter is closed at maximum speed, and opened at the configurated attenuation speed.
parameters:	OFF or 0 The “shutters” are closed. ON or 1 The “shutters” are open.
response:	none
example:	OUTP:STAT:ALL OFF

command:	**:OUTPut*[n]*[:CHANNel[*m*]]:STATe:APOWeron**
syntax:	:OUTPut*[n]*[:CHANNel[*m*]]:STATe:APOWeron<wsp>OFF(0) | ON(1)
affects:	N775xA and N776xA attenuators.
description:	Sets the state of the “shutter” when the attenuator is turned on. This setting has to be saved (see “:CONFigure:MEASurement:SETTing:SAVE" on page 79) before it is active.
parameters:	OFF or 0 The “shutter” is closed after power on. ON or 1 The “shutter” is open after power on.

response:	none
example:	OUTP1:APOW OFF

command:	**:OUTPut*[n]*[:CHANNel[*m*]]:STATe:APOWeron?**
syntax:	:OUTPut*[n]*[:CHANNel[*m*]]:STATe:APOWeron?
affects:	N775xA and N776xA attenuators.
description:	Queries the state of the shutter at power on.
parameters:	none
response:	*boolean* 0 The “shutter” is closed after power on. 1 The “shutter” is open after power on.
example:	OUTP1:APOW? $\longrightarrow$ 0<END>

command:	**:OUTPut*[n]*[:CHANNel[*m*]]:ATIMe**
syntax:	:OUTPut*[n]*[:CHANNel[*m*]]:ATIMe<wsp><value>[NS | US | MS| S]
affects:	N775xA and N776xA attenuators.
description:	Sets the powermeter averaging time The power control always has an averaging time of 2ms, to ensure the best possible PDL and input power compensation.
parameters:	<value> The averaging time (in seconds if no unit specified).
response:	none
example:	OUTP1:ATIM 1s

command:	**:OUTPut*[n]*]:CHANNel[*m*]]:ATIMe?**
syntax:	:OUTPut*[n]*[:CHANNel[*m*]]:ATIMe?
affects:	N775xA and N776xA attenuators.
description:	Queries the powermeter averaging time.
parameters:	none
response:	*4 byte Intel floating point; the averaging time in seconds*
example:	OUTP1:ATIM? $\longrightarrow$ 1<END>

command:	**:OUTPut*[n]*[:CHANNel[*m*]]:CORRection:COLLection:ZERO**
syntax:	:OUTPut*[n]*[:CHANNel[*m*]]:CORRection:COLLection:ZERO
affects:	N775xA and N776xA attenuators and power meter channels in N775xA attenuators.
description:	Zeros the electrical offsets of the attenuator’s integrated powermeter. Note: While zeroing an attenuator's power monitor or a power meter in an N775*x*A attenuator instrument, all attenuators in the unit are stopped and all shutters are closed. This assures the best possible zeroing performance.
parameters:	none

response:	none
example:	OUTP1:CORR:COLL:ZERO

command:	**:OUTPut*[n]*[:CHANnel[*m*]]:CORRection:COLLection:ZERO?**
syntax:	:OUTPut*[n]*[:CHANnel[*m*]]:CORRection:COLLection:ZERO?
affects:	N775xA and N776xA attenuators and power meter channels in N775xA attenuators.
description:	Queries the status of the last :OUTPut[n][:CHANnel[m]]:CORRection:COLLection:ZERO operation.
parameters:	none
response:	integer 0 = OK, otherwise not OK.
example:	OUTP1:CORR:COLL:ZER0? → 0<END>

command:	**:OUTPut[:CHANnel[*m*]]:CORRection:COLLection:ZERO:ALL**
syntax:	:OUTPut[:CHANnel[*m*]]:CORRection:COLLection:ZERO:ALL
affects:	N775xA and N776xA attenuators and power meter channels in N775xA attenuators.
description:	Zero all available powermeter channels. Note: While zeroing an attenuator's power monitor or a power meter in an N775*x*A attenuator instrument, all attenuators in the unit are stopped and all shutters are closed. This assures the best possible zeroing performance.
parameters:	none
response:	none
example:	OUTP:CORR:COLL:ZERO:ALL

command:	**:OUTPut[:CHANnel[*m*]]:CORRection:COLLection:ZERO:ALL?**
syntax:	:OUTPut[:CHANnel[*m*]]:CORRection:COLLection:ZERO:ALL?
affects:	N775xA and N776xA attenuators and power meter channels in N775xA attenuators.
description:	Returns the status of the most recent zero all command. The result is backed up in the nonvolatile RAM. Note: If a channel fails to zero, it continues to use the result of the last successful zeroing.
parameters:	none

response:	A hexadecimal integer value which represents the result for all channels. Each hexadecimal digit represents one channel . • 0: zero succeeded without errors. • Any other number: remote zeroing failed (the number is the error code returned from the operation).
example:	outp:chan:corr:coll:zero:all? ⟶ 272 272 decimal = 0x110. This means zeroing failed on channels 2 and 3. All other channels were successful. Alternatively the result can be found with the individual commands: outp1:chan:corr:coll:zero? ⟶ 0 outp2:chan:corr:coll:zero? ⟶ 1 outp3:chan:corr:coll:zero? ⟶ 1

The table of wavelength-dependent offsets

When enabled, the attenuator uses its λ offset table to compensate for wavelength dependent losses in the test set-up. This table contains, for each wavelength specified, the additional power offset to be applied.

- If the attenuator is set to a wavelength corresponding to an entry in its λ offset table, the stored offset is added to the global power offset.
- If the attenuator is set to a wavelength between entries in its λ offset table, linear interpolation is used to calculate the appropriate offset to add to the global power offset.
- If the attenuator is set to a wavelength beyond the range of the entries in its λ offset table, the offset stored for the nearest wavelength is added to the global power offset.
- Whether an exact, interpolated, or extrapolated offset value is applied, the algorithm applied can be queried using ":STATusn:OPERation:CONDition?" on page 51

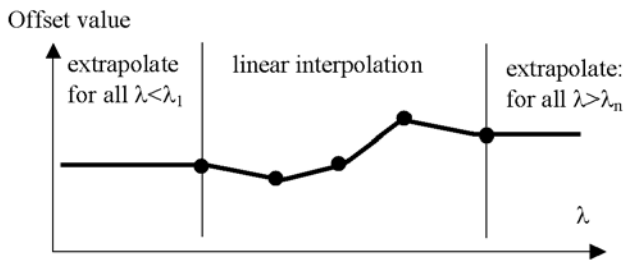

Offset value
extrapolate for all $\lambda<\lambda_1$
linear interpolation
extrapolate: for all $\lambda>\lambda_n$
λ

Figure 1 Extrapolation and interpolation of attenuator λ offset table

command:	**:CONFigure[*n*][:CHANnel[*m*]]:OFFSet:WAVelength:STATe**
syntax:	:CONFigure[n]][:CHANnel[m]]:OFFSet:WAVelength:STATe<wsp>OFF(0) | ON(1)
affects:	N775xA and N776xA attenuators.
description:	Specifies whether the attenuator uses its λ offset table to compensate for wavelength dependent losses in the the test set-up. This table contains the additional power offset to be applied, for each wavelength specified. This command does not affect the instrument's internal enviromental temperature and optical wavelength compensation, which remain active.
parameters:	OFF or 0 — The offset table is not used to compensate for wavelength dependent losses. ON or 1 — The attenuator adds the appropriate value from its λ offset table to the global power offset.
response:	none
example:	CONF1:OFFS:WAV:STAT ON

command:	**:CONFigure[*n*][:CHANnel[*m*]]:OFFSet:WAVelength:STATe?**
syntax:	:CONFigure[n]][:CHANnel[m]]:OFFSet:WAVelength:STATe?
affects:	N775xA and N776xA attenuators.
description:	Queries whether the attenuator uses power values from its λ offset table .
parameters:	none
response:	*boolean* 0 The offset table is not used. 1 The attenuator uses its λ offset table.
example:	CONF1:OFFS:WAV:STAT? → 0<END>

command:	**:CONFigure[*n*][:CHANnel[*m*]]:OFFSet:WAVelength:VALue**
syntax:	:CONFigure[n]][:CHANnel[m]]:OFFSet:WAVelength:VALue<wsp><lambda>[PM | NM | UM | MM| M],<offset[DB]> | TOREF
affects:	N775xA and N776xA attenuators.
description:	Adds a value pair (wavelength; offset) to the offset table, or overwrites an existing value pair. The offset table entries are ordered from shortest to longest wavelength. NOTE: You can only edit the table of wavelength dependent offsets if the table state has been set to OFF. That is, :CONFigure:OFFSet:WAVelength:STATe 0. The table of wavelength dependent offsets is automatically stored in the non-volatle RAM when the offset table state has been set to ON. (That is, :CONFigure:OFFSet:WAVelength:STATe 1).

parameters:	<lambda>	The wavelength for the offset table entry, in m
	<offset>	The power offset to be applied at <lambda>. To query the current power value measured by an attenuator see "The FETCh subtree" on page 115 and "The READ subtree" on page 117.
	TOREF	Calculates the difference between the power measured by an external powermeter and the power measured by the attenuator's integrated powermeter, and stores it as the offset. $P_{\text{Offset}(\lambda)}(\text{dB}) = P_{\text{att}}(\text{dBm}) - P_{\text{ext}}(\text{dBm})$ See: ":CONFigure[n][:CHANnel[m]]:OFFSet:WAVelength:REFerence?" on page 112.
response:	none	
example:	CONF1:OFFS:WAV:VAL +1.55000000E-006,TOREF	

command:	**:CONFigure[*n*][:CHANnel[*m*]]:OFFSet:WAVelength:REFerence**
syntax:	:CONFigure[n]][:CHANnel[m]]:OFFSet:WAVelength:REFerence<wsp><slot>,<channel>
affects:	N775xA and N776xA attenuators.
description:	Specifies the slot and channel of the external powermeter used by TOREF. See: ":CONFigure[n][:CHANnel[m]]:OFFSet:WAVelength:VALue" on page 111
parameters:	<slot> Slot number of the powermeter. <channel> Channel number of the powermeter.
response:	none
example:	CONF1:OFFS:WAV:REF 4,2

command:	**:CONFigure[*n*][:CHANnel[*m*]]:OFFSet:WAVelength:REFerence?**
syntax:	:CONFigure[n]][:CHANnel[m]]:OFFSet:WAVelength:REFerence?
affects:	N775xA and N776xA attenuators.
description:	Queries the currently selected slot and channel of the external powermeter used by TOREF. See: ":CONFigure[n][:CHANnel[m]]:OFFSet:WAVelength:VALue" on page 111
parameters:	none
response:	the slot and channel of the external powermeter as *integer* values.
example:	CONF1:OFFS:WAV:REF? → +2,+1<END>

command:	**:CONFigure[*n*][:CHANnel[*m*]]:OFFSet:WAVelength:VALue:DELete:ALL**
syntax:	:CONFigure[*n*]][:CHANnel[*m*]]:OFFSet:WAVelength:VALue:DELete:ALL **CAUTION** *This command clears the offset table!*
affects:	N775xA and N776xA attenuators.
description:	Deletes every value pair (wavelength:offset) from the offset table.
parameters:	none

response:	*none*
example:	CONF1:OFFS:WAV:VAL:DEL:ALL

command:	**:CONFigure[*n*][:CHANNel[*m*]]:OFFSet:WAVelength:TABle?**
syntax:	:CONFigure[*n*]][:CHANNel[*m*]]:OFFSet:WAVelength:TABle?
affects:	N775xA and N776xA attenuators.
description:	Queries the complete offset table.
parameters:	none
response:	*Binary block format format (Intel byte order); wavelength:offset pairs in ascending order.* Each value pair is transferred as 12 bytes; 8 bytes represent the wavelength, 4 bytes represent the offset.
example:	CONF1:OFFS:WAV:TAB? ⟶ binary block interpreted as, for example: 1.55e-6 | 12 | 1.7e-6 |3.4 |.....

command:	**:CONFigure[*n*][:CHANNel[*m*]]:OFFSet:WAVelength:TABle:SIZE?**	
syntax:	:CONFigure[*n*]][:CHANNel[*m*]]:OFFSet:WAVelength:TABle:SIZE?<wsp>MAX | MIN	
affects:	N775xA and N776xA attenuators.	
description:	Without optional parameter, queries the size of the offset table.	
parameters:	MAX	Queries the maximum size of the offset table. (available flash memory ⟶ 1000 entries)
	MIN	Queries the minimum size of the offset table. (should ⟶ 0)
response:	*4 byte unsigned integer; offset table size.*	
example:	CONF1:OFFS:WAV:TAB:SIZE? ⟶ 50	

command:	**:CONFigure[*n*][:CHANNel[*m*]]:OFFSet:WAVelength:VALue:WAVelength?**
syntax:	:CONFigure[n]][:CHANNel[m]]:OFFSet:WAVelength:VALue:WAVelength?<wsp><index>
affects:	N775xA and N776xA attenuators.
description:	Included for compatibility.
parameters:	none
response:	none
example:	

command:	**:CONFigure[*n*][:CHANNel[*m*]]:OFFSet:WAVelength:VALue:OFFSet?**
syntax:	:CONFigure[n]][:CHANNel[m]]:OFFSet:WAVelength:VALue:OFFSet?<wsp><index | wavelength [PM | NM | UM | MM| M],>
affects:	N775xA and N776xA attenuators.
description:	Included for compatibility.
parameters:	none

response:	none
example:	

command:	**:CONFigure[*n*][:CHANnel[*m*]]:OFFSet:WAVelength:VALue:PAIR?**
syntax:	:CONFigure[n]][:CHANnel[m]]:OFFSet:WAVelength:VALue:PAIR?<wsp><index | wavelength[PM | NM | UM | MM| M],>
affects:	N775xA and N776xA attenuators.
description:	Included for compatibility.
parameters:	none
response:	none
example:	

command:	**:CONFigure[*n*][:CHANnel[*m*]]:OFFSet:WAVelength:VALue:DELete**
syntax:	:CONFigure[*n*]][:CHANnel[*m*]]:OFFSet:WAVelength:VALue:DELete<wsp><index | wavelength[PM | NM | UM | MM| M],>
affects:	N775xA and N776xA attenuators.
description:	Included for compatibility. Use ":CONFigure[n][:CHANnel[m]]:OFFSet:WAVelength:VALue:DELete:ALL" on page 112.
parameters:	none
response:	*none*
example:	

Measurement Functions – The FETCh, INITiate, READ and SENSe Subsystems

These subsystems let you control measurement parameters for the power meter. Generally the parameter n refers to the optical port number. The Channel parameter m is not needed and only included for backward compatibility.

The FETCh subtree

command:	**:FETCh[*n*][:CHANnel[*m*]][:SCAlar]:POWer[:DC]?**
syntax:	:FETCh[*n*][:CHANnel[*m*]][:SCAlar]:POWer[:DC]?
affects:	MPPM Attenuator.
description:	Reads the current power meter value. It does not provide its own triggering and so must be used with either continuous software triggering (see ":INITiate[n][:CHANnel[m]]:CONTinuous?" on page 117) or a directly preceding immediate software trigger (see ":INITiate[n][:CHANnel[m]][:IMMediate]" on page 117). It returns the value the previous software trigger measured. Any subsequent FETCh command will return the same value, if there is no subsequent software trigger.
parameters:	none
response:	The current value as a **float** value in dBm,W or dB. If the reference state is absolute, units are dBm or W. If the reference state is relative, units are dB.
example:	fetc1:pow? ⟶ +6.73370400E-04<END>

command:	**:FETCh[*n*][:CHANnel[*m*]][:SCALar:]:POWer:ALL?**
syntax:	:FETCh[n][:CHANnel[m]][:SCALar]:POWer[::DC]:ALL?
affects:	MPPM Attenuator.
description:	Reads all current power meter values. It does not provide its own triggering and so must be used with either continuous software triggering (see ":INITiate[n][:CHANnel[m]]:CONTinuous?" on page 117) or a directly preceding immediate software trigger (see ":INITiate[n][:CHANnel[m]][:IMMediate]" on page 117). It returns the value the previous software trigger measured. Any subsequent FETCh command will return the same values, if there is no subsequent software trigger. The power meters must be running for this command to be effective.
parameters:	none

response:	4-byte Intel *float* values in a binary block in Intel byte order. The values are ordered by slot. See "Data Types" on page 20 for more information on Binary Blocks. Data values are always in Watt.
example:	fetc:pow:all? → interpreted as +1.33555600E-006 | +1.34789100E-006 | +1.37456900E-006<END>

command:	**:FETCh[*n*][:CHANnel[*m*]][:SCALar:]:POWer:ALL:CSV?**
syntax:	:FETCh[n][:CHANnel[m]][:SCALar]:POWer[::DC]:ALL:CSV?
affects:	MPPM Attenuator.
description:	Reads all current power meter values. It does not provide its own triggering and so must be used with either continuous software triggering (see ":INITiate[n][:CHANnel[m]]:CONTinuous?" on page 117) or a directly preceding immediate software trigger (see ":INITiate[n][:CHANnel[m]][:IMMediate]" on page 117). It returns the value the previous software trigger measured. Any subsequent FETCh command will return the same values, if there is no subsequent software trigger. The power meters must be running for this command to be effective.
parameters:	none
response:	string containing the power values from each available channel in a comma separated format. Data values are always in Watt.
example:	fetc:pow:all:CSV? → "+1.33555600E-06, +1.34789100E-06, +1.37456900E-06"<END>

command:	**:FETCh[*n*][:CHANnel[*m*]]:POWer:ALL:CONFig?**
syntax:	:FETCh[*n*][:CHANnel[*m*]][:SCALar]:POWer[:DC]:ALL:CONFig?
affects:	MPPM Attenuator.
description:	Returns the slot and channel numbers for all available power meter channels. Use this command to match returned power values to the appropriate slot and channel number.
parameters:	none
response:	A binary block (Intel byte order) consisting of 2-byte unsigned integer value pairs (so each pair has 4 bytes). The first member of the pair represents the the slot number, the second member of the pair represents the channel number.
example:	fetc:pow:all:conf? → interpreted as 1 | 1 | 2 | 1 | 3 | 1 | 4 | 1<END> This 16-byte block means that there are four powermeters present: Slot 1, Channel 1 Slot 2, Channel 1 Slot 3, Channel 1 Slot 4, Channel 1

The INITiate subtree

command:	**:INITiate[*n*][:CHANnel[*m*]][:IMMediate]**
syntax:	:INITiate[*n*][:CHANnel[*m*]][:IMMediate]
affects:	N774xA multiport and N775xA power meters.
description:	Initiates the software trigger system and completes one full trigger cycle, that is, one measurement is made.
parameters:	none
response:	none
example:	init

command:	**:INITiate[*n*][:CHANnel[*m*]]:CONTinuous**	
syntax:	:INITiate[*n*][:CHANnel[*m*]]:CONTinuous<wsp><boolean>	
affects:	N774xA multiport and N775xA power meters.	
description:	Sets the software trigger system to continuous measurement mode.	
parameters:	A *boolean* value:	0 or OFF: do not measure continuously 1 or ON: measure continuously
response:	none	
example:	init2:cont 1	

command:	**:INITiate[*n*][:CHANnel[*m*]]:CONTinuous?**	
syntax:	:INITiate[*n*][:CHANnel[*m*]]:CONTinuous?	
affects:	N774xA multiport and N775xA power meters.	
description:	Queries whether the software trigger system operates continuously or not	
parameters:	none	
response:	A *boolean* value:	0 or OFF: measurement is not continuous 1 or ON: measurement is continuous
example:	init2:cont? $\longrightarrow$ 1<END>	

The READ subtree

command:	**:READ[*n*][:CHANnel[*m*]][:SCALar:]:POWer:ALL?**
syntax:	:READ[n][:CHANnel[m]][:SCALar:]:POWer[:DC]:ALL?
affects	MPPM Attenuator..

description:	Reads all available power channels. It provides its own software triggering and does not need a triggering command.
parameters:	none
response:	4-byte Intel *float* values in a binary block in Intel byte order. The values are ordered by slot. See “Data Types" on page 20 for more information on Binary Blocks. Data values are always in Watt.
example:	read:pow:all? ⟶ interpreted as +1.33555600E-006|+1.34789100E-006|+1.37456900E-006<END>

command:	**:READ[*n*][:CHANnel[*m*]][:SCALar:]:POWer:ALL:CONFig?**
syntax:	:READ[*n*][:CHANnel[*m*]]:POWer[:DC]:ALL:CONFig?
affects	MPPM Attenuator..
description:	Returns the slot numbers for all available power meters. Use this command to match returned power values to the appropriate slot.
parameters:	none
response:	A binary block (Intel byte order) consisting of 2-byte unsigned integer value pairs (so each pair has 4 bytes). The first member of the pair represents the the slot number, the second member of the pair represents the channel number. The channel number is always 1.
example:	read1:pow:all:conf? ⟶ interpreted as 1|1|2|1|3|1|4|1<END> This 16-byte block means that there are four powermeters present

command:	**:READ[*n*][:CHANnel[*m*]][:SCALar]:POWer[:DC]?**
syntax:	:READ[*n*][:CHANnel[*m*]][:SCALar]:POWer[:DC]?
affects	MPPM Attenuator..
description:	Reads the current power meter value. It provides its own software triggering and does not need a triggering command. If the software trigger system operates continuously (see “:INITiate[n][:CHANnel[m]]:CONTinuous?” on page 117), this command is identical to “:FETCh[n][:CHANnel[m]][:SCAlar]:POWer[:DC]?” on page 115. If the software trigger system does not operate continuously, this command is identical to generating a software trigger (“:INITiate[n][:CHANnel[m]][:IMMediate]” on page 117) and then reading the power meter value. The power meter must be running for this command to be effective.
parameters:	none
response:	The current power meter reading as a *float* value in dBm, W or dB. If the reference state is absolute, units are dBm or W. If the reference state is relative, units are dB.
example:	read1:pow? ⟶ +1.33555600E-006<END>

The ROUTe subtree

command:	**:ROUTe[*n*][:CHANnel[*m*]]**	
syntax:	:ROUTe[*n*]:[CHANnel[*m*]]<wsp><channel_list>	
description:	Sets the channel route between two ports.	
NOTE	When you use switches with dependent connections (e.g. the 2x2 switch), it is possible that one route configuration automatically changes another connection!	
parameters:	n:	the slot number of the switch module
	m:	the switch channel within the selected switch module. e.g. for dual 1 x 2 module m = 1 for switch 1; m = 2 for switch 2
	channel_list	the route between left and right ports.
		channel_list format: [A....Z],[1....n]
response:	If an invalid route is selected the following error message is returned - "StatParamError"	
example:	rout3:chan1 A,2 (module in slot 3,channel 1, connect port A with port 2)	
affects:	All switch modules	

command:	**:ROUTe[*n*][:CHANnel[*m*]]?**	
syntax:	:ROUTe[*n*]:[CHANnel[*m*]]<wsp><channel_list>	
description:	Queries the current channel route of the switch for a specific module and switch channel.	
parameters:	n:	the slot number of the switch module
	m:	the switch channel within the selected switch module. Default value is 1.
response:	[A.....Z],[1.....n];[A.....Z],[1.....n] as a text string. "," separates input and output ports of a specific connection.	
	";" separates parallel connections (as used in 2x2 switch).	
example:	rout3:chan1? → A,1 simple 1xN switch	
	rout2:chan1? → A,2;B,1 (2x2 crossover switch in crossover config).	
affects:	All switch modules	

command:	**:ROUTe[*n*][:CHANnel[*m*]]:CONFig?**
syntax:	:ROUTe[*n*]:[CHANnel[*m*]]:CONFig?
description:	Queries the switch configuration of the instrument. For each channel, the minimum and maximum channel number of each port is given.
NOTE	
parameters:	none

response:	<j>,<k>;<l>,<m> as text string where: <j> is the first port character on the left <k> is the last port character on the left <l> is the minmimum port number on the right <m> is the maximum port number on the right
example:	rout2:conf? → A,B;1,2 (2 left and 2 right ports)
affects:	All switch modules

command:	**:ROUTe[*n*][:CHANnel[*m*]]:CONFig:ROUTe?**
syntax:	:ROUTe[*n*]:[CHANnel[*m*]]:CONFig:ROUTe?
description:	Queries the allowed switch routes of an instrument.
NOTE	
parameters:	none
response:	[A.....Z],[1.....n];[A.....Z],[1.....n].[A.....Z],[1.....n] as a text string. "," separates input and output ports of a single connection. ";" separates parallel connections "." separates possible switch states
example:	rout2:conf:rout? → A,1;B,2.A,2;B,1 2x2 x-over switcch: state 1: When A to 1 then B to 2nd connection (straight) state 2: When A to 2 then B to 1st connection (cross-over)
affects:	All switch modules

The SENSe subtree

command:	**:SENSe[*n*][:CHANnel[*m*]]:CORRection[:LOSS][:INPut][:MAGNitude]**
syntax:	:SENSe[*n*][:CHANnel[*m*]]:CORRection[:LOSS][:INPUT][:MAGNitude]<wsp><value>[DB|MDB]
affects:	N774xA multiport and N775xA power meters.
description:	Enters a calibration value for a power meter..
parameters:	The calibration factor as a *float* value If no unit type is specified, decibels (dB) is implied.
response:	none
example:	sens1:corr 10DB

command:	**:SENSe[*n*][:CHANnel[*m*]]:CORRection[:LOSS][:INPut][:MAGNitude]?**
syntax:	:SENSe[*n*][:CHANnel[*m*]]:CORRection[:LOSS][:INPUT][:MAGNitude]?
affects:	N774xA multiport and N775xA power meters.
description:	Returns the calibration factor for a power meter..
parameters:	none
response:	The calibration factor as a *float* value. Units are in dB, although no units are returned in the response message.
example:	sens1:corr? ⟶ +1.00000000E+000<END>

command:	**:SENSe[*n*][:CHANnel[*m*]]:CORRection:COLLect:ZERO**
syntax:	:SENSe[*n*][:CHANnel[*m*]]:CORRection:COLLect:ZERO
affects:	MPPM, Attenuator.
description:	Zeros the electrical offsets for a power meter. Note: While zeroing an attenuator's power monitor or a power meter in an N775*x*A attenuator instrument, all attenuators in the unit are stopped and all shutters are closed. This assures the best possible zeroing performance.
parameters:	none
response:	none
example:	sens1:corr:coll:zero

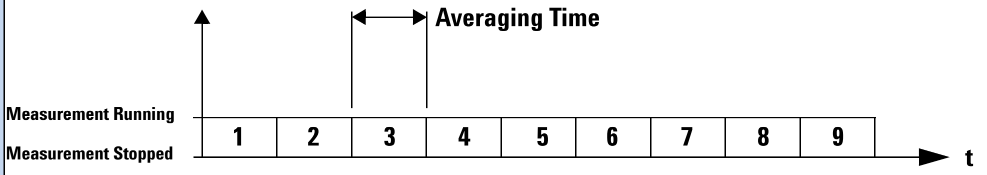
command:	**:SENSe[*n*][:CHANnel[*m*]]:CORRection:COLLect:ZERO?**	
syntax:	:SENSe[*n*][:CHANnel[*m*]]:CORREction:COLLect:ZERO?	
affects:	MPPM, Attenuator.	
description:	Returns the status of the most recent zero command.	
parameters:	none	
response:	0:	zero succeeded without errors.
	any other number:	remote zeroing failed (the number is the error code returned from the operation).
example:	sens1:corr:coll:zero? ⟶ 0<END>	

command:	**:SENSe[*n*][:CHANnel[*m*]]:CORRection:COLLect:ZERO:ALL**
syntax:	SENSe[*n*][:CHANnel[*m*]]:CORRection:COLLect:ZERO:ALL
affects:	MPPM, Attenuator.
description:	Zeros the electrical offsets for all installed power meters. Note: While zeroing an attenuator's power monitor or a power meter in an N775*x*A attenuator instrument, all attenuators in the unit are stopped and all shutters are closed. This assures the best possible zeroing performance.
parameters:	none
response:	none
example:	sens:chan:corr:coll:zero:all

command:	**:SENSe[n][:CHANnel[m]]:CORRection:COLLect:ZERO:ALL?**
syntax:	:SENSe[n][:CHANnel[m]]:CORRection:COLLect:ZERO:ALL?
affects:	MPPM, Attenuator.
description:	Returns the status of the most recent zero all command. The result is backed up in the nonvolatile RAM. Note: If a channel fails to zero, it continues to use the result of the last successful zeroing.
parameters:	none
response:	A hexadezimal integer value which represents the result for all channels. Each hexadecimal digit represents one channel . • 0: zero succeeded without errors. • Any other number: remote zeroing failed (the number is the error code returned from the operation).
example:	sens:chan:corr:coll:zero:all? ⟶ 272 272 decimal = 0x110. This means zeroing failed on channels 2 and 3. All other channels were successful. Alternatively the result can be found with the individual commands: sens1:chan:corr:coll:zero? ⟶ 0 sens2:chan:corr:coll:zero? ⟶ 1 sens3:chan:corr:coll:zero? ⟶ 1

NOTE Setting parameters for the logging function sets some parameters, including hidden parameters, for the stability and MinMax functions and vice versa. You must use the :SENSe[n][:CHANnel[m]]:FUNCtion:PARameter:LOGGing command to set parameters before you start a logging function using the :SENSe[n][:CHANnel[m]]:FUNCtion:STATe command.

command:	**:SENSe[*n*][:CHANnel[*m*]]:FUNCtion:PARameter:LOGGing**
syntax:	:SENSe[*n*][:CHANnel[*m*]]:FUNCtion:PARameter:LOGGing<wsp><data points>, <averaging time>[NS|US|MS|S]
affects:	N774xA multiport power meters.
description:	Sets the number of data points and the averaging time for the logging data acquisition function.

parameters:	Data Points: Data Points is the number of samples that are recorded before the logging mode is completed. Data Points is an **integer** value. Averaging time: Averaging time is a time value in seconds. There is no time delay between averaging time periods. Use "::SENSe[n][:CHANnel[m]]:FUNCtion:PARameter:STABility?" on page 126 if you want to use delayed measurement.  Averaging Time Measurement Running Measurement Stopped 1 2 3 4 5 6 7 8 9 t If you specify no units for the averaging time value in your command, seconds are used as the default. See ":SENSe[n][:CHANnel[m]]:FUNCtion:STATe" on page 131 for information on starting/stopping a data acquisition function. See ":SENSe[n][:CHANnel[m]]:FUNCtion:RESult?" on page 130 for information on accessing the results of a data acquisition function. See "Triggering and Power Measurements" on page 139 for information on how triggering affects data acquisition functions.
response:	none
example:	sens1:func:par:logg 64,1ms

NOTE

For Logging / Stability, set the Autogain to the same value for all channels. Send the Autogain commands before setting Average Time or configuring Logging / Stability.

Details are in the Application Note "Transient Optical Power Measurements with the N7744A and N7745A" http://cp.literature.agilent.com/litweb/pdf/5990-3710EN.pdf

command:	**:SENSe[n][:CHANnel[m]]:FUNCtion:PARameter:LOGGing?**
syntax:	:SENSe[n][:CHANnel[m]]:FUNCtion:PARameter:LOGGing?
affects:	N774xA multiport power meters.
description:	Returns the number of data points and the averaging time for the logging data acquisition function.
parameters:	none
response:	Returns the number of data points as an integer value and the averaging time, tavg, as a float value in seconds.
example:	sens1:func:par:logg? ¨ +64,+1.00000000E-001<END>

NOTE

Setting parameters for the MinMax function sets some parameters, including hidden parameters, for the stability and logging functions and vice versa. You must use the :SENSe[n][:CHANnel[m]]:FUNCtion:PARameter:MINMax command to set parameters before you start a MinMax function using the :SENSe[n][:CHANnel[m]]:FUNCtion:STATe command.

command:	**:SENSe[*n*][:CHANnel[*m*]]:FUNCtion:PARameter:MINMax**	
syntax:	:SENSe[*n*][:CHANnel[*m*]]:FUNCtion:PARameter:MINMax<wsp> CONTinous|WINDow|REFResh,<data points>	
affects:	N774xA multiport and N775xA power meters.	
description:	Sets the MinMax mode and the number of data points for the MinMax data acquisition function.	
parameters:	CONTinous:	continuous MinMax mode
	WINDow:	window MinMax mode WINDow mode has the same function as REFResh mode. It is included to ensure compatibility.
	REFResh:	refresh MinMax mode
	Data Points is the number of samples that are recorded in the memory buffer used by the WINDow and REFResh modes. Data Points is an **integer** value. See Chapter 3 of the Agilent N7744A / N7745A Multiport Optical Power Meter User's Guide, for more information on MinMax mode. See ":SENSe[n][:CHANnel[m]]:FUNCtion:STATe" on page 131 for information on starting/stopping a data acquisition function. See ":SENSe[n][:CHANnel[m]]:FUNCtion:RESult?" on page 130 for information on accessing the results of a data acquisition function. See "Triggering and Power Measurements" on page 139 for information on how triggering affects data acquisition functions.	
response:	none	
example:	sens1:func:par:minm WIND,10	

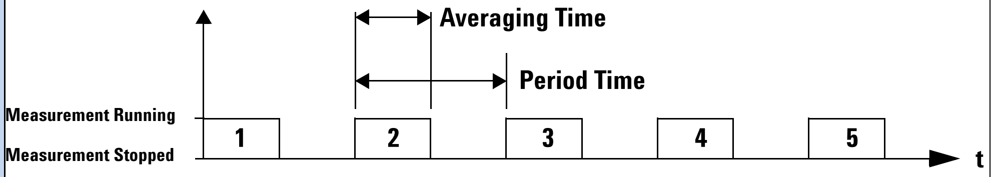
NOTE

The time between samples in MinMax mode is at least the averaging time, but not less than about 2 ms, depending on the interface command rate. On N775xA power meters it can be longer if one of the channels is set to an averate time longer than 2ms.

command:	**:SENSe[n][:CHANnel[m]]:FUNCtion:PARameter:MINMax?**
syntax:	:SENSe[n][:CHANnel[m]]:FUNCtion:PARameter:MINMax?
affects:	N774xA multiport and N775xA power meters.
description:	Returns the MinMax mode and the number of data points for the MinMax data acquisition function.
parameters:	none
response:	CONT: continuous MinMax mode WIND: window MinMax mode WINDow mode has the same function as REFResh mode. It is included to ensure compatibility. REFR: refresh MinMax mode The number of data points is returned as an integer value.
example:	sens1:func:par:minm? ¨ WIND,+10<END>

NOTE Setting parameters for the stability function sets some parameters, including hidden parameters, for the logging and MinMax functions and vice versa. You must use the :SENSe[n][:CHANnel[m]]:FUNCtion:PARameter:STABility command to set parameters before you start a stability function using the :SENSe[n][:CHANnel[m]]:FUNCtion:STATe command.

command:	**:SENSe[*n*][:CHANnel[*m*]]:FUNCtion:PARameter:STABility**
syntax:	:SENSe[*n*][:CHANnel[*m*]]:FUNCtion:PARameter:STABility<wsp> <total time>[NS|US|MS|S],<period time>[NS|US|MS|S],<averaging time>[NS|US|MS|S]
affects:	N774xA multiport power meters
description:	Sets the total time, period time, and averaging time for the stability data acquisition function.

parameters:	Total time: The total time from the start of stability mode until it is completed. Period time: A new measurement is started after the completion of every period time. Averaging time: A measurement is averaged over the averaging time.  Averaging Time Period Time Measurement Running Measurement Stopped 1 2 3 4 5 t The total time should be longer than the period time. The period time should be longer than the averaging time. The number of data points is equal to the total time divided by the period time. Total time, period time, and averaging time are time values in seconds. If you specify no units in your command, seconds are used as the default. See “:SENSe[n][:CHANnel[m]]:FUNCtion:STATe” on page 131 for information on starting/stopping a data acquisition function. See “:SENSe[n][:CHANnel[m]]:FUNCtion:RESult?” on page 130 for information on accessing the results of a data acquisition function. See “Triggering and Power Measurements" on page 139 for information on how triggering affects data acquisition functions.
response:	none
example:	sens1:func:par:stab 1s,0.1s,0.1s

NOTE

For Logging / Stability, set the Autogain to the same value for all channels. Send the Autogain commands before setting Average Time or configuring Logging / Stability.

Details are in the Application Note "Transient Optical Power Measurements with the N7744A and N7745A" http://cp.literature.agilent.com/litweb/pdf/5990-3710EN.pdf

command:	**::SENSe[n][:CHANnel[m]]:FUNCtion:PARameter:STABility?**
syntax:	:SENSe[n][:CHANnel[m]]:FUNCtion:PARameter:STABility?
affects:	N774xA multiport power meters.
description:	Returns the total time, period time, and averaging time for the stability data acquisition function.
parameters:	none
response:	Total time, delay time, and averaging time are float values in seconds.
example:	sens1:func:par:stab? ¨ +1.00000000E+000, +1.00000000E-001,+1.00000000E-001<END>

Using data buffers for simultaneous measurement and upload

Enhanced logging functionality is available using two data buffers for each port. The MPPM ensures there is no time lag between logging and data availability by using a data buffer with a capacity of 1M samples for each channel. You can read the most recent results out of one buffer while the next logging measurement is filling the other buffer.
You can use this to speed up applications with repeated logging operations. It is especially valuable for monitoring signals over extended periods to detect transient events.

To use the data buffers, set the logging function to perform a fixed or indefinite number of "LOOP"s (available from firmware versions 1.11 and later).

- When LOOP = 1, default logging behavior is selected, that is measurements are made and the results are written to the first buffer.
- If LOOP = 0, logging continues, writing the results into alternate buffers, until you stop it.
- When LOOP = *n*>1, then upon completion of a logging measurement, another is started and the results are written in rotation to the two buffers (first to buffer A, then buffer B, then buffer A, and so on).
  This is repeated until *n* logging measurements have been performed. For example, if *n*=2, both buffers are filled, and you can upload a total of 2M samples for the channel.

You can also use input triggers to begin individual loops.

The "Index" value is updated to indicate that data from each LOOP is available for readout.

See also:

:SENSe[n][:CHANnel[m]]:FUNCtion:RESult:INDex?

:SENSe[n][:CHANnel[m]]:FUNCtion:RESult:BUFA?

:SENSe[n][:CHANnel[m]]:FUNCtion:RESult:BUFB?

Example: Programming Streaming Data

NOTE

Details are in the Application Note "Transient Optical Power Measurements with the N7744A and N7745A"

http://cp.literature.agilent.com/litweb/pdf/5990-3710EN.pdf

1 Set Loop Parameter

SENS1:FUNC:LOOP 0	- For unlimited loops

2 ConfiTRIGn:INP MMEguration: for all ports n

SENSn:FUNC:STAT LOGG,STOP	- Cleaning up, in case the last run didn't finish
INITn:CONT 0	- Disable continuous triggering
SENSn:POW:GAIN:AUTO 0	- optional, disable Auto Gain if required
SENSn:POW:RANG xDBM	- Choose range
SENS:POW:UNIT 1	- Fastest upload with units Watts
TRIGn:INP MME	- Sets the incoming trigger response for multiple measurements
SENSn:FUNC:PAR:LOGG x,y us	- Enter logging samples and avg. time

3 Enable Logging: for all ports n

SENSn:FUNC:STAT LOGG,STAR	- Starts logging functions, which begin measurements synchronized with trigger.

4 Start Logging: for all ports

INIT1:IMM	- generate a trigger to start logging.

5 Check for Measurement Finished

SENSn:FUNC:RES:INDex?	- Index incremented when loop finishes; checking one port is normally sufficient.

6 Get Logging results for all ports

SENSn:FUNC:RES?	- Data in binary block form, "LSB" byte ordering

```
Stop Loops when finished
    SENS1:FUNC:LOOP 1
```

NOTE Use a MPPM Firmware Version > 1.16 for minimal streaming latency.

command:	**:SENSe[n][:CHANnel[m]]:FUNCtion:LOOP**
syntax:	:SENSe[n][:CHANnel[m]]:FUNCtion:LOOP<wsp><value>
affects:	N774xA multiport power meters
description:	Sets the number of logging loops
parameters:	Number of Loops, an integer value. • 0 = Endless Streaming • 1 = 1 (Default) • 2 = 2 (For 2 Million Points with Buffer A and B) .... • n
response:	none
example:	SENS1:FUNC:LOOP 0

command:	**:SENSe[n][:CHANnel[m]]:FUNCtion:LOOP?**
syntax:	:SENSe[n][:CHANnel[m]]:FUNCtion:LOOP?
affects:	N774xA multiport power meters
description:	Gets the number of logging loops. For details look at description and example in "Using data buffers for simultaneous measurement and upload" on page 127.
parameters:	none
response:	Number of loops: Number of loops is an integer value. • 0 = Endless Streaming • 1 = 1 (Default) • 2 = 2 (For 2 Million Points with Buffer A and B) .... • n
example:	:SENS1:FUNC:LOOP? → 0

command:	**:SENSe[n][:CHANnel[m]]:FUNCtion:RESult:INDex?**
syntax:	:SENSe[n][:CHANnel[m]]:FUNCtion:RESult:INDex?
affects:	N774xA multiport power meters.

description:	Gets the number of already finished logging loops. For details look at description and example in "Using data buffers for simultaneous measurement and upload" on page 127.
parameters:	none
response:	Number of loops: Number of loops is an integer value.
example:	sens1:func:res:index? $\longrightarrow$ 1

command:	**:SENSe[*n*][:CHANnel[*m*]]:FUNCtion:RESult?**
syntax:	:SENSe[*n*][:CHANnel[*m*]]:FUNCtion:RESult?
affects:	N774xA multiport and N775xA power meters.
description:	Returns the data array of the last data acquisition function.
parameters:	none
response:	The last data acquisition function's data array as a binary block. For Logging and Stability Data Acquisition functions, one measurement value is a 4-byte-long **float** in Intel byte order. For the MinMax Data Acquisition function, the query returns the minimum, maximum and current power values. See "Data Types" on page 20 for more information on Binary Blocks. See "How to Log Results" on page 149 for information on logging using VISA calls. There are some tips about how to use float format specifiers to convert the binary blocks into **float** values. If you use LabView or Agilent VEE, we recommend using the Agilent N7744A / N7745A VXI*plug&play* Instrument Driver to perform the Logging and Stability Data Acquisition functions.
example:	sens1:func:res? $\longrightarrow$ returns a data array for Logging and Stability Data Acquisition functions sens1:func:res? $\longrightarrow$ #255 Min: 7.24079E-04, Max: 7.24252E-04, Act: 7.24155E-04 returns the minimum, maximum and current power values for the MinMax Data Acquisition function

command:	**:SENSe[n][:CHANnel[m]]:FUNCtion:RESult:BUFA?**
syntax:	:SENSe[n][:CHANnel[m]]:FUNCtion:RESult:BUFA?
affects:	N774xA multiport power meters
description:	Returns the data array of the last data acquisition function in Buffer A. This works only for Logging and Stability Data Acquisition in the loop mode, not for the MinMax Data Acquisition. For details look at description and example in :"Using data buffers for simultaneous measurement and upload" on page 127
parameters:	none

response:	The last data acquisition function's data array as a binary block. For Logging and Stability Data Acquisition functions, one measurement value is a 4-byte-long float in Intel byte order. See "Data Types" on page 20 for more information on Binary Blocks. See "How to Log Results" on page 149 for information on logging using VISA calls. There are some tips about how to use float format specifiers to convert the binary blocks into float values.
example:	sens1:func:res:bufa? ⟶ #255

command:	**:SENSe[n][:CHANnel[m]]:FUNCtion:RESult:BUFB?**
syntax:	:SENSe[n][:CHANnel[m]]:FUNCtion:RESult:BUFB?
affects:	N774xA multiport power meters.
description:	Returns the data array of the last data acquisition function in Buffer B. This works only for Logging and Stability Data Acquisition in the loop mode, not for the MinMax Data Acquisition. For details look at description and example in :"Using data buffers for simultaneous measurement and upload" on page 127
parameters:	none
response:	The last data acquisition function's data array as a binary block. For Logging and Stability Data Acquisition functions, one measurement value is a 4-byte-long float in Intel byte order. See "Data Types" on page 20 for more information on Binary Blocks. See "How to Log Results" on page 149 for information on logging using VISA calls. There are some tips about how to use float format specifiers to convert the binary blocks into float values.
example:	sens1:func:res:bufb? ⟶ #255

command:	**:SENSe[*n*][:CHANnel[*m*]]:FUNCtion:STATe**
syntax:	:SENSe[*n*][:CHANnel[*m*]]:FUNCtion:STATe<wsp> LOGGing|STABility|MINMax,STOP|STARt
affects:	N774xA multiport and N775xA power meters
description:	Enables/Disables the logging, MinMax, or stability data acquisition function mode.

parameters:	LOGGing:	Logging data acquisition function (N774xA multiport power meters only)
	STABility:	Stability data acquisition function (N774xA multiport power meters only)
	MINMax:	MinMax data acquisition function
	STOP:	Stop data acquisition function
	STARt:	Start data acquisition function
	See ":SENSe[n][:CHANnel[m]]:FUNCtion:PARameter:LOGGing" on page 122 for more information on the logging data acquisition function.	
	Stop any functions on all channels before you try to set up a new function. Some parameters cannot be set until you stop the function.	
response:	none	
example:	sens1:func:stat logg,star	

command:	**:SENSe[*n*][:CHANnel[*m*]]:FUNCtion:STATe?**	
syntax:	:SENSe[*n*][:CHANnel[*m*]]:FUNCtion:STATe?	
affects:	N774xA multiport and N775xA power meters.	
description:	Returns the function mode and the status of the data acquisition function.	
parameters:	none	
response:	NONE	No function mode selected
	LOGGING_STABILITY	Logging or stability data acquisition function (N774xA multiport power meters only)
	MINMAX	MinMax data acquisition function
	PROGRESS	Data acquisition function is in progress
	COMPLETE	Data acquisition function is complete
example:	sens1:func:stat? → LOGGING_STABILITY,COMPLETE<END>	

command:	**:SENSe[*n*][:CHANnel[*m*]]:POWer:ATIMe**
syntax:	:SENSe[*n*][:CHANnel[*m*]]:POWer:ATIMe<wsp><averaging time>[NS|US|MS|S]
affects:	N774xA multiport and N775xA power meters.
description:	Sets the averaging time for the slot.
parameters:	The averaging time as a float value in seconds. If you specify no units in your command, seconds are used as the default. Note: for N775xA power meters the internal granularity of the averaging time is 2ms.
response:	none
example:	sens1:pow:atim 1s

command:	**:SENSe[*n*][:CHANnel[*m*]]:POWer:ATIMe?**
syntax:	:SENSe[*n*][:CHANnel[*m*]]:POWer:ATIMe?
affects:	N774xA multiport and N775xA power meters.
description:	Returns the averaging time for the slot.
parameters:	none

response:	The averaging time as a *float* value in seconds.
example:	sens1:pow:atim? ⟶ +1.00000000E+000<END>

command:	**:SENSe[*n*][:CHANnel[*m*]]:POWer:RANGe:AUTO**
syntax:	SENSe[*n*][:CHANnel[*m*]]:POWer:RANGe:AUTO <wsp><boolean>
affects:	N774xA multiport and N775xA power meters.
description:	Enables or disables automatic power ranging for the slot. If automatic power ranging is enabled, ranging is automatically determined by the slot. Otherwise, it must be set by the sensn:pow:rang command.
parameters:	A *boolean* value: 0 or OFF: automatic ranging disabled 1 or ON: automatic ranging enabled
response:	none
example:	sens1:pow:rang:auto 1

command:	**:SENSe[*n*][:CHANnel[*m*]]:POWer:RANGe:AUTO?**
syntax:	:SENSe[*n*][:CHANnel[*m*]]:POWer:RANGe:AUTO?
affects:	N774xA multiport and N775xA power meters.
description:	Returns whether automatic power ranging is being used by the slot.
parameters:	none
response:	A *boolean* value: 0: automatic ranging is not being used. 1: automatic ranging is being used.
example:	sens1:pow:rang:auto? ⟶ 1<END>

command:	**:SENSe[*n*][:CHANnel[*m*]]:POWer:RANGe[:UPPer]**
syntax:	:SENSe[*n*][:CHANnel[*m*]]:POWer:RANGe[:UPPer]<wsp><value>[DBM]
affects:	N774xA multiport and N775xA power meters.
description:	Sets the power range for the module. The range changes at 10 dBm intervals. The corresponding ranges for linear measurements (measurements in Watts) is given below: **Range** — Upper Linear Power Limit +10 dBm — 19.999 mW 0 dBm — 1999.9 µW –10 dBm — 199.99 µW –20 dBm — 19.999 µW –30 dBm — 1999.9 nW
parameters:	The range as a **float** value in dBm. The number is rounded to the closest multiple of 10, because the range changes at 10 dBm intervals. Units are in dBm.
response:	none
example:	sens1:pow:rang -20DBM

Range	Upper Linear Power Limit
+10 dBm	19.999 mW
0 dBm	1999.9 µW
–10 dBm	199.99 µW
–20 dBm	19.999 µW
–30 dBm	1999.9 nW

command:	**:SENSe[*n*][:CHANnel[*m*]]:POWer:RANGe[:UPPer]?**
syntax:	:SENSe[n][:CHANnel[m]]:POWer:RANGe[:UPPer]?
affects:	N774xA multiport and N775xA power meters.
description:	Returns the range setting for the slot.
parameters:	none
response:	The range setting as a $float$ value in dBm
example:	sens1:pow:rang? ⟶ -2.00000000E+001<END>

command:	**:SENSe[n][:CHANnel[m]]:POWer:GAIN:AUTO**
syntax:	:SENSe[n][:CHANnel[m]]:POWer:GAIN:AUTO<wsp><value>
affects:	N774xA multiport and N775xA power meters
description:	Set the Auto Gain.
parameters:	• 0 = Auto Gain Off. This is the position for best transient response. • 1 = Auto Gain On (Default) This is the Position for best dynamic. Auto gain only works for averaging times>10µs. For shorter averaging times, the auto gain is always disabled. The Auto Gain setting works also in the logging and stability modes, where it also increases dynamic or enhances transient response. **NOTE:** For Logging / Stability, set the Autogain to the same value for all channels. Send the Autogain commands before setting Average Time or configuring Logging / Stability. **NOTE:** Details are in the Application Note "Transient Optical Power Measurements with the N7744A and N7745A" http://cp.literature.agilent.com/litweb/pdf/5990-3710EN.pdf
response:	none
example:	sens1:pow:gain:auto 1

command:	**:SENSe[n][:CHANnel[m]]:POWer:GAIN:AUTO?**
syntax:	:SENSe[n][:CHANnel[m]]:POWer:GAIN:AUTO?
affects:	N774xA multiport and N775xA power meters
description:	Get the Auto Gain status.
parameters:	none
response:	• 0 = Auto Gain Off. This is the position for best transient response. • 1 = Auto Gain On (Default) This is the Position for best dynamic.
example:	sens1:pow:gain:auto? ⟶ 1

command:	**:SENSe[*n*][:CHANnel[*m*]]:POWer:REFerence**
syntax:	:SENSe[*n*][:CHANnel[*m*]]:POWer:REFerence<wsp> TOMODule | TOREF,<value>PW | NW | UW | MW | Watt | DBM | DB | MDB
affects:	N774xA multiport and N775xA power meters.
description:	Sets the sensor reference value.
parameters:	TOMODule: Sets the reference value in dB used if you choose measurement relative to another slot TOREF: Sets the reference value in Watts or dBm if you choose measurement relative to a constant reference value The reference as a **float** value. You must append a unit type • dB if you use TOMODule or • Watts or dBm if you use TOREF. The two reference values are completely independent. When you change the reference mode using the command ":SENSe[n][:CHANnel[m]]:POWer:REFerence:STATe:RATio" on page 136, the slot uses the last reference value entered for the selected reference mode.
response:	none
example:	sens1:pow:ref tomod,-40DB

command:	**:SENSe[*n*][:CHANnel[*m*]]:POWer:REFerence?**
syntax:	:SENSe[*n*][:CHANnel[*m*]]:POWer:REFerence?<wsp>TOMODule | TOREF
affects:	N774xA multiport and N775xA power meters.
description:	Returns the sensor reference value.
parameters:	TOMODule: Returns the reference value in dB used if you choose measurement relative to another slot TOREF: Returns the reference value in Watts or dBm if you choose measurement relative to a constant reference value
response:	The reference as a **float** value.
example:	sens1:pow:ref? toref ⟶ +1.00000000E-006<END>

command:	**:SENSe[*n*][:CHANnel[*m*]]:POWer:REFerence:DISPlay**
syntax:	:SENSe[*n*][:CHANnel[*m*]]:POWer:REFerence:DISPlay
affects:	N774xA multiport and N775xA power meters.
description:	Takes the input power level value as the reference value.
parameters:	none
response:	none
example:	sens1:pow:ref:disp

command:	**:SENSe[*n*][:CHANnel[*m*]]:POWer:REFerence:STATe**
syntax:	:SENSe[*n*][:CHANnel[*m*]]POWer:REFerence:STATe<wsp><boolean>

affects:	N774xA multiport and N775xA power meters.
description:	Sets the measurement units to relative or absolute units.
parameters:	A *boolean* value: 0 or OFF: absolute 1 or ON: relative
response:	none
example:	sens1:pow:ref:stat 1

command:	**:SENSe[*n*][:CHANnel[*m*]]:POWer:REFerence:STATe?**
syntax:	:SENSe[*n*][:CHANnel[*m*]]POWer:REFerence:STATe?
affects:	N774xA multiport and N775xA power meters.
description:	Inquires whether the current measurement units are relative (dB) or absolute (Watts or dBm).
parameters:	none
response:	A *boolean* value: 0: absolute 1: relative
example:	sens1:pow:ref:stat? ⟶ 1<END>

command:	**:SENSe[*n*][:CHANnel[*m*]]:POWer:REFerence:STATe:RATio**
syntax:	:SENSe[*n*][:CHANnel[*m*]]POWer:REFerence:STATe:RATio<wsp> <slot number>|255|TOREF,<channel number>
affects:	N774xA multiport and N775xA power meters.
description:	Selects the reference for the slot.
parameters:	slot number: an **integer** value representing the slot number you want to reference 255 or TOREF: results are displayed relative to an absolute reference channel number: an **integer** value representing the channel number you want to reference. For the Multiport Optical Power Meter the channel number is always 1. If you want to reference another power sensor, use an integer value corresponding to the slot for the first parameter and an integer value of 1 for the channel value. If you want to use an absolute reference, use TOREF as the first parameter and any integer value as the second parameter.
response:	none
examples:	sens1:pow:ref:stat:rat 2,1 — References slot 2. channel 1 sens1:pow:ref:stat:rat TOREF,1 — References an absolute reference

command:	**:SENSe[*n*][:CHANnel[*m*]]:POWer:REFerence:STATe:RATio?**
syntax:	:SENSe[*n*][:CHANnel[*m*]]POWer:REFerence:STATe:RATio?
affects:	N774xA multiport and N775xA power meters.
description:	Returns the reference setting for the slot.
parameters:	none

response:	results are displayed relative to an absolute reference or to the current power reading from another slot.	
examples:	sens1:pow:ref:stat:rat? ⟶ +255,+0<END>	results are displayed relative to an absolute reference
	sens1:pow:ref:stat:rat? ⟶ +2,+1<END>	results are displayed relative to slot 2

command:	**:SENSe[*n*][:CHANnel[*m*]]:POWer:UNIT**	
syntax:	:SENSe[*n*][:CHANnel[*m*]]:POWer:UNIT<wsp>DBM|0|Watt|1	
affects:	N774xA multiport and N775xA power meters.	
description:	Sets the sensor power unit	
parameters:	An *integer* value:	0: dBm 1: Watt
	or DBM or Watt	
response:	none	
example:	sens1:pow:unit 1	

command:	**:SENSe[*n*][:CHANnel[*m*]]:POWer:UNIT?**	
syntax:	:SENSe[*n*][:CHANnel[*m*]]:POWer:UNIT?	
affects:	N774xA multiport and N775xA power meters.	
description:	Queries the current sensor power unit	
parameters:	none	
response:	An *integer* value:	0: Current power units are dBm. 1: Current power units are Watts.
example:	sens1:pow:unit? ⟶ +1<END>	

command:	**:SENSe[:CHANnel[*m*]]:POWer:UNIT:ALL:CSV?**
syntax:	:SENSe[:CHANnel[*m*]]:POWer:UNIT:ALL:CSV?
affects:	All instruments
description:	Queries the power unit for all ports of the instrument, returned in csv data format.
parameters:	none
response:	*Integer* values, separated by commas: • 0: Current power units are dBm. • 1: Current power units are Watts.
example:	sens:pow:unit:all:csv? ⟶ 1,1,1,1,1,1,1,1,<END>

command:	**:SENSe[:CHANnel[*m*]]:POWer:WAVelength**
syntax:	:SENSe[:CHANnel[*m*]]:POWer:WAVelength<wsp><value>|MIN|MAX|DEF [PM|NM|UM|MM|M]

affects:	N774xA multiport and N775xA power meters.	
description:	Sets the sensor wavelength.	
parameters:	The wavelength as a *float* value in meters.	
	Also allowed are:	MIN: minimum programmable value MAX: maximum programmable value DEF: This is not the preset (*RST) default value but is half the sum of, the minimum programmable value and the maximum programmable value
response:	none	
example:	sens1:pow:wav 1550nm	

command:	**:SENSe[*n*][:CHANnel[*m*]]:POWer:WAVelength?**	
syntax:	:SENSe[*n*][:CHANnel[*m*]]:POWer:WAVelength?[<wsp>MIN|MAX|DEF]	
affects:	N774xA multiport and N775xA power meters.	
description:	Inquires the current sensor wavelength.	
parameters:	none	
	Also allowed are:	MIN: minimum programmable value MAX: maximum programmable value DEF: This is not the preset (*RST) default value but is half the sum of, the minimum programmable value and the maximum programmable value
response:	The wavelength as a *float* value in meters.	
example	sens1:pow:wav? $\longrightarrow$ +1.55000000E-006<END>	

Triggering - The TRIGger Subsystem

The TRIGger Subsystem allows you to configure how the instrument reacts to incoming or outgoing triggers.

Table 1 Triggering and Power Measurements

Hardware Triggering	Software Triggering		Data Acquisition Functions sens:func:stat	
trig:inp	init:imm	init:cont	MINMax	LOGGing | STABility
IGNore	One power measurement is performed.	Automatically performs power measurements.		Automatically performs power measurements until the function is finished.
SMEasure	Every hardware trigger starts a new power measurement.			Every hardware trigger starts a new power measurement until the function is finished.
CMEasure				The first hardware trigger starts the function. Subsequent power measurements are automatically performed until the function is finished.
MMEeasure				Similar to CME, the first hardware trigger starts the function and if LOOP is not 1, then multiple functions will be performed without waiting for further triggers.

Table 2 Generating Output Triggers from Power Measurements

Hardware Triggering	Software Triggering		Data Acquisition Functions sens:func:stat	
trig:outp	init:imm	init:cont	MINMax	LOGGing | STABility
DISabled	An output trigger will never be generated.			
AVGover	An output trigger is generated for every new power measurement when the averaging time period finishes.			Applies for all subsequent data acquisition functions.
MEASure	An output trigger is generated for every new power measurement when the averaging time period begins.			Applies for all subsequent data acquisition functions.

command:	**:TRIGger[*n*][:CHANnel[*m*]]:INPut**
syntax:	:TRIGger[*n*][:CHANnel[*m*]]:INPut<wsp><trigger response>
affects:	MPPM, Attenuator.
description:	Sets the incoming trigger response and arms the slot.
parameters:	IGNore: Ignore incoming trigger. SMEasure: Start a single measurement. If a measurement function is active, see ":SENSe[n][:CHANnel[m]]:FUNCtion:STATe" on page 131, one sample is performed and the result is stored in the data array, see ":SENSe[n][:CHANnel[m]]:FUNCtion:RESult?" on page 130. CMEasure: Start a complete measurement. If a measurement function is active, see":SENSe[n][:CHANnel[m]]:FUNCtion:STATe" on page 131, a complete measurement function is performed. MMeasure Starts multiple complete measurements. Similar to CME, if a measurement function is active and the LOOP parameter is not 1, see ":SENSe[n][:CHANnel[m]]:FUNCtion:LOOP" on page 129", then multiple complete measurement functions are performed without waiting for further triggers. If LOOP is 1, then MME is identical to CME. For the N775*x* and N776*x* Attenuators, this command has no effect as init[n]:cont always stays at 1 to ensure the power control works. For the N775*x* power meters, the SME and CME have the same effect and work without arming a function. For the N774*x* Multiport power meters you must prearm a measurement function before an action can be triggered: First, set the incoming trigger response. Then: prearm a measurement function using ":SENSe[n][:CHANnel[m]]:FUNCtion:STATe" on page 131. NOTE: If a trigger signal arrives at the Input Trigger Connector at the same time that the :SENSe[n][:CHANnel[m]]:FUNCtion:STATe command is executed, the first measurement value is invalid. You should always discard the first measurement value in this case. The slot performs the appropriate action when it is triggered.
response:	none
example:	trig1:inp ign If you use the VXIplug&play Instrument Driver, you can trigger power measurements.

command:	**:TRIGger[*n*][:CHANnel[*m*]]:INPut?**
syntax:	:TRIGger[*n*][:CHANnel[*m*]]:INPut?
affects:	MPPM, Attenuator.
description:	Returns the incoming trigger response.
parameters:	none

response:	IGNore: Ignore incoming trigger. SMEasure: Start a single measurement. If a measurement function is active, see ":SENSe[n][:CHANnel[m]]:FUNCtion:STATe" on page 131, one sample is performed and the result is stored in the data array, see ":SENSe[n][:CHANnel[m]]:FUNCtion:RESult?" on page 130. CMEasure: Start a complete measurement. If a measurement function is active, see ":SENSe[n][:CHANnel[m]]:FUNCtion:STATe" on page 131, a complete measurement function is performed. MMeasure Starts multiple complete measurements. Similar to CME, if a measurement function is active and the LOOP parameter is not 1, see ":SENSe[n][:CHANnel[m]]:FUNCtion:LOOP" on page 129", then multiple complete measurement functions are performed without waiting for further triggers. If LOOP is 1, then MME is identical to CME.
example:	trig1:inp? —> ign<END>

command:	**:TRIGger[*n*][:CHANnel[*m*]]:OUTPut**
syntax:	:TRIGger[*n*][:CHANnel[*m*]]:OUTPut
affects:	MPPM, Attenuator.
description:	Specifies when an output trigger is generated and arms the slot. **NOTE:** The BNC output connector is controlled by the lowest-number channel with enabled trigger out.
parameters:	DISabled: Never. AVGover: When averaging time period finishes. MEASure: When averaging time period begins.
response:	none
example:	trig1:outp dis

command:	**:TRIGger[*n*][:CHANnel[*m*]]:OUTPut?**
syntax:	:TRIGger[*n*][:CHANnel[*m*]]:OUTPut?
affects:	MPPM, Attenuator.
description:	Returns the condition that causes an output trigger.
parameters:	none
response:	DISabled: Never. AVGover: When averaging time period finishes. MEASure: When averaging time period begins.
example:	trig1:outp? —> dis<END>

command:	**:TRIGger:CONFiguration**
syntax:	:TRIGger:CONFiguration<wsp><triggering mode>
affects:	All instruments.
description:	Sets the hardware trigger configuration with regard to Output and Input Trigger Connectors.

parameters:	0 or DISabled:	Trigger connectors are disabled.
	1 or DEFault:	The Input Trigger Connector is activated, the incoming trigger response for each slot ":TRIGger[n][:CHANnel[m]]:INPut" on page 140 determines how each slot responds to an incoming trigger, all slot events (see ":TRIGger[n][:CHANnel[m]]:OUTPut" on page 141) can trigger the Output Trigger Connector.
	2 or PASSthrough:	The same as DEFault but a trigger at the Input Trigger Connector generates a trigger at the Output Trigger Connector automatically.
	3 or LOOPback:	The same as PASSthrough. This is included for compatibility reasons.
response:	none	
example:	trig:conf dis	

command:	**:TRIGger:CONFiguration?**	
syntax:	:TRIGger:CONFiguration?	
affects:	All instruments.	
description:	Returns the hardware trigger configuration.	
parameters:	none	
response:	DIS:	Trigger connectors are disabled.
	DEF:	The Input Trigger Connector is activated, the incoming trigger response for each slot ":TRIGger[n][:CHANnel[m]]:INPut" on page 140 determines how each slot responds to an incoming trigger, all slot events (see ":TRIGger[n][:CHANnel[m]]:OUTPut" on page 141) can trigger the Output Trigger Connector.
	PASS:	The same as DEFault but a trigger at the Input Trigger Connector generates a trigger at the Output Trigger Connector automatically.
	LOOP:	The same as PASSthrough. This is included for compatibility reasons.
example:	trig:conf? $\longrightarrow$ DEF<END>	

5
VISA Programming Examples

These programming examples are implemented using MS Developer Studio. Regardless of the programming environment you use, keep the following in mind:

- The resultant application is a "console application"
- Make sure the header files visa.h and visatype.h are included.
- Make sure the library path includes visa32.lib
- Ensure that the PATH environment variable allows loading visa32.dll.

The programming examples do not cover the full command set for the instruments. They are intended only as an introduction, how to program the instrument using VISA library calls.

The VISA calls used, are explained in detail in the VISA User's Guide.

NOTE

Never use VISA calls and the Agilent N7744A / N7745A VXI*plug&play* Instrument Driver in the same program.

TIP: Additional programming examples are provided on the Support Disk CD-ROM N7744-90CD1

How to Use VISA Calls

The following example demonstrates how to communicate using VISA calls. Also, the use of instrument identification commands is demonstrated.

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

/* This function checks and displays errors, using the error query of the instrument;
Call this function after every command to make sure your commands are correct */

void checkError(ViSession session, ViStatus err_status )
 {
  ViStatus error;
  ViChar errMsg[256];
    /* queries what kind of error occurred */
    error = viQueryf(session,"%s\n","%t","SYST:ERR?",errMsg);
    /*if this command times out, a system error is probable;
     check the GPIB bus communication */
    if (error == VI_ERROR_TMO)
     {
     printf("System Error!\n") ;
     exit(1);
     }
    else
     {
     /* display the error number and the error message */
     if(errMsg[0] != '+')
     printf("error:%ld --> %s\n", err_status,errMsg) ;
     }

 }

void main (void)
 {
 ViStatus     errStatus;    /*return error code from visa call */
 ViSession    defaultRM;    /*default visa resource manager variable*/
 ViSession    vi;           /*current session handle */
 ViChar       replyBuf[256]; /*buffer holding answers from the instrument*/
 ViChar       c;
  /* Initialize visa resource manger */
       errStatus = viOpenDefaultRM (&defaultRM);
  if(errStatus < VI_SUCCESS)
   { printf("Failed to open VISA Resource manager\n");
    exit(errStatus);
   }

  /* Open session to GPIB device at address 20; the VI_NULL parameters 3,4
    are mandatory and not used for VISA 1.0*/
  errStatus = viOpen (defaultRM, "GPIB::20::INSTR", VI_NULL,VI_NULL,&vi);
  if(errStatus < VI_SUCCESS)
    { printf("Failed to open instrument\n");
      exit(errStatus);
    }

  /* set timeout to 20 sec; this should work for all commands except for zeroing or READ commands with
averaging times greater than the timeout */
  errStatus = viSetAttribute(vi,VI_ATTR_TMO_VALUE,20000);
  checkError(vi,errStatus);

  /* get the identification string of the instrument mainframe*/
  errStatus = viQueryf(vi,"%s\n","%t","*IDN?",replyBuf);
  if(errStatus < VI_SUCCESS)
    { checkError(vi,errStatus); }
  else printf("%s",replyBuf);

  /* identify the installed modules */
  errStatus = viQueryf(vi,"%s\n","%t","*OPT?",replyBuf);
  if(errStatus < VI_SUCCESS)
    { checkError(vi,errStatus); }
  else printf("%s",replyBuf);

  /* get information about the available options of a slot */
  errStatus = viQueryf(vi,"%s","%t","SLOT1:OPT?\n",replyBuf);
  if(errStatus < VI_SUCCESS)
    { checkError(vi,errStatus); }
  else printf("%s",replyBuf);


  /*loop, until a key is pressed */
  while(!scanf("%c",&c));
  /*close the session */
  viClose(vi);
 }
```

How to Measure Power using FETCh and READ

The example shows the difference between a "FETCh" and a "READ" command.

Install a power meter in Slot 1, before executing this example.

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <visa.h>

/* function prototypes for this examples */

/* function for a simple error handling explained in example 1 */
void checkError(ViSession session, ViStatus err_status );

void main (void)
 {
 ViStatus      errStatus;    /* returned error code from visa call */
 ViSession     defaultRM;    /* default visa resource manager variable */
 ViSession     vi;           /* current session handle */
 ViChar        replyBuf[256]; /* buffer holding answers of the instrument*/
 ViChar        compBuf[256];  /* buffer used for comparsion */
 ViChar        c;            /* used in the keyboard wait loop */
 ViReal64      averagingTime; /* averaging time */
 ViInt32       i;            /* loop counter */

  errStatus = viOpenDefaultRM (&defaultRM);
  if(errStatus < VI_SUCCESS)
   {
    printf("Failed to open VISA Resource manager\n");
    exit(errStatus);
   }

  errStatus = viOpen (defaultRM, "GPIB::20::INSTR", VI_NULL,VI_NULL,&vi);
  if(errStatus < VI_SUCCESS)
   {
    printf("Failed to open instrument\n");
    exit(errStatus);
   }
  /*set timeout to 20 sec; this should work for all commands
   except zeroing */
  errStatus = viSetAttribute(vi,VI_ATTR_TMO_VALUE,20000);
  if (errStatus < VI_SUCCESS) checkError(vi,errStatus);

  /* make sure that the reference is not used */
  errStatus = viPrintf(vi,"SENS1:CHAN1:POW:REF:STATE 0\n");
  if (errStatus < VI_SUCCESS) checkError(vi,errStatus);
```

```
   /* clear the error queue */
   errStatus = viPrintf(vi,"*CLS\n");
   if (errStatus < VI_SUCCESS) checkError(vi,errStatus);

   /* turn auto range on */
   errStatus = viPrintf(vi,"SENS1:CHAN1:POW:RANGE:AUTO 1\n");
   if (errStatus < VI_SUCCESS) checkError(vi,errStatus);

   /* change the power unit to watt */
   errStatus = viPrintf(vi,"SENS1:CHAN1:POW:UNIT W\n");
   if (errStatus < VI_SUCCESS) checkError(vi,errStatus);

   /*set the averaging time for measuring to 0.5s*/
   averagingTime = 0.5;

   errStatus = viPrintf(vi,"SENS1:CHAN1:POW:ATIME %f\n",averagingTime);
   if (errStatus < VI_SUCCESS) checkError(vi,errStatus);

   /* turn continous measuring off */
   errStatus = viPrintf(vi,"INIT1:CHAN1:CONT 0\n");
   if (errStatus < VI_SUCCESS) checkError(vi,errStatus);

     /* trigger a measurement */
   errStatus = viPrintf(vi,"INIT1:CHAN1:IMM\n");
   if (errStatus < VI_SUCCESS) checkError(vi,errStatus);

   /* read 10 values and display the result; */
   for (i = 0; i < 10; i++)
    {
  /* Now because an averaged value is available, the value will be fetched */
     errStatus = viQueryf(vi,"%s","%s","FETCH1:CHAN1:POW?\n",replyBuf);
     if (errStatus < VI_SUCCESS) checkError(vi,errStatus);
  /* two consecutive values are compared; if they are equal it will be marked; because no evaluation is triggered,
all values will be the same */
     if(i)
      { if(!strcmp(compBuf,replyBuf))
        { printf("Same:%s\n",replyBuf); }
      else printf("New:%s\n",replyBuf);
      }
     else printf("First:%s\n",replyBuf);
     strcpy(compBuf,replyBuf);
    }
   /* now the read command is used in the same manner to demonstrate the difference between fetch and read
*/

    /* read also 10 values, compare them and display the result; */
    for (i = 0; i < 10; i++)
    {
    /* In comparision to the "FETCH" command, the "READ" command implies        triggering a measurement.
Make sure the timeout set is greater than the adjusted averaging time, so that the READ command will not time
out; */

     /* send the read command */
```

```
        errStatus = viQueryf(vi,"READ1:CHAN1:POW?\n","%t",replyBuf);
        checkError(vi,errStatus);

         if(i)
          {
          if(!strcmp(compBuf,replyBuf))  printf("Same:%s",replyBuf);
          else printf("New  :%s",replyBuf);
          }
         else printf("\nFirst:%s",replyBuf);
         /*copy new value to compare buffer*/
         strcpy(compBuf,replyBuf);
       }
     /* loop, until a key is pressed */
     while(!scanf("%c",&c));

     checkError(vi,errStatus);
     /* close the session */
     viClose(vi);
   }

   void checkError(ViSession session, ViStatus err_status )
    { ViStatus error;
     ViChar errMsg[256];
        error = viQueryf(session,"SYST:ERR?\n","%t",errMsg);
        if (error == VI_ERROR_TMO)
         {
         printf("System Error!\n") ;
         exit(1);
         }
        else
         {
          /* only errors should be displayed */
          if(errMsg[0] != '+')
          printf("error:%ld --> %s\n", err_status,errMsg) ;
         }
    }
```

How to Log Results

This example demonstrates how to use logging functions.

Install a Power Sensor in Slot 1, before executing this example.

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <visa.h>

#define MAX_LOG_VALUES 4000 /* max number of values the instrument is able to log */
#define HEADER_SIZE    7    /* includes 6 bytes header and 1 CR */

/* function prototypes for this examples/*

/* function for a simple error handling explained in example 1 */
void      checkError(ViStatus session, ViStatus err_status );

/* initialize the visa interface */
ViStatus  InitVisa ( ViSession *iHandle);

/*globals*/
static unsigned char logBuffer[MAX_LOG_VALUES * sizeof(ViReal64) + HEADER_SIZE];
/*array for the float results */
static ViReal32 logResults[MAX_LOG_VALUES];

void main (void)
  {

   ViStatus     errStatus;      /* returned error code from visa call */
   ViSession    vi;             /* current session handle */
   ViChar       replyBuf[256];  /* buffer holding answers from the
instrument */
   ViChar       c;              /* used in the keyboard wait loop */
   ViInt32      slot;           /* slot number where the power meter is plugged */
   ViInt32      chan;           /* channel to be logged */
   ViInt32      i;              /* loop counter */
   ViInt32      noOfValues;     /* number of values to be logged*/
   ViReal64     averagingTime;  /* aveaging time used in a logging cycle */
   ViPChar      replySubStr;    /* pointer to a substring of the instruments reply */
   ViInt32      noOfDigits;     /*number of digits, specifing the amount data
 to be read */
   ViUInt32     retCnt;         /* returns the number of bytes read calling viRead */

    errStatus = InitVisa(&vi);

    if(errStatus < VI_SUCCESS)
      {
      exit(errStatus);
      }

    /* clear instrument error queue */
    errStatus = viPrintf(vi,"*CLS\n");
    checkError(vi,errStatus);

    /* turn auto range on */
    errStatus = viPrintf(vi,"SENS1:CHAN1:POW:RANGE:AUTO 1\n");
    checkError(vi,errStatus);

    /* send the command sequence for continuous logging */
    slot = 1;
    chan = 1;
    noOfValues = 100;     /* log 100 values */
    averagingTime = 0.02; /* set averaging time to 20ms */
    viPrintf(vi,"SENS%1d:CHAN%1d:FUNC:PAR:LOGG %d,%f\n",
             slot,
             chan,
             noOfValues,
             averagingTime);
               checkError(vi,errStatus);

    /* start logging */
        viPrintf(vi,"SENS%1d:CHAN%1d:FUNC:STAT LOGG,START\n",slot,chan);
               checkError(vi,errStatus);
    /* to display the results, logging should be completed */
    /* the instrument has to be polled about the progress of the logging */
    do
      {
        errStatus = viQueryf(vi,"SENS%1d:CHAN%1d:FUNC:STATE?\n","%t",slot,chan,replyBuf);
        /* if an error occurs break the loop */
        if (errStatus < VI_SUCCESS)
          {
          checkError(vi,errStatus);
          break;
          }

        /* find the substring "COMPLETE" in the reply of the instrument */

        replySubStr = replyBuf;
                while(*replySubStr)
          {
            if(!strncmp(replySubStr,"COMPLETE",strlen("COMPLETE"))) break;
            replySubStr ++;
          }
      }while (!*replySubStr); /*substring "COMPLETE" not found */
                            /*continue polling */

    /* The instrument returns the logging result in the following format:     #xyyyffff...; the first digits after the
hash denotes the number of ascii digits following (y) ; y specifies the number of binary data following; "ffff"
represent the 32Bit floats as log result. */
    /* get the result */
    errStatus =  viPrintf(vi,"SENS%1d:CHAN%1d:FUNC:RES?\n",slot,chan);
  /* only query an error, if there is one, else the query will be interrupted ! */
  if(errStatus < VI_SUCCESS)checkError(vi,errStatus);

  /* read the binary data */
  errStatus = viRead(vi, logBuffer, MAX_LOG_VALUES * sizeof(ViReal32) + HEADER_SIZE, &retCnt);
  checkError(vi,errStatus);

  if(logBuffer[0] != '#')
   {
   printf("invalid format returned from logging\n");
   exit(1);
   }
  else
   {
   noOfDigits = logBuffer[1] -'0';
   memcpy( logResults, &logBuffer[2 + noOfDigits ],
          MAX_LOG_VALUES * sizeof(ViReal32));
   }

  /* stop logging */
  viPrintf(vi,"SENS%1d:CHAN%1d:FUNC:STAT LOGG,STOP\n",slot,chan);
  checkError(vi,errStatus);

  /* display the values using %g, a float format specifier, you may also use %e or %f */
  for ( i = 0; i < noOfValues; i++)
  printf("\t%g\n",logResults[i]);

  /* loop, until a key is pressed */
  while(!scanf("%c",&c));

  /* close the session */
  viClose(vi);

 }

void checkError(ViStatus session, ViStatus err_status )
 {
  ViStatus error;
  ViChar errMsg[256];
     error = viQueryf(session,"SYST:ERR?\n","%t",errMsg);
     if (error == VI_ERROR_TMO)
       {
       printf("System Error!\n") ;
       exit(1);
       }
     else
       {
        /* only errors should be displayed */
        if(errMsg[0] != '+')
        {
        printf("error:%ld --> %s\n", err_status,errMsg) ;
        if
```

```
          ((!strncmp(errMsg,
          "-303,\"Module slot empty or slot / channel invalid\"",
          strlen("-303,\"Module slot empty or slot / channel invalid\"")))
          ||
          (!strncmp(errMsg,
          "-301,\"Module doesn't support this command (StatCmdUnknown)\"",
          strlen(
          "-301,\"Module doesn't support this command (StatCmdUnknown)\""))))
           {
             printf("No power meter in slot 1 so exiting\n\n");
         exit(1);
            }
          }
         }
     }

    ViStatus InitVisa ( ViSession *iHandle)
     {
       ViStatus    errStatus;     /* returned error code from visa call */
       ViSession   defaultRM;     /* default visa resource manager variable */

       /* First get initialized the visa library (see example 1) */
       errStatus = viOpenDefaultRM (&defaultRM);
       if (errStatus < VI_SUCCESS)
           printf("Failed to open VISA Resource manager\n");

       /* Open session to GPIB device at address 20; */
       errStatus = viOpen (defaultRM, "GPIB::20::INSTR",
                     VI_NULL,VI_NULL,iHandle);
       if (errStatus < VI_SUCCESS)
           printf("Failed to open instrument\n");

       return errStatus;
     }
```

6
Agilent 816x VXI*plug&play* Instrument Driver

NOTE

The N7744A/45A Multiport Optical Power Meter and the N7751A/52A/61A/62A/64A/66A/68A are supported by the standard 816x Plug & Play Instrument Driver version 4.3 and higher.

The N7744A/45A Multiport Optical Power Meter are also supported by IVI-COM and IVI-C drivers. These are available for download at http://www.agilent.com/find/ivi-com

NOTE

The N773xA Optical Switches are supported by the standard 816x Plug & Play Instrument Driver version 4.4 and higher.

This chapter gives you extra information about installing and getting started with the Agilent 816x VXI*plug&play* instrument driver.

There are details about opening and closing an instrument session, data types and constants used, error handling, and the programming environments supported.

Installing the Agilent 816x Instrument Driver

NOTE The N7744A/45A Multiport Optical Power Meter and the N7751A/52A/61A/62A/64A/66A/68 are supported by the standard 816x Plug & Play Instrument Driver version 4.3 and higher.

NOTE The N773xA Optical Switches are supported by the standard 816x Plug & Play Instrument Driver version 4.4 and higher.

The Agilent 816x VXI*plug&play* Instrument Driver comes as a self-extracting archive with an installation wizard. The installation wizard extracts all the files to preset destinations, asking you appropriate questions as it does so.

You install the driver by running the executable hp816x.exe.

1 Run hp816x.exe,

   The welcome screen for the InstallShield Wizard used to install the Agilent 816x VXI*plug&play* Instrument Driver is displayed.

2 Press Next> to continue.

   Specify the folder to which files will be saved.

3 Press Next> to continue.

   Files are copied and extracted.

   If necessary, a dialog requests your premission to overwrite existing files.

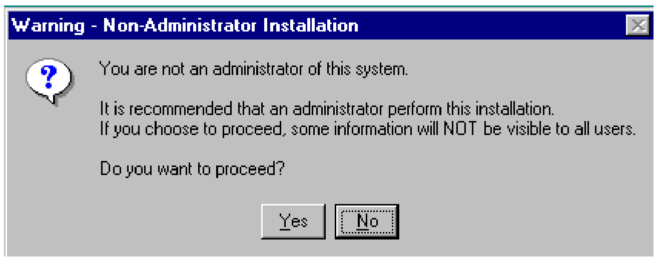
   The vesrion number of the instrument driver is displayed.

   You may now elect to skip installation at this PC. Copy the extracted disk images to floppy, and use them to install the instrument driver at another PC.

4 Press OK> to continue.

   If you are not an administrator, you see a VXI*plug&play* window, and a message telling you that if you proceed with the installation, some information will NOT be visible to all users. This means that any program menu options will only be available to the user that performed the installation. If you are the administrator all program menu options will be visible for all users.

If you see the message in Figure 1, press Yes to install the driver or press No and contact your administrator.



Warning - Non-Administrator Installation

You are not an administrator of this system.

It is recommended that an administrator perform this installation.
If you choose to proceed, some information will NOT be visible to all users.

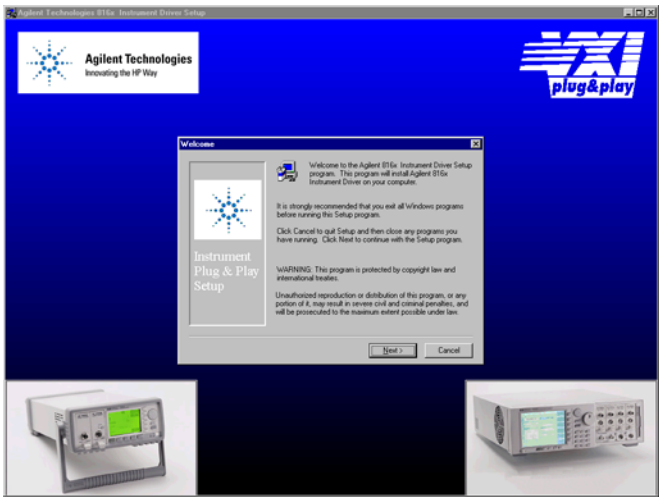
Do you want to proceed?

Figure 1 Non-Administrator Installation Pop-Up Box

NOTE

If Agilent 816x VXI*plug&play* Instrument Driver is already installed on your system, you see a message asking you if you want to uninstall the old version.
Press Yes, if required, then wait until you see a message telling you that the uninstall has been successful. You may be asked for permission to remove shared files.
Then press OK to continue.

5 You see a message, as shown in Figure 2, advising you to close the programs that you have running.



Welcome

Welcome to the Agilent 816x Instrument Driver Setup program. This program will install Agilent 816x Instrument Driver on your computer.

It is strongly recommended that you exit all Windows programs before running this Setup program.

Click Cancel to quit Setup and then close any programs you have running. Click Next to continue with the Setup program.

WARNING: This program is protected by copyright law and international treaties.

Unauthorized reproduction or distribution of this program, or any portion of it, may result in severe civil and criminal penalties, and will be prosecuted to the maximum extent possible under law.

Figure 2 Welcome Screen

6 Close these programs and press Next> to continue. Then, you see a message informing you if VISA is installed on your PC.

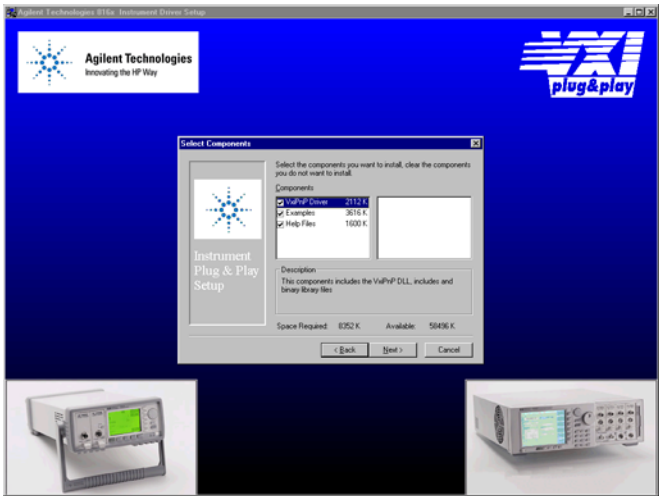
NOTE If you do not have VISA installed, press Cancel to temporarily exit this installation procedure; install VISA on your PC, then run hp816x.exe again.

If you have VISA installed, press Next> to continue. You see a window that requests you to choose your Setup.

7 You can choose a <u>T</u>ypical, <u>C</u>ompact, or C<u>u</u>stom Setup. Choose a setup option and press Next> to continue.

NOTE If you choose the C<u>u</u>stom Setup, you may choose the options you want to install from the screen in Figure 3. These options are:

- VxiPnP Driver, you may choose to install the Agilent 816x VXI*plug&play* instrument driver.
- Examples, you may choose to install Visual Basic, Visual C, LabView, Agilent VEE and VISA programming examples.
- Help Files, you may choose to install the help file.


Agilent Technologies 816x Instrument Driver Setup
Agilent Technologies
Innovating the HP Way
VXI plug&play
Instrument Plug & Play Setup

Select Components

Select the components you want to install, clear the components you do not want to install.

Components

✓ VxiPnP Driver	2112 K
✓ Examples	3616 K
✓ Help Files	1600 K

Description

This components includes the VxiPnP DLL, includes and binary library files

Space Required: 8352 K Available: 58496 K

< Back | Next > | Cancel

Figure 3 Customizing Your Setup

Select the components you want to install.

8 Press <u>N</u>ext> to continue.

Specify the program folder required; the default choice is VXIPNP.

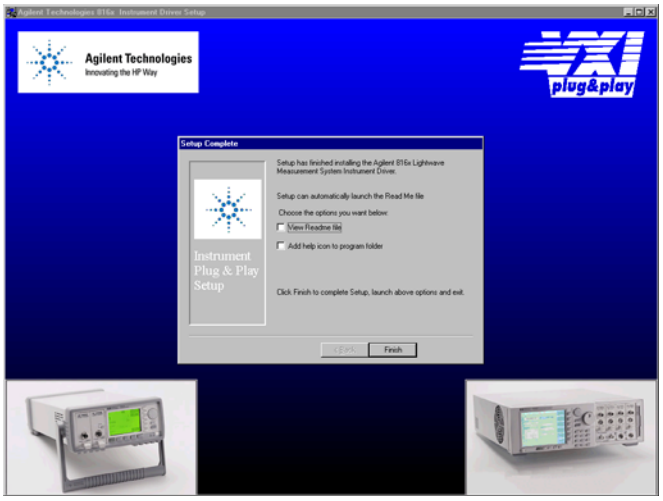
9 Press <u>N</u>ext> to continue.

Review the settings that you have specified.

If you want to review or change any settings press <u>B</u>ack>

10 Press <u>N</u>ext> to continue.

The instrument driver is installed.


Agilent Technologies 816x Instrument Driver Setup
Agilent Technologies
Innovating the HP Way
VXI plug&play
Instrument Plug & Play Setup
< Back
Finish

Setup Complete

Setup has finished installing the Agilent 816x Lightwave Measurement System Instrument Driver.

Setup can automatically launch the Read Me file

Choose the options you want below:

☐ View Readme file

☐ Add help icon to program folder

Click Finish to complete Setup, launch above options and exit.

Figure 4 Program Folder Item Options

You may elect to:

- Automatically launch the Readme file, which provides the instrument driver's version history
- Include a help icon in your program folder, which launches on-line documentation for the instrument driver

11 Press Finish to complete installation

If you elected to automatically launch the Readme file, it is displayed.

A webpage explaining how to get started with the Agilent 816x VXI*plug&play* Instrument Driver using Agilent VEE or LabView appears.

Using Visual Programming Environments

Getting Started with Agilent VEE

Agilent Technologies Visual Engineering Environment (Agilent VEE) is a visual programming language optimized for instrument control applications. To develop programs in Agilent VEE, you connect graphical 'objects' instead of writing lines of code. These programs resemble easy-to-understand block diagrams with lines.

Agilent VEE allows you to leverage your investment in textual languages by integrating with languages such as C, C++, Visual Basic, FORTRAN, Pascal, and Agilent BASIC.

Agilent VEE controls GPIB, VXI, Serial, PC Plug-in, and LAN instruments directly over the interfaces or by using instrument drivers.

Agilent VEE supports VXI*plug&play* drivers in the WIN, WIN95, WINNT, and Agilent-UX frameworks. In addition, versions 3.2 and above of Agilent VEE support the graphical Function Panel interface, providing a function tree of the hierarchy of the driver.

NOTE

This appendix assumes that you are using Windows 95. If you are using Windows NT, please replace every reference to win95 with winnt. Windows 95 and Windows NT are registered trademarks of Microsoft corporation.

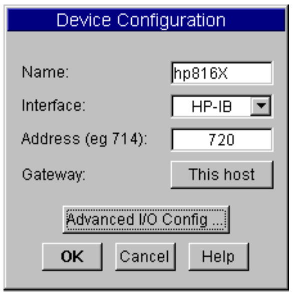
Agilent VEE automatically calls the *initialize* and *close* functions to perform automatic error checking.

GPIB Interfacing in Agilent VEE

Agilent VEE supports interfacing with an instrument from a GPIB card. Before you can do this, you must do the following:

1 Select Instrument Manager from the I/O menu.

2 Double-click on the Add button to bring up the Device Configuration screen, see Figure 5.



Device Configuration

Name: hp816X

Interface: HP-IB

Address (eg 714): 720

Gateway: This host

Figure 5 Device Configuration

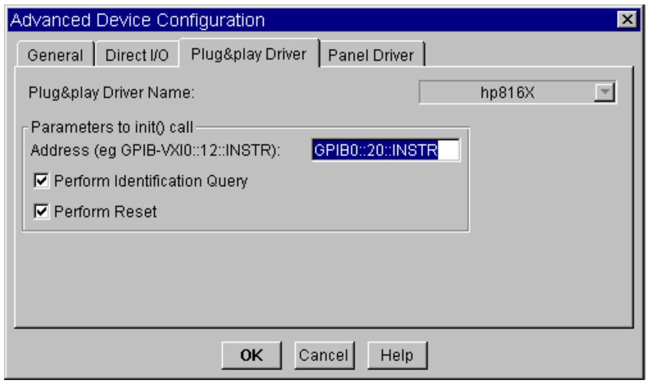
3 Enter the following information:

- Name: enter hp816x.
- Interface: GPIB
- Address: Enter the GPIB address of your GPIB interface board (the default is 7). Append the GPIB address of your instrument (the default is 20).

NOTE To find out or change the instrument's GPIB address, press the *Config* hardkey on the instrument's front panel and choose GPIB address. The instrument's GPIB address appears, you may edit it if you wish.

- Gateway: This host.

4 Press Advanced I/O Config ..., the Advanced Device Configuration box pops up. Select the Plug&play Driver tab, the box in Figure 6 appears.


Advanced Device Configuration

General | Direct I/O | Plug&play Driver | Panel Driver

Plug&play Driver Name: hp816X

Parameters to init() call

Address (eg GPIB-VXI0::12::INSTR): GPIB0::20::INSTR

☑ Perform Identification Query

☑ Perform Reset

OK | Cancel | Help

Figure 6 Advanced Device Configuration - Plug&play Driver

- Select hp816x *from the Plug&play* Driver Name drop-down list.

NOTE If you do not see this driver in the list, the driver has not installed properly.

If you do not see this driver in the list, the driver has not installed properly.

5 Enter the Parameters to the init() call by entering GPIB::xx::INSTR where xx is your instrument’s GPIB address.

NOTE 20 is the default GPIB address for your instrument.

6 Select whether to Perform Reset or to Perform Identification Query whenever Agilent VEE opens the instrument for interaction.

7 Confirm the selections pressing the OK button.

8 Return to the Instrument Manager screen and press the Save Config to save the configuration.

Getting Started with LabView

The 32-bit Agilent 816x driver can be used with LabView 5.0 and above. LabView 5.0 is a 32-bit version of LabView which runs on Windows 95 and Windows NT.

After installing the Agilent 816x instrument driver, the driver must be converted for use with LabView.

NOTE The steps below apply for Labview versions 6.5 and earlier. For later Labview versions, the conversion process is replaced by the Instrument Driver Import Wizard, available from the National Instruments web site.

1 To convert the driver follow these steps:

   a If you are updating from a previously installed driver, perfrorm the following three steps:

   b Locate the LabView program folder. By default, this is <drive>:Program Files\National Instruments\LabView.

   c This folder contains a subfolder named instr.lib.

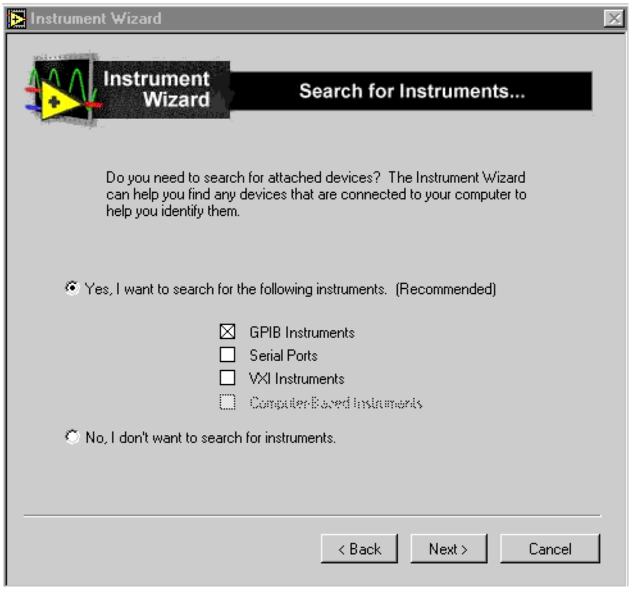
2 Run LabView.

3 On the first window that appears, click on the Solution Wizards button.

4 The LabView Solution Wizard window appears, click on the Launch Wizard... button.

5 The *W*elcome to Instrument Wizard! window appears, click on the Next > button.

6 The Search for Instruments... window appears, click on the Next > button. Check that the options are the same as displayed in the figure below:



Do you need to search for attached devices? The Instrument Wizard can help you find any devices that are connected to your computer to help you identify them.

[✓ Yes, I want to search for the following instruments. (Recommended)]

[✓ GPIB Instruments]
Serial Ports
VXI Instruments
Computer-Based Instruments

No, I don't want to search for instruments.

Figure 7 Search for GPIB Instruments

7 Click on the Next > button.

8 The Identify Found Instruments... window appears, click on the Next > button.

9 The Update VXI Plug and Play Drivers window appears, select HP816x, and click on the Convert button.

10 The *Manage Instrument Drivers* window appears, click on the Finish button.

11 The first window appears again, click on the New VI button.

12 Select File and then select Convert CVI FP file.

13 The Select a CVI Function Panel file window appears, locate the hp816x.fp file, which is normally installed into the path <drive>:VXIPNP\winXX\hp816x, where XX stands for NT, or 95.

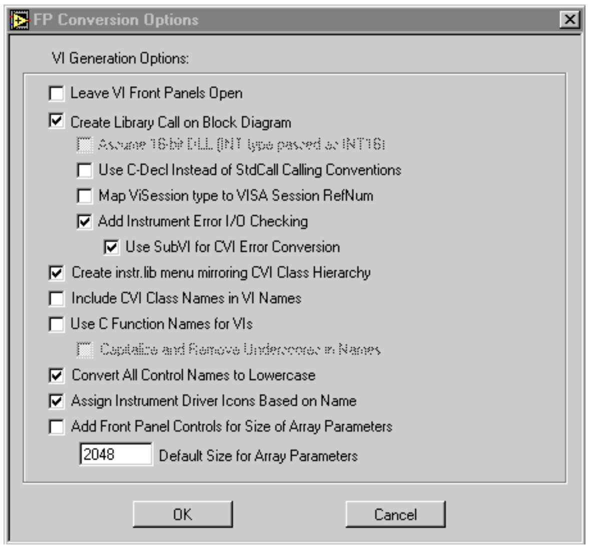
14 Press Open.

15 The CVI Function Panel Converter window appears.

16 Click on Browse... and browse to the following Destination Directory: \LabView\instr.lib\hp816x\hp816x.llb

17 Press Save.

18 Press Options..., the FP Conversion Options window appears. Check that the options are the same as displayed in the figure below:.



FP Conversion Options

VI Generation Options:

- ☐ Leave VI Front Panels Open
- ☑ Create Library Call on Block Diagram
  - ☐ Assume 16-bit DLL (INT type passed as INT16)
  - ☐ Use C-Decl Instead of StdCall Calling Conventions
  - ☐ Map ViSession type to VISA Session RefNum
  - ☑ Add Instrument Error I/O Checking
    - ☑ Use SubVI for CVI Error Conversion
- ☑ Create instr.lib menu mirroring CVI Class Hierarchy
- ☐ Include CVI Class Names in VI Names
- ☐ Use C Function Names for VIs
  - ☐ Capitalize and Remove Underscores in Names
- ☑ Convert All Control Names to Lowercase
- ☑ Assign Instrument Driver Icons Based on Name
- ☐ Add Front Panel Controls for Size of Array Parameters

2048 Default Size for Array Parameters

Figure 8 FP Conversion Options Box

NOTE

You must check the Add Front Panel Controls for Size of Array Parameters box. There will be a front panel control created for each VI that requires you to assign the array size.

19 Press OK. The CVI Function Panel Converter window appears.

20 Press OK.

21 The Select a library window appears. Browse to <drive>:\vxipnp\winXX\Bin, where XX stands for NT, or 95, select hp816x_32.dll and click on Open.

22 The CVI Conversion Status window is displayed until the conversion is completed.

NOTE

You must use the 32-bit version of the Agilent 816x VXI*plug&play* Instrument Driver with LabView 5.0.

NOTE

LabView is a trademark of National Instruments Corporation.

Getting Started with LabWindows

The 32-bit Agilent 816x VXI*plug&play* Instrument Driver can be used with LabWindows 4.0 and above. LabWindows 4.0 is a 32-bit version of LabWindows which runs on Windows 95 and Windows NT.

To access the functions of the Agilent 816x VXI*plug&play* Instrument Driver from within LabWindows, select INSTRUMENT from the main menu, and then select the LOAD... submenu item.

In the file selection dialog box which appears, select hp816x.fp and click on the OK button. LabWindows loads the function panel and instrument driver.

The driver now appears as a selection on the Instrument menu, and can be treated like any LabWindows driver.

NOTE

LabWindows is a trademark of National Instruments Corporation.

Features of the Agilent 816x Instrument Driver

The Agilent 816x VXI*plug&play* instrument driver conforms to all aspects of the VXI*plug&play* driver standard which apply to conventional rack and stack instruments.

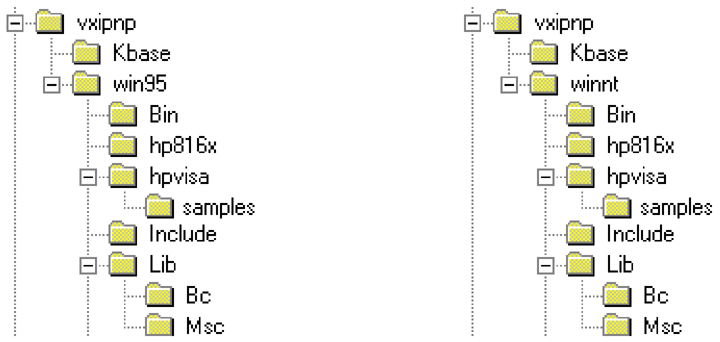
The following features are available:

- The Agilent 816x VXI*plug&play* Instrument Driver conforms with the VXI*plug&play* standard.
- There is one exception as the Agilent 816x driver does not have a soft front panel or a knowledge-based file.
- The Agilent 816x VXI*plug&play* Instrument Driver is built on top of VISA, and uses the services provided.
- VISA supports GPIB and VXI protocols. The driver can be used with any GPIB card for which the manufacturer has provided a VISA DLL.
- The Agilent 816x VXI*plug&play* Instrument Driver includes a Function Panel (.fp) file.
- The .fp file allows the driver to be used with visual programming environments such as Agilent VEE, LabWindows, and LabView.
- The Agilent 816x VXI*plug&play* Instrument Driver includes a comprehensive on-line help file which complements the instrument manual.
- The help file contains application programming examples, a cross-reference between instrument commands and driver functions, and detailed documentation of each function with examples.
- The Agilent 816x VXI*plug&play* Instrument Driver includes a Visual Basic (.BAS) file which contains the function calls in Visual Basic syntax, and allows the driver functions to be called from Visual Basic.

  You should only use Visual Basic with this driver if you are familiar with C/C++ function declarations. You must take particular care when working with C/C++ pointers.

Directory Structure

The setup program which installs the Agilent 816x instrument driver creates the VXIPNP directory if it does not already exist. The structures for the Windows NT and Windows 95 vxipnp subdirectory tree are shown in Figure 9.



```
vxipnp                      vxipnp
  Kbase                       Kbase
  win95                       winnt
    Bin                         Bin
    hp816x                      hp816x
    hpvisa                      hpvisa
      samples                     samples
    Include                     Include
    Lib                         Lib
      Bc                          Bc
      Msc                         Msc
```

Figure 9 Windows 95 and Windows NT VXIPNP Directory Structure

In the directory example, hp816x is a directory containing the instrument driver. There would be a directory for each instrument driver.

Opening an Instrument Session

To control an instrument from a program, you must open a communication path between the computer/controller and the instrument. This path is known as an instrument session, and is opened with the function

```
ViStatus hp816x_init( ViRsrc InstrDesc, ViBoolean id_query, ViBoolean reset,
ViPSession instrumentHandle );
```

Instruments are assigned a handle when the instrument session is opened. The handle, which is a pointer to the instrument, is the first parameter passed in all subsequent calls to driver functions.

The parameters of the function hp816x_init include:

- ViRsrc InstrDesc: the address of the instrument
- ViBoolean id_query: a Boolean flag which indicates if in-system verification should be performed.
  Passing VI_TRUE (1) will perform an in-system verification; passing VI_FALSE (0) will not.
  If you set id_query to false, you can use the generic functions of the instrument driver with other instruments.
- ViBoolean reset: a Boolean flag which indicates if the instrument should be reset when it is opened.
  Passing VI_TRUE (1) will perform a reset when the session is opened; passing VI_FALSE (0) will not perform a reset,
- ViPSession instrumentHandle: a pointer to an instrument session.
  InstrumentHandle is the handle which addresses the instrument, and is the first parameter passed in all driver functions.
- Successful completion of this function returns VI_SUCCESS

Closing an Instrument Session

Sessions (instrumentHandle) opened with the `hp816x_init()` function are closed with the function:

```
hp816x_close( ViSession instrumentHandle);
```

When no further communication with an instrument is required, the session must be explicitly closed (`hp816x_close()` function).

VISA does not remove sessions unless they are explicitly closed. Closing the instrument session frees all data structures and system resources allocated to that session.

VISA Data Types and Selected Constant Definitions

The driver functions use VISA data types. VISA data types are identified by the Vi prefix in the data type name (for example, ViInt16, ViUInt16, ViChar).

The file visatype.h contains a complete listing of the VISA data types, function call casts and some of the common constants.

NOTE

You can find a partial list of the type definitions and constant definitions for the visatype.h in the Agilent 816x VXI*plug&play* Instrument Driver Online Help.

Error Handling

Events and errors within a instrument control program can be detected by polling (querying) the instrument. Polling is used in application development environments (ADEs) that do not support asynchronous activities where callbacks can be used.

Programs can set up and use polling as shown below.

1. Declare a variable to contain the function completion code.

   ```
   ViStatus errStatus;
   ```

   Every driver function returns the completion code ViStatus.

   If the function executes with no I/O errors, driver errors, or instrument errors, ViStatus is 0 (VI_SUCCESS).

   If an error occurs, ViStatus is a negative error code.

   Warnings are positive error codes, and indicate the operation succeeded but special conditions exist.

2. Enable automatic instrument error checking following each function call.

   ```
   hp816x_errorQueryDetect
   (instrumentHandle, VI_TRUE);
   ```

   When enabled, the driver queries the instrument for an error condition before returning from the function.

   If an error occurred, errStatus (Step 1) will contain a code indicating that an error was detected (hp816x_INSTR_ERROR_DETECTED).

3. Check for an error (or event) after each function.

   ```
   errStatus = hp816x_cmd(instrumentHandle, "SENS1:POW:RANG");
   ```

   ```
   check(instrumentHandle, errStatus);
   ```

   After the function executes, errStatus contains the completion code.

   The completion code and instrument ID are passed to an error checking routine. In the above statement, the routine is called 'check'.

4. Create a routine to respond to the error or event. This example queries whether an error has occured, checks if the error is an instrument error and then checks if the error is a driver error.

```
void check (ViSession instrumentHandle, ViStatus errStatus)
{
 /* variables for error code and message */
 ViInt32 inst_err;
 ViChar err_message[256];

 /* VI_SUCCESS is 0 and is defined in VISATYPE.h */
 if(VI_SUCCESS > errStatus)
 {
 /* hp816x_INSTR_ERROR_DETECTED defined in hp816x.h */
  if(hp816x_INSTR_ERROR_DETECTED == errStatus)

  {
  /* query the instrument for the error */
  hp816x_error_query(instrumentHandle, &inst_err, err_message);

  /* display the error */
  printf("Instrument Error : %ld, %s\n", inst_err, err_message);
  }
  else     /* driver error */
  {
  /* get the driver error message */
  hp816x_error_message(instrumentHandle, errStatus, err_message);

  /* display the error */
  printf("Driver Error : %ld, %s\n", errStatus, err_message);

  }
  /* optionally reset the instrument, close the instrument handle */
  hp816x_reset(instrumentHandle);
  hp816x_close(instrumentHandle);
  exit(1);
 }
 return;
```

Introduction to Programming

Example Programs

See the Online Help and “VISA Programming Examples” on page 143.

VISA-Specific Information

The following information is useful if you are using the driver with a version of VISA.

Instrument Addresses

When you are using Agilent VXI*plug&play* instrument drivers, you should enter the instrument addresses using only upper case letters. This is to ensure maximum portability.

For example, use `GPIB0::22` rather than `gpib0::22`.

Callbacks

Callbacks are not supported by this driver.

Development Environments

These sections contains suggestions as to how you can use `hp816x_32.dll` within various application development environments.

Microsoft Visual C++ 4.0 (or higher) and Borland C++ 4.5 (or higher)

Please refer to your Microsoft Visual C++ or Borland C++ manuals for information on linking and calling DLLs.

Microsoft Visual Basic 4.0 (or higher)

Please refer to your Microsoft Visual Basic manual for information on calling DLLs.

The BASIC include file is `hp816x.bas`. You can find this file in the directory `~vxipnp\win95\include`, where ~ is the directory in the VXIPNP variable.

By default, ~ is equivalent to `C:\`. This means that the file is in `C:\vxipnp\win95\include`.

You may also need to include the file visa.bas. visa.bas is provided with your VISA DLL.

Agilent VEE 5.01 (or higher)

Your copy of Agilent VEE for Windows contains a document titled *Using VXIplug&play drivers with Agilent VEE for Windows*. This document contains the detailed information you need for Agilent VEE applications.

LabWindows CVI/ (R) 4.0 (or higher)

The Agilent 816x VXI*plug&play* Instrument Driver is supplied as a Dynamic Link Library (.DLL) file.

There are several advantages to using the .DLL form of the driver, including those listed below:

- transportability across different computer platforms,
- a higher level of support for the compiled driver from Agilent Technologies,
- a faster load time for your project.

LabWindows/CVI (R) will attempt by default to load the source version of the instrument driver. To load the DLL, you must include the file hp816x.fp in your project. hp816x.fp can be found in the directory vxipnp\win95\hp816x.

Do not include hp816x.C in your project.

You must provide an include file for hp816x.H. You do this by ensuring that the directory ~vxipnp\win95\include is added to the include paths (CVI Project Option menu).
~ is the directory in the VXIPNP variable. By default, ~ is equivalent to C:\. This means that the file is in C:\vxipnp\win95\include.

Online Information

The latest copy of this driver can be downloaded via:

http://www.agilent.com/comms/comp-test

If you do not have web access, use the version of hp816x.exe on your OCT Support CD, or contact your Agilent Technologies supplier.

Lambda Scan Applications

These functions combine multiple SCPI commands into a single, functional operation. They are designed to allow quick and easy access to common instrument command sequences.

These application functions allow you to perform a Multi Frame Lambda Scan - a Lambda Logging operation where an Agilent 816xA/B Lightwave Measurement System with a Tunable Laser module performs a wavelength sweep and the Tunable Laser module is coordinated with the Agilent N7744A / N7745A Multiport Power Meter. The instruments must be connected by GPIB, LAN or USB. The Output Trigger Connector of the Agilent 816xA/B Lightwave Measurement System mainframe must be connected to the Input Trigger Connector of the Multiport Power Meter.

The following two functions apply to Multi Frame Lambda Scan applications:

- The Set Lambda Scan Wavelength (hp816x_set_LambdaScan_wavelength) function allows you to use a different wavelength than 1550 nm during a Lambda Scan operation. All Power Meters taking part in the Lambda Scan operation will be set to the chosen wavelength.
- The Enable High Sweep Speed (hp816x_enableHighSweepSpeed) function enables/disables the highest available sweep speed (40 nanometers per second) for Lambda Scan operations. The Lambda Scan operation chooses the highest possible sweep speed for the chosen step size.
    - If you choose Enable, the highest sweep speed possible will be used. This may lead to less accurate measurements.
    - If you choose Disable, the highest sweep speed will not be used.

Equally Spaced Datapoints

A linear interpolation is optional for the Multi Frame Lambda Scan Application.

The advantage of spacing all measurements equally is that presenting results through use of a spreadsheet is greatly simplified. The operation returns one wavelength array and a power array for each power meter channel.

The disadvantage of using equally spaced datapoints is that the linear interpolation is analogous to the use of a low pass filter. Figure 10 shows the original curve as measured directly by a Power Meter and the interpolated curve.

Interpolation will always tend to produce a smoother curve by rounding off any peaks in the curve.

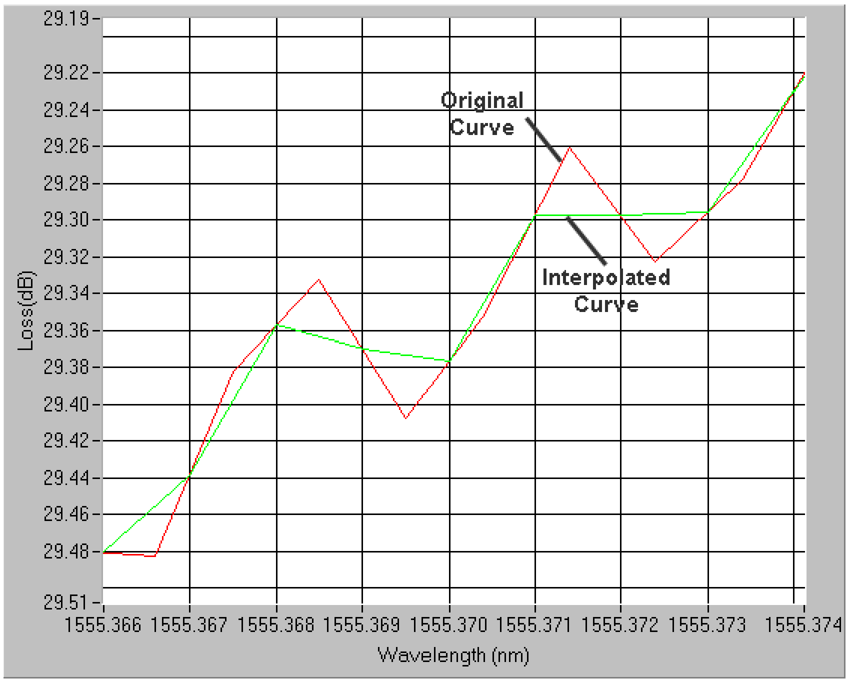


Figure 10 Equally Spaced Datapoints

How to Perform a Multi-Frame Lambda Scan Application

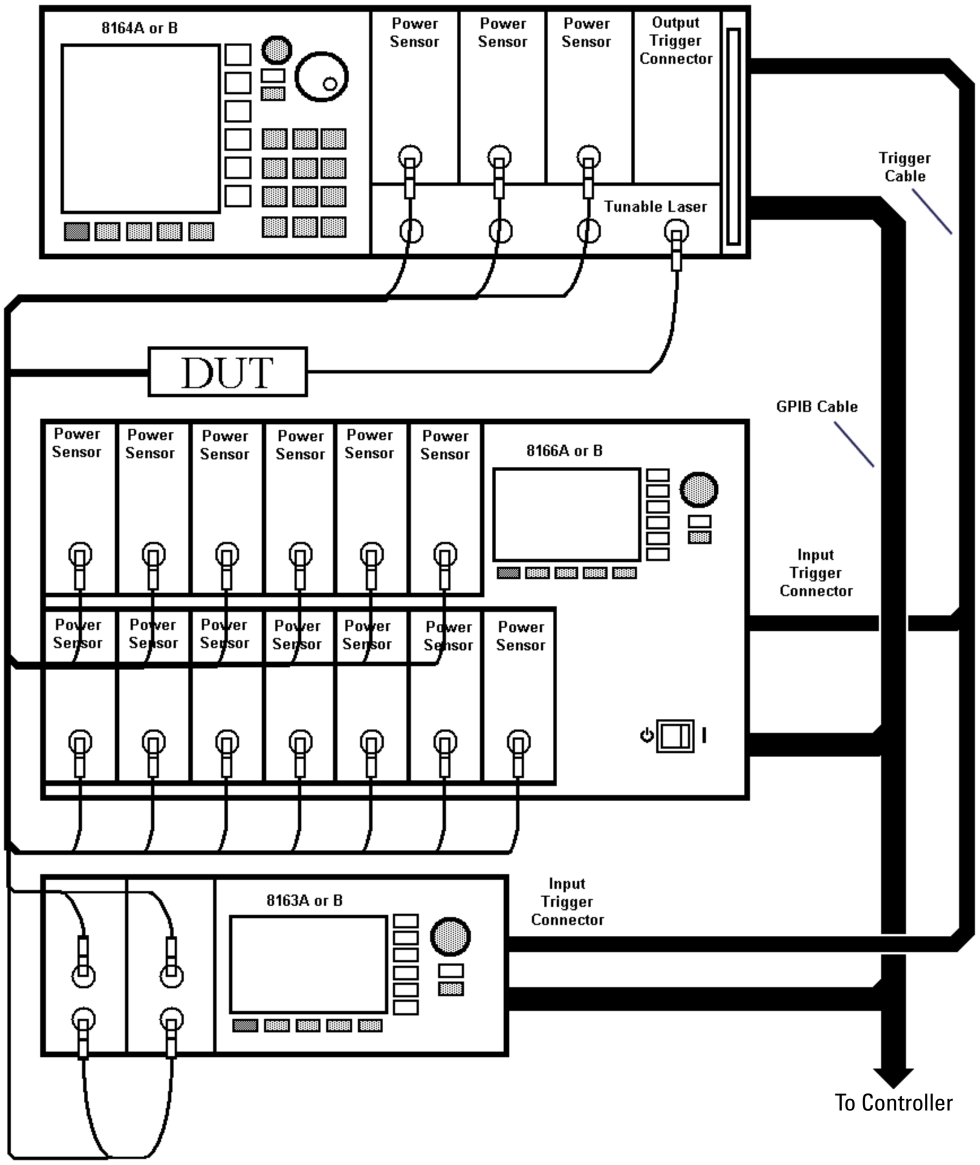


Figure 11 Multi Frame Lambda Scan Operation Setup

The Equally Spaced Datapoints Function

The Equally Spaced Datapoints (hp816x_returnEquidistantData) function allows you to select whether you the results will be equally spaced by performing a linear interpolation on the wavelength point and power measurement data, see“Equally Spaced Datapoints” on page 175 for more details.

This function is used because Lambda Scan functions make use of Lambda Logging to log the exact wavelength that measurements were triggered at. This results in Lambda Array wavelength points that are not equally spaced.

NOTE Lambda Logging is not available if your Tunable Laser module firmware revision is lower than 2.0.

Equally Spaced Datapoints is enabled as a default.

The Register Mainframe Function

Use the Register Mainframe (hp816x_registerMainframe) function to register your mainframe as a participant in a Multi Frame Lambda Scan operation. The mainframe must be connected to the GPIB bus and have their Input Trigger Connector connected to the Output Trigger Connector of the Agilent 8164A/B Lightwave Measurement System mainframe that the Tunable Laser module is installed in.

The Unregister Mainframe Function

Use the Unregister Mainframe function (hp816x_unregisterMainframe) to remove a mainframe from a Multi Frame Lambda Scan operation and clear the driver's internal data structures.

If you use LabView 5.0 the following items should be noted:

- All multi frame functions are not re-entrant, if the driver is running and initialized more than once, results may be unpredictable.
- To avoid wrong results, call the Unregister Mainframe function prior to the Initialize function (hp816x_init). This is especially necessary during program debugging, if the Close function (hp816x_close) is not called.

The Prepare Multi Frame Lambda Scan Function

The Prepare Multi Frame Lambda Scan (hp816x_prepareMfLambdaScan) function prepares a Lambda Scan operation for multiple Mainframes.

That is, it prepares an operation where a Agilent 8164A/B Lightwave Measurement System with a back-loadable Tunable Laser module and Power Meter Channels located in the Multiport Optical Power Meter. The function performs a wavelength sweep where the Tunable Laser module and Power Sensors are co-ordinated with each other.

The Prepare Multi Frame Lambda Scan (hp816x_prepareMfLambdaScan) function must be called before a Multi Frame Lambda Scan is executed. Use the return values of this function (Number of Datapoints and Number of Power Arrays) to allocate arrays for the Execute Multi Frame Lambda Scan (hp816x_executeMfLambdaScan) function.

The function scans all mainframes to find back-loadable Tunable Laser Sources. The function scans each mainframe in the order that they were originally registered by the Register Mainframe function (hp816x_registerMainframe). The first back-loadable Tunable Laser Source found will perform the sweep operation.

To obtain a higher precision, the Tunable Laser Source is set 1 nm before the Start Wavelength, this means, you have to choose a Start Wavelength 1 nm greater than the minimum possible wavelength. Also, the wavelength sweep is actually started 90 pm before the Start Wavelength and ends 90 pm after the Stop Wavelength, this means, you have to choose a Stop Wavelength 90 pm less than the maximum possible wavelength.

Triggers coordinate the Tunable Laser module with all Power Meters. The function sets for the lowest possible averaging time available for the installed Power Meters and, then, sets the highest possible sweep speed for the selected Tunable Laser module sweep. All mainframes must be connected to the GPIB bus and have their Input Trigger Connector connected to the Output Trigger Connector of the Agilent 8164A/B Lightwave Measurement System mainframe that the Tunable Laser module is installed in.

If one of the following circumstances occurs, the "parameter mismatch" error will be returned:

1 If one Power Meter is out of the specification at 1550 nm, the error "powermeter wavelength does not span 1550nm" will be returned. For example, the HP 81530A Power Sensor and the HP 81520A Optical Head are out of specification at 1550 nm. Remove the Power Meter that is out of specification at 1550 nm from the mainframe.
2 If the Step Size is too small and results in a trigger frequency that is to high for the installed Power Meters,

the error "could not calculate a sweep speed!" will be returned. Increase the Step Size.

3 If the chosen wavelength range is too large and Step Size is too small, the error "too many datapoints to log!" will be returned. In this case, reduce the wavelength range and/or increase the Step Size.

The Get MF Lambda Scan Parameters Function

The Get MF Lambda Scan Parameters (hp816x_getMFLambdaScanParameters_Q) function returns all parameters that the Prepare Multi Frame Lambda Scan (hp816x_prepareMfLambdaScan) function adjusts or automatically calculates.

The Execute Multi Frame Lambda Scan Function

The Execute Multi Frame Lambda Scan (hp816x_executeMfLambdaScan) function runs a Lambda Scan operation and returns an array that contains the wavelength values at which power measurements are made.

That is, it executes an operation where a Agilent 8164A/B Lightwave Measurement System with a back-loadable Tunable Laser module and a Multiport Optical Power Meter, performs a wavelength sweep where the Tunable Laser module and Power Sensors are coordinated with each other.

Use the values returned from the Prepare Multi Frame Lambda Scan (hp816x_prepareMfLambdaScan) function to set the parameters of the Execute Multi Frame Lambda Scan (hp816x_executeMfLambdaScan) function.

The Get Lambda Scan Result Function

The Get Lambda Scan Result (hp816x_getLambdaScanResult) function returns for a given Power Meter channel a power value array and a wavelength value array.

These arrays contains the results of the last Multi Frame Lambda Scan operation.

The Get Number of PWM Channels Function

The Get Number of PWM Channels (hp816x_getNoOfRegPWMChannnels_Q) function returns the number of Power Meter channels in a test setup.

Only Power Meters whose mainframe was registered using the Register Mainframe (hp816x_registerMainframe) function are counted.

The Get Channel Location Function

The Get Channel Location function (hp816x_getChannelLocation_Q) returns the location of the chosen Power Meter channel as used in a Multi Frame Lambda Scan operation.

The maximum number of channels that may be specified is 1000.

7
Error Codes

This chapter gives information about error codes used with the Agilent N7744A / N7745A Multiport Optical Power Meter, Agilent N775xA/6xA Multichannel Attenuators and N7711A/14A Tunable Laser System Source.

SCPI Error Strings

NOTE

Error strings in the range -100 to -183 are defined by the SCPI standard, downloadable from: http://www.scpiconsortium.org/scpistandard.htm

String descriptions taken from this standard (VERSION 1999.0 May, 1999), whether in whole or in part, are enclosed by [ ].

Table 1 Overview for Supported Strings

Error	
Number	**String**
-100 to -199 Command Errors	
-100	"Command Error" [This is the generic syntax error used when a more specific error cannot be detected. This code indicates only that a Command Error as defined in *IEEE 488*.2,11.5.1.1.4 has occurred.]
-101	"Invalid character" [A syntactic element contains a character which is invalid for that type; for example, a header containing an ampersand, SETUP&. This error might be used in place of error -114 and perhaps some others.]
-102	"Syntax error" [An unrecognized command or data type was encountered; for example, a string was received when the **device** does not accept strings.]
-103	"Invalid separator" [The parser was expecting a separator and encountered an illegal character; for example, the semicolon was omitted after a program message unit]
-104	"Data type error" [The parser recognized a data element different than one allowed; for example,numeric or string data was expected but block data was encountered.]
-105	"GET not allowed" [A Group Execute Trigger was received within a program message (see *IEEE488*.2, 7.7).]
-108	"Parameter not allowed" [More parameters were received than expected for the header]
-109	"Missing parameter" [Fewer parameters were recieved than required for the header]
-112	"Program mnemonic too long" [The header contains more than twelve characters (see *IEEE 488*.2, 7.6.1.4.1).]
-113	"Undefined header" [The header is syntactically correct, but it is undefined for this specific **devic**e; for example, *XYZ is not defined for any device.]

Table 1 Overview for Supported Strings

Error	
Number	String
-114	"Header suffix out of range" An invalid subopcode was used.
-120	"Numeric data error" [This error, as well as errors -121 through -129, are generated when parsing a data element which appears to be numeric, including the nondecimal numeric types. This error message is used if the **device** cannot detect a more specific error.]
-121	"Invalid character in number" [An invalid character for the data type being parsed was encountered; for example, an alpha in a decimal numeric]
-123	"Exponent too large" [The magnitude of the exponent was larger than 32000 (see *IEEE 488.*2,7.7.2.4.1).]
-124	"Too many digits" [The mantissa of a decimal numeric data element contained more than 255 digits excluding leading zeros (see *IEEE 488.*2, 7.7.2.4.1).]
-128	"Numeric data not allowed" [A legal numeric data element was received, but the **device** does not accept one in this position for the header.]
-131	"Invalid suffix" [The suffix does not follow the syntax described in *IEEE 488.*2, 7.7.3.2, or thesuffix is inappropriate for this **devic**e.]
-134	"Suffix too long" [The suffix contained more than 12 characters (see *IEEE 488.*2, 7.7.3.4).]
-138	"Suffix not allowed" [A suffix was encountered after a numeric element which does not allow suffixes.]
-141	"Invalid character data" [Either the character data element contains an invalid character or the particular element received is not valid for the header.]
-148	"Character data not allowed" [A legal character data element was encountered where prohibited by the **devic**e.]
-150	"String data error" [This error, as well as errors -151 through -159, are generated when parsing a string data element. This error message is used when the **device** cannot detect a more specific error.]
-151	"Invalid string data" [A string data element was expected, but was invalid for some reason (see *IEEE 488.*2, 7.7.5.2); for example, an END message was received before the terminal quote character.]
-158	"String data not allowed" [A string data element was encountered but was not allowed by the **device** at this point in parsing.]

Table 1 Overview for Supported Strings

Error	
Number	**String**
-161	“Invalid block data” [A block data element was expected, but was invalid for some reason (see *IEEE 488*.2, 7.7.6.2); for example, an END message was received before the length was satisfied.]
-168	“Block data not allowed” [A legal block data element was encountered but was not allowed by the **device** at this point in parsing.]
-170	“Expression error” [This error, as well as errors -171 through -179, are generated when parsing an expression data element. This particular error message is used when the **device** cannot detect a more specific error.]
-171	“Invalid expression” [The expression data element was invalid (see *IEEE 488*.2, 7.7.7.2); for example, unmatched parentheses or an illegal character.]
-178	“Expression data not allowed” [A legal expression data was encountered but was not allowed by the **device** at this point in parsing.]
-181	“Invalid outside macro definition” [Indicates that a macro parameter placeholder ($<number) was encountered outside of a macro definition.]
-183	“Invalid inside macro definition” [Indicates that the program message unit sequence, sent with a *DDT or *DMC command, is syntactically invalid (see *IEEE 488*.2, 10.7.6.3).]
-185	“Subop out of range” *Description:* Suboperations are parameters that are passed to refine the destination of a command. They are used to address slots, channels, laser selections and GPIB/SCPI register levels. This error is generated if the parameter is not valid in the current context or system configuration. *Example:* This error occurs if the user queries the status of a summary register and passes an invalid status level (also see "Status for 816x" on page 28 programmer's guide). *Note:* Incorrect slots and channels addresses are handled by error code -301
-200 to -299 Execution Errors	
-200	“Execution error (StatExecError)" *Description:* This error occurs when the current function, instrument or module state (or status) prevents the execution of a command. This is a generic error which can for a number of reasons. *Example:* When a powermeter has finished a logging application and data is available, the user is not able to reconfigure the logging application parameters. First, the user must stop the logging application.

Table 1 Overview for Supported Strings

Error	
Number	**String**
-201	"Please be patient - GPIB currently locked out" *Description:* Some operations block the complete system. Since no sensible measurements are possible while this is true, the GPIB is locked out. *Example:* When ARA, Lambda zeroing or zeroing is executing on a TLS module, the GPIB is not accessible.
-205	"Powermeter not running (StatMeterNotRunning)" *Description:* Some command and actions may stop the data aquisition unit of a powermeter. If a command fetches data, there may be no measurement values and this error is generated. Please check module state and repeat operation.
-211	"Trigger ignored" *Description:* A trigger has been detected but ignored because of timing contraints. (For Example: average time to large).
-212	"Arm ignored" *Description:* Included for compatibility.
-213	"Init ignored" *Description:* The INIT:IMM command (page 117) initiates a trigger and completes a full measurement cycle. The continuous measurement must be DISABLED. This error code is generated if the powermeter is still in cont. measurement mode.
-220	"Parameter error (StatParmError)" *Description:* The user has passed a parameter that cannot be changed in this way. The device cannot detect one of the following more specific errors:
-220	-220, "Parameter error (StatParmOutOfRange)" *Description:* The user has passed a parameter that exceeds the valid range for this parameter.
-220	"Parameter error (StatParmIllegalVal)" *Description:* The user has passed a parameter that does not match a value in a list of possible values.
-221	"Settings conflict (StatParmInconsistent)" *Description:* The user has passed a parameter that conflicts with other already configured parameters.
-222	"Data out of range"

Table 1 Overview for Supported Strings

Error	
Number	**String**
-222	"Data out of range (ParmTooLarge)" *Description:* The user has passed a parameter that is too large.
-222	"Data out of range (ParmTooSmall)" *Description:* The user has passed a parameter that is too small.
-222	"Data out of range (StatParmTooLarge)" *Description:* The user has passed a continuous parameter that is too large. *Example:* Wavelength 1800nm when maximum wavelength is 1700nm.
-222	"Data out of range (StatParmTooSmall)" *Description:* The user has passed a continuous parameter that is too small. *Example:* Wavelength 700nm when minimum wavelength is 800nm.
-223	“Too much data” *Description:* A function returns more data or the user requests more data than the application is able to handle.
-224	“Illegal parameter value” [Used where exact value, from a list of possibles, was expected.]
-225	"Out of memory" *Description:* The request application or function cannot be executed because the instrument runs out of memory.
-231	"Data questionable (StatValNYetAcc)" *Description:* The data that is retured is not accurate or reliable. The user should repeat the operation. The reason for this error is unspecific. *Example:* A powermeter configured a long average time has not completed its current measurement cycle when the user queries the current power.
-231	"Data questionable (StatRangeTooLow)" *Description:* As -231 (StatValNYetAcc) but for a more specific reason: The powermeter readout data is not reliable because the currently set (manual) range does not correspond with the input power.

Table 1 Overview for Supported Strings

Error Number	 String
-261	"Math error in expression (StatUnitCalculationError)" *Description:* This may occur when the user attempts to transform data in a way that is currently not possible. *Example:* When a powermeter is measuring very small powe values in dBm (such as noise power), negative power values in Watt may also be present (such as when the powermeter calibration wavelength does not correspond to the wavelength of input signal). The instrument cannot transform negative Watt values to dBm because the logarithm of a negative value is not defined.
-272	“Macro execution error” [Indicates that a syntactically legal macro program data sequence could not beexecuted due to some error in the macro definition (see *IEEE 488*.2, 10.7.6.3.)]
-273	“Illegal macro label” [Indicates that the macro label defined in the *DMC command was a legal string syntax, but could not be accepted by the **device** (see *IEEE 488*.2, 10.7.3 and 10.7.6.2); for example, the label was too long, the same as a common command header, or contained invalid header syntax.]
-276	“Macro recursion error” [Indicates that a syntactically legal macro program data sequence could not be executed because the device found it to be recursive (see *IEEE 488*.2, 10.7.6.6).]
-277	“Macro redefinition not allowed” [Indicates that a syntactically legal macro label in the *DMC command could not be executed because the macro label was already defined (see *IEEE 488*.2,10.7.6.4).]
-278	“Macro header not found” [Indicates that a syntactically legal macro label in the *GMC? query could not be executed because the header was not previously defined.]
-284	"Function currently running (StatModuleBusy)" *Description:* This error is generated when a function is currently running on a module so that it cannot process another commands. *Example:* When a powermeter is running a logging application, you are not able to configure the logging application parameters (also see -200).
-286	"No function currently running" *Description:* This error is generated when a user tries to execute a command which requires a particular set of data that is not available. *Example:* Application data is necessary to execute SENSE:FUNC:RES?. If no suitable function has completed, there is no data and this error is generated. (also see -200).

Table 1 Overview for Supported Strings

Error	
Number	**String**
-290 (Multiport Power Meter)	“Application currently running - no SCPI support” *Description:* Included for compatibility
-290 (Variable optical attenuator)	"Invalid setting" The selected setting is invalid (e.g. if a non-existing setting is recalled or an invalid index is used)
-291	"Inconsistent setting"
-292	"Validation failed" For example, if a measurement was started with inconsistent parameter setting.
-300 to -399 or between 1 and 32767 Device-Specific Errors (Module)	
-300	“Internal error (StatVals Lost)” “Internal error (StatInternalError)” *Description* These are generic device-dependent errors used when the instrument cannot detect more specific errors.
-301	"Module doesn't support this command (StatCmdUnknown)" *Description:* The addressed module does not support the SCPI command. *Example:* When a command from the SENSe SCPI tree is sent to a fixed or tunable laser source.
-302	"Internal timeout error (StatTimedOut)" *Description:* A command has not returned in the expected time.
-303	"Module slot empty or slot / channel invalid" *Description:* The user has send a command to an empty slot.
-304	"Command was aborted (StatAborted)" *Description:* The command has been interrupted by another event.
-305	"Internal messaging error (StatCmdError)" "Internal messaging error (StatCmdNotAllowed)" "Internal messaging error (StatWrongLength)" "Internal messaging error (StatWrongReceiver)" "Internal messaging error (StatBufAllocError)" "Internal messaging error (StatDPRamFull)"; } "Internal messaging error (StatSemError)" *Description:* An error has occured in the instrument communication system. Please report this error with a description of the circumstances that generated the error and the configuration of the system.

Table 1 Overview for Supported Strings

Error Number	String
-306	"Channel doesn't support this command (StatCmdUnknownForSlave)" *Description:* Slave channels have limited functionality. The module supports this command, but the command must be sent to the master channel.
-307	"Channel without head connection (StatHeadless)" *Description:* The channel supports this command, but it cannot be executed because the optical measurement head is not plugged into the interface module.
-310	“System error” [Indicates that some error, termed “system error” by the device, has occurred. This code is device-dependent.]
-310	“System error (CaldataAccessError)”
-310	“System error (SettingAccessError)”
-310	“System error (EepromAccessError)”
-310	“System error (RtcAccessError)” RTC = real time clock
-311	“Hardware not available / option not installed”
-321	“Out of memory” [An internal operation needed more memory than was available.]
-322	"Flash programming error (StatFlashEraseFailed)" "Flash programming error (StatFlashWriteFailed)" "Flash programming error (StatFlashDataCntError)" "Flash programming error (StatFlashDPAlgoFailed)" *Description:* An error has occured in a module. Please report this error with a description of the circumstances that generated the error and the configuration of the system.
-323	"Flash programming error (StatUserCalTable Empty)" It is not possible to activate the offset (I) functionality when the offset table is empty "Flash programming error (StatUserCalTable Full)" The offset (I) table is full and no more ? can be stored "Flash programming error (StatUserCalActive)" It is not possible to program the offset (I) table when the offset (I) feature is activated. Deactivate first.
-330	“Self-test failed” *Description:* You have started the self test, but the module has detected an error while executing it
-340	"Printing error (StatPrintError)" *Description:* An unspecified problem occurred while communicating with the printer.

Table 1 Overview for Supported Strings

Error	
Number	**String**
-341	"Printing error - paper out (StatPaperOut)" *Description:* The instrument cannot print because there is no paper in the connected printer.
-342	"Printing error - offline (StatOffline)" *Description:* The instrument cannot print because the connected printer is offline.
-350	"Queue overflow" [A specific code entered into the queue in lieu of the code that caused the error. This code indicates that there is no room in the queue and an error occurred but was not recorded.]
-400 to -499 Query Errors	
-400	"Query error" [This is the generic query error for **devices** that cannot detect more specific errors. This code indicates only that a Query Error as defined in *IEEE 488*.2, 11.5.1.1.7 and 6.3 has occurred.]
-410	"Query INTERRUPTED" [Indicates that a condition causing an INTERRUPTED Query error occurred (see *IEEE 488*.2, 6.3.2.3); for example, a query followed by DAB or GET before a response was completely sent.]
-420	"Query UNTERMINATED" [Indicates that a condition causing an UNTERMINATED Query error occurred (see *IEEE 488*.2, 6.3.2.2); for example, the **device** was addressed to talk and an incomplete program message was received.]
-430	"Query DEADLOCKED" [Indicates that a condition causing an DEADLOCKED Query error occurred (see *IEEE 488*.2, 6.3.1.7); for example, both input buffer and output buffer are full and the device cannot continue.]
-440	"Query UNTERMINATED after indef resp" [Indicates that a query was received in the same program message after an query requesting an indefinite response was executed (see *IEEE 488*.2, 6.5.7.5).]
Other Errors	
-601	"Not allowed while (re)starting network"
-602	"Network startup failed"
-610	"Not allowed before bootup finished" Hardware not fully initialized
-611	"Not allowed because of bootup error" Hardware not fully initialized
-612	"Not allowed while measurement is running"
-613	"Not allowed during firmware download"
-614	"Not allowed during selfcal"
-616	"Not allowed while ongoing zeroing"

Table 2 Overview for Unsupported Strings

Error	
Number	**String**
all positive errors	
-110	“Command header error”
-111	“Header seperator error”
-114	“Header suffix out of range”
-130	“Suffix error”
-140	“Character data error”
-144	“Character data too long”
-160	“Block data error”
-201	“Invalid while in local”
-202	“Settings lost due to ???”
-210	“Trigger error”
-214	“Trigger deadlock”
-215	“Arm deadlock”
-230	“Data corrupt or stale”
-240	“Hardware error”
-241	“Hardware missing”
-260	“Expression error”
-280	“Program error”
-281	“Cannot create program”
-282	“Illegal program name”
-283	“Illegal variable name”
-285	“Program syntax error”
-286	“Program runtime error”
-311	“Memory error” [checksum or parity]
-312	“Protect user data memory lost”
-313	“Calibration memory lost”
-314	“Save/Recall Memory lost”
-315	“Configuration memory lost”

Index

www.agilent.com

Fifth edition, September 2011

N7744-90C01

Agilent Technologies