MT9085 Series
ACCESS Master
SCPI Remote Control
Operation Manual

Second Edition

- For safety and warning information, please read this manual before attempting to use the equipment.
- Additional safety and warning information is provided within the MT9085 Series ACCESS Master Operation Manual. Please also refer to it before using the equipment.
- Keep this manual with the equipment.

ANRITSU CORPORATION

Document No. M-W3972AE-2.0
Safety Symbols

To prevent the risk of personal injury or loss related to equipment malfunction, Anritsu Corporation uses the following safety symbols to indicate safety-related information. Ensure that you clearly understand the meanings of the symbols BEFORE using the equipment. Some or all of the following symbols may be used on all Anritsu equipment. In addition, there may be other labels attached to products that are not shown in the diagrams in this manual.

Symbols used in manual

⚠️ **DANGER**
This indicates a very dangerous procedure that could result in serious injury or death if not performed properly.

⚠️ **WARNING**
This indicates a hazardous procedure that could result in serious injury or death if not performed properly.

⚠️ **CAUTION**
This indicates a hazardous procedure or danger that could result in light-to-severe injury, or loss related to equipment malfunction, if proper precautions are not taken.

Safety Symbols Used on Equipment and in Manual

The following safety symbols are used inside or on the equipment near operation locations to provide information about safety items and operation precautions. Ensure that you clearly understand the meanings of the symbols and take the necessary precautions BEFORE using the equipment.

- This indicates a prohibited operation. The prohibited operation is indicated symbolically in or near the barred circle.
- This indicates an obligatory safety precaution. The obligatory operation is indicated symbolically in or near the circle.
- This indicates a warning or caution. The contents are indicated symbolically in or near the triangle.
- This indicates a note. The contents are described in the box.
- These indicate that the marked part should be recycled.
Notes On Export Management

This product and its manuals may require an Export License/Approval by the Government of the product's country of origin for re-export from your country.

Before re-exporting the product or manuals, please contact us to confirm whether they are export-controlled items or not.

When you dispose of export-controlled items, the products/manuals need to be broken/shredded so as not to be unlawfully used for military purpose.
About This Manual

The operation manuals for the MT9085 Series ACCESS Master consist of separate documents for the main unit, remote control, and quick guide. This operation manual describes the SCPI (Standard Commands for Programmable Instruments) commands for the MT9085A/B/C ACCESS Master (hereinafter ACCESS Master).
Table of Contents

About This Manual ........................................ I

Chapter 1 Overview ........................................ 1-1
  1.1 About Remote Control .................................. 1-2
  1.2 Applications ............................................. 1-3

Chapter 2 Before Use ....................................... 2-1
  2.1 Preparing Equipment ................................... 2-2
  2.2 Connecting Equipment .................................. 2-3
  2.3 Setting Ethernet ......................................... 2-6
  2.4 Network Setup on PC ................................... 2-10
  2.5 Checking Connection .................................... 2-13
  2.6 Message Format ......................................... 2-14
  2.7 Checking Instrument Status ......................... 2-17
  2.8 Controlling Message Sync ............................ 2-22
  2.9 Switching SM Port and MM Port ..................... 2-25
  2.10 Moving to Another Measurement Mode from
       Top Menu .................................................. 2-27

Chapter 3 Platform SCPI Commands ....................... 3-1
  3.1 IEEE 488.2 Common Commands ....................... 3-2
  3.2 System Commands ....................................... 3-10
  3.3 Status Subsystem Commands ......................... 3-12
  3.4 Instrument Subsystem Commands ..................... 3-29
  3.5 TOPMenu Subsystem Commands ....................... 3-33
Chapter 4 OTDR Commands .....................4-1

4.1 Command Summary ...........................................4-2
4.2 Root Level Commands .........................................4-13
4.3 SOURce Subsystem Commands ..........................4-17
4.4 SENSE Subsystem Commands ..........................4-27
4.5 TRACe Subsystem Commands ..........................4-45
4.6 DISPlay Subsystem Commands ..........................4-55
**Commands**

*CLS ................................................................................................. 3-2
*ESE ................................................................................................. 3-2
*ESE? ................................................................................................. 3-3
*ESR? ................................................................................................. 3-3
*IDN? ................................................................................................. 3-4
*OPC ................................................................................................. 3-4
*OPC? ................................................................................................. 3-5
*RST ................................................................................................. 3-6
*SRE ................................................................................................. 3-6
*SRE? ................................................................................................. 3-7
*STB? ................................................................................................. 3-8
*TST? ................................................................................................. 3-8
*WAI ................................................................................................. 3-9
SYSTem:ERRor? ................................................................................. 3-10
SYSTem:VERSion? ............................................................................ 3-10
SYSTem:LIGHt? ................................................................................ 3-11
SYSTem:LIGHt .................................................................................. 3-11
STATus:OPERation[:EVENt]? ......................................................... 3-12
STATus:OPERation:CONDition? ..................................................... 3-12
STATus:OPERation:BIT<n>:CONDition? ......................................... 3-13
STATus:OPERation:BIT<n>:ENABle ................................................ 3-13
STATus:OPERation:BIT<n>:ENABle? ................................................ 3-14
STATus:OPERation:BIT<n>[:EVENt]? ............................................... 3-14
STATus:OPERation:ENABle .......................................................... 3-15
STATus:OPERation:ENABle? ......................................................... 3-15
STATus:OPERation:INSTrument:CONDition? ............................... 3-15
STATus:OPERation:INSTrument:ENABle ................................. 3-16
STATus:OPERation:INSTrument:ENABle? ..................................... 3-16
STATus:OPERation:INSTrument[:EVENt]? .................................... 3-17
STATus:OPERation:INSTrument:ISUMmary<n>:
  CONDition? ............................................................................... 3-17
STATus:OPERation:INSTrument:ISUMmary<n>:ENABle .... 3-18
STATus:OPERation:INSTrument:ISUMmary<n>:ENABle? .. 3-18
STATus:OPERation:INSTrument:ISUMmary<n>[:EVENt]? .. 3-19
STATus:QUESTionable[:EVENt]? ................................................... 3-19
STATus:QUESTionable:CONDition? .............................................. 3-20
STATus:QUESTionable:BIT<n>:CONDition? ......................... 3-20
STATus:QUESTionable:BIT<n>:ENABle ................................. 3-21
STATus:QUESTionable:BIT<n>:ENABle? .................................. 3-21
STATus:QUESTionable:BIT<n>[:EVENt]? .......................... 3-22
STATus:QUESTionable:ENABle ............................................ 3-22
STATus:QUESTionable:ENABle? .......................................... 3-23
STATus:QUESTionable:INSTrument:CONDition? ................. 3-23
STATus:QUESTionable:INSTrument:ENABle ........................ 3-24
STATus:QUESTionable:INSTrument:ENABle? ...................... 3-24
STATus:QUESTionable:INSTrument[:EVENt]? ..................... 3-25
STATus:QUESTionable:INSTrument:ISUMmary<n>:
    CONDition? .................................................................. 3-25
STATus:QUESTionable:INSTrument:ISUMmary<n>:
    ENABle ...................................................................... 3-26
STATus:QUESTionable:INSTrument:ISUMmary<n>:
    ENABle? ..................................................................... 3-26
STATus:QUESTionable:INSTrument:ISUMmary<n>[:EVENt]?  .. 3-27
STATus:PRESet ..................................................................... 3-27
INSTrument:CATalog? ....................................................... 3-29
INSTrument:CATalog:FULL? ............................................. 3-29
INSTrument:NSELect ......................................................... 3-30
INSTrument:NSELect? ....................................................... 3-30
INSTrument[:SELect] .......................................................... 3-31
INSTrument[:SELect]? ....................................................... 3-31
INSTrument:STATe ............................................................. 3-32
INSTrument:STATe? ............................................................ 3-32
TOPMenu:UNIT:CATalog? ............................................... 3-33
TOPMenu:UNIT:CATalog:FULL? ..................................... 3-33
TOPMenu:UNIT[:SELect] .................................................... 3-34
TOPMenu:UNIT[:SELect]? .................................................. 3-34
TOPMenu:UNIT:NSELect ................................................... 3-34
TOPMenu:UNIT:NSELect? .................................................. 3-35

ABORt ................................................................................. 4-13
STOP ................................................................................ 4-13
INITiate ............................................................... 4-14
INITiate:AUTO ..................................................... 4-14
INITiate:RTIME .................................................. 4-15
INITiate? ......................................................... 4-16
SOURce:WAVElength:AVailable? .................... 4-17
SOURce:WAVElength ........................................ 4-17
SOURce:WAVElength? .................................... 4-18
SOURce:RANGE:AVailable? ............................. 4-18
SOURce:RANGE ................................................. 4-19
SOURce:RANGE? ............................................... 4-19
SOURce:RESo:AVailable? ............................... 4-20
SOURce:RESo ................................................. 4-21
SOURce:RESo? ............................................... 4-21
SOURce:PULSE:AVailable? .............................. 4-22
SOURce:PULSE ............................................... 4-22
SOURce:PULSE? ............................................... 4-23
SOURce:PULSE:ENHanced:AVailable? .......... 4-23
SOURce:PULSE:ENHanced .............................. 4-24
SOURce:PULSE:ENHanced? ............................ 4-25
SOURce:AVERAGEs:TIME .................................. 4-25
SOURce:AVERAGEs:TIMe? .............................. 4-26
SENSe:AVERAGEs? ......................................... 4-27
SENSe:AVERAGEs:TIME? ............................. 4-27
SENSe:TRACE:READY? ................................. 4-28
SENSe:CONCheck? ....................................... 4-28
SENSe:CONCheck ........................................... 4-29
SENSe:LIVCheck? .......................................... 4-29
SENSe:LIVCheck ............................................ 4-30
SENSe:FIBer:IOR ............................................ 4-31
SENSe:FIBer:IOR? .......................................... 4-31
SENSe:FIBer:BSC ........................................... 4-32
SENSe:FIBer:BSC? ......................................... 4-32
SENSe:HOFFset .............................................. 4-33
SENSe:HOFFset? ........................................... 4-33
SENSe:VOFFset .............................................. 4-34
SENSe:VOFFset? ........................................... 4-34
SENSe:ACURsor ............................................ 4-35
SENSe:ACURsor? ........................................... 4-36
SENSe:BCURsor ............................................ 4-36
Chapter 1  Overview

This chapter explains remote control of the ACCESS Master.

1.1 About Remote Control ........................................ 1-2
1.2 Applications.........................................................1-3
1.1 About Remote Control

The remote control function sends commands via the communications interface from the remote control PC (hereinafter PC) to set the measuring instrument and read the measurement results and measuring instrument conditions.

The ACCESS Master uses Ethernet or Wi-Fi as communication interface.

This document explains the commands that follow the format defined by SCPI (Standard Commands for Programmable Instruments).

The commands to control the ACCESS Master are called “control commands” and the commands to query data from the ACCESS Master are called “query commands”.

Commands are comprised of character strings. For example, the following command sets the measurement wavelength to 1550 nm:

```
SOURce:WAVelength 1550
```

A query command has the question symbol (?) appended to the string. For example, sending the following command queries the Distance Range set at the ACCESS Master.

```
SOURce:RANge?
```

The PC receives the following response from the ACCESS Master.

```
100
```

This response indicates that the Distance Range setting is 100 km.

When the ACCESS Master is in the remote control state, Local Operation is displayed on a softkey. During the remote control, all the keys except the power key and Local Operation cannot be used. To cancel remote control, touch Local Operation.
1.2 Applications

The main uses for remote control are listed below:

Automating Measurements
Instead of panel operations, measurement can be automated by controlling the ACCESS Master by executing programs.

Controlling Multiple Instruments
The characteristics of multiple DUT (device under test)s can be measured simultaneously by remote control of multiple instruments.

Remote Control of Instruments
Measuring instruments at remote locations can be controlled over communications lines to collect measurement data.
Chapter 2  Before Use

This chapter explains remote control of the ACCESS Master.

2.1 Preparing Equipment ......................................................2-2
2.2 Connecting Equipment ..................................................2-3
  2.2.1 Connecting Ethernet Cable ................................2-3
  2.2.2 Dongle Connection ..............................................2-5
2.3 Setting Ethernet .............................................................2-6
  2.3.1 Ethernet Settings .................................................2-6
  2.3.2 Wi-Fi Settings ......................................................2-7
2.4 Network Setup on PC .....................................................2-10
2.5 Checking Connection .....................................................2-13
2.6 Message Format .............................................................2-14
  2.6.1 Message Formats ..................................................2-14
  2.6.2 Header Composition ..............................................2-14
  2.6.3 Data Formats .......................................................2-15
2.7 Checking Instrument Status .............................................2-17
  2.7.1 Register Structure ..................................................2-17
  2.7.2 Status Byte Register ..............................................2-19
  2.7.3 Event Register .......................................................2-20
2.8 Controlling Message Sync ..............................................2-22
2.9 Switching SM Port and MM Port ......................................2-25
2.10 Moving to Another Measurement Mode from Top Menu ..........2-27
2.1 Preparing Equipment

The following equipment is required to perform remote control.

- PC
- USB network devices
- Ethernet cable
- Program development tools

PC
The PC must be Ethernet-equipped and able to run the program development tools.

USB network devices
Connect the following devices to the USB port of the ACCESS Master.

<table>
<thead>
<tr>
<th>Device</th>
<th>Specification</th>
</tr>
</thead>
<tbody>
<tr>
<td>USB·Ethernet converter</td>
<td>USB1.1/2.0, 10/100BASE·T</td>
</tr>
<tr>
<td>Wi-Fi dongle</td>
<td>USB1.1/2.0, IEEE 802.11b/g/n</td>
</tr>
</tbody>
</table>

Ethernet cable
Use a category-5 or better Ethernet cable.

Program Development Tools
Software to specify and transmit Ethernet address and port number.
2.2 Connecting Equipment

Connect the ACCESS Master to a network via the USB Ethernet converter or Wi-Fi dongle.

When connecting the USB Ethernet converter to the powered-on ACCESS Master:
This interrupts the measurement and displays the Ethernet Settings screen.

When connecting the USB Ethernet converter to the powered-off ACCESS Master:
Power on the ACCESS Master, and then touch Remote Setup in the Top Menu.

The settings of IP address, etc. are saved when the power is turned Off. The last settings are loaded when the power is turned On next time.

2.2.1 Connecting Ethernet Cable

Connect the Ethernet cable to the USB port (General) of the ACCESS Master, via the USB Ethernet converter.

To connect the ACCESS Master and PC directly, connect the LAN port of the USB Ethernet converter and that of PC using a cross cable. To connect multiple external devices, use a network hub and straight cables.
Note:
Sometimes communications with the ACCESS Master are not established due to the external equipment communications status. For stable communications, connect the ACCESS Master and PC directly using a crossover cable.
2.2.2 Dongle Connection

Connect the Wi-Fi dongle to the USB port (General) of the ACCESS Master.

![Wi-Fi dongle connection](image)

Figure 2.2.2-1 Dongle Connection
2.3 Setting Ethernet

Instrument settings such as IP address are set at the Remote Setup Screen. After setting the IP Address of the ACCESS Master, also set the IP address of the PC according to 2.4 “Network Setup on PC”.

2.3.1 Ethernet Settings

1. In the Top Menu, touch **Remote Setup**.

2. Touch **Ethernet Settings**.

   ![Ethernet Settings Screen](image)

   **Figure 2.3.1-1 Ethernet Settings Screen**

3. If you want to set IP address automatically, touch **DHCP** to turn it on. Then proceed to Step 8.

4. Touch **IP Address** and set IP address for the ACCESS Master. The input setting range is 0.0.0.1 to 255.255.255.254. The default setting is 192.168.1.2.

5. Touch **IP Netmask** and set IP netmask for the ACCESS Master. The input setting range is 0.0.0.0 to 255.255.255.255 and the default setting is 255.255.255.0.

6. Touch **Default Gateway** and set IP address for the default gateway. The input setting range is 0.0.0.1 to 255.255.255.254 and the default is None.

7. Touch **SCPI Port Number** and set a port number to be used...
for SCPI command communication. The input setting range is 1 to 65535 and the default is 2288.

8. When you finish editing the selected item, touch **Apply**.

9. Touch **Connect**.

10. Check that the following icon appears on the ACCESS Master screen. This means the ACCESS Master has successfully connected to the network.

   If DHCP is turned On in Step 3, IP Address and IP Netmask can be checked on the Ethernet Settings screen.

   ![Network Connection Icon](image)

### 2.3.2 Wi-Fi Settings

Follow the steps below to configure Wi-Fi settings.

1. In the Top Menu, touch **Remote Setup**.

2. Touch **Wi-Fi Settings**.

3. Touch **Selected Network**.

   ![Wi-Fi Setting Screen](image)

   **Figure 2.3.2-1 Wi-Fi Setting Screen**
4. Touch **Refresh** to update the SSID display. If a network you want to connect to is not displayed, touch **Other** to add the SSID.

![Choose Network Screen](image)

**Figure 2.3.2-2  Choose Network Screen**

5. In the SSID list, touch a network to connect, and then touch **Select**. The SSID of the selected network is displayed on the Wi-Fi Settings screen.

6. To set IP address automatically, touch DHCP to turn it On. Then proceed to Step 12.

7. Touch **IP Address** and set IP address for the ACCESS Master. The input setting range is 0.0.0.1 to 255.255.255.254. The default setting is 192.168.1.2.

8. Touch **IP Netmask** and set IP netmask for the ACCESS Master. The input setting range is 0.0.0.0 to 255.255.255.255 and the default setting is 255.255.255.0. Some combinations of numbers are invalid for IP netmask even if they are in the setting range.

9. Touch **Default Gateway** and set IP address for the default gateway. The input setting range is 0.0.0.1 to 255.255.255.254 and the default is None.

10. Touch **SCPI Port Number** and set a port number to be used for SCPI command communication.
2.3 Setting Ethernet

The input setting range is 1 to 65535 and the default is 2288.

11. When you finish editing the selected item, touch **Apply**.

12. Touch **Connect**.

13. Check that the following icon appears on the ACCESS Master screen. This means the ACCESS Master has successfully connected to the network.

   If DHCP is turned On in Step 6, IP Address and IP Netmask can be checked on the Wi-Fi Settings screen.
2.4 Network Setup on PC

If it is necessary to set IP address, follow the setting procedure of the PC operating system. The example uses the Window 10 screens.

1. Go to Start, click Settings, and then click Network & Internet.

2. Click Change adapter options.

3. To use the USB Ethernet converter, right-click Ethernet, and then click Properties.

Note:

When using a Wi-Fi dongle, right-click Wi-Fi.
2.4 Network Setup on PC

4. In the Ethernet Properties dialog box, click Internet Protocol Version 4 (TCP/IPv4), and then click Properties.

5. In the Internet Protocol Version 4 (TCP/IPv4) Properties dialog box, click Use the following IP address.
6. Assign an IP address that is different from the one set for the ACCESS Master. Here, configure as follows and click **OK**:
   - IP address: 192.168.1.3
   - Subnet mask: 255.255.255.0

7. In the **Ethernet Properties** dialog box, click **OK**.
2.5 Checking Connection

1. Click **Programs** at the Windows Start menu.
2. Click **Accessories**.
3. Click **Command Prompt**.
4. Input the commands shown in the screen below. This example shows how to set the IP address to 192.168.1.2.

```
C:\Users>ping 192.168.1.2
Ping 192.168.1.2 with 32 bytes of data:
Reply from 192.168.1.2: bytes=32 time=1ms TTL=64
Reply from 192.168.1.2: bytes=32 time=1ms TTL=64
Reply from 192.168.1.2: bytes=32 time=1ms TTL=64
Reply from 192.168.1.2: bytes=32 time=1ms TTL=64

Ping statistics for 192.168.1.2:
  Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
  Approximate round trip times in milli-seconds:
    Minimum = 1ms, Maximum = 1ms, Average = 1ms
```

5. Check that the “Request timed out” message is not displayed. Check that the following contents are correct.

- IP address, IP Netmask, Default Gateway, SCPI Port Number
- Cable connection
- USB Ethernet converter connection
- Wi-Fi dongle connection
2.6 Message Format

Messages are composed of character strings for executing commands and character strings indicating the message end. When sending messages to the ACCESS Master, terminate the line with CR/LF; received messages are terminated with CR/LF.

2.6.1 Message Formats

Messages are in the following formats, depending on the send direction.

Program Messages:
Messages sent from PC to the ACCESS Master.
Program messages are divided into the following two types.
- Command
  Used for setting measurement conditions or starting measurement, etc.
- Query
  Used for querying the state or setup of the ACCESS Master.
  If a query is sent, the ACCESS Master creates a response message.

These messages are composed of header and data parts separated by a white space. Program messages must have an appended header but may not include data.

Response Messages:
Messages sent from the ACCESS Master to PC.
Response messages must have appended data but may not include a header.

2.6.2 Header Composition

The header is composed of alphanumeric characters and underbars while the first character is an alphabetic character. However, common commands defined by IEEE 488.2 have an asterisk (*) appended to the head of the string. The program is not case-sensitive.

Command with only header:
*RST
*CLS
2.6 Message Format

INITiate:AUTo

Command with header and data:
*ESE 255
SOURce:WAVelength 1310
INStrument:SELect OTDR_STD

Messages with multiple data use commas (,) to separate the data parts.
Example:
SENSe:LSALeft 0.0,10.0
SENS:ANAL:PAR 0.05,-60.0,3.0

Queries have a question mark (?) appended to the header.
Example:
*ESR?
TOPMenu:UNIT:CATalog?
SOUR:PULS:ENH?

When linking multiple program messages, separate the message using semicolons (;). The maximum number of linked messages is 12. If 13 or more messages are sent, the 13th and subsequent messages are discarded.

Example:
SOUR:WAV 1310;SOUR:RAN 100;SOUR:RES 0;INIT;*WAI

2.6.3 Data Formats

The data format is character string data, numeric data, and binary data.

Character String Data
This is a string of ASCII code.
The following example shows a program message for switching to the OTDR (Standard) mode from the Top Menu.

Example:
INStrument:SELect OTDR_STD

Numeric Data
This is described in binary numerals.
Example:

```
SYSTem:LOCK 1
TOPM:UNIT 2
SENSe:LOSS:MODE 5
SENSe:FIB:IOR 1.500000
SENSe:VOFF -10.0
```

When using binary numbers, input numeric values either as integers or floating point representation.

**Binary Data**

The head string starts with a number sign (#) and continues with data after a numeric value indicating the data length.

The character after the number sign (#) indicates the number of digits in the data.

The binary data follows the number indicating the data length.

Example:

```
#524047an%qe4445+¥...
```

5 digits 24047 bytes of binary data
2.7 Checking Instrument Status

The ACCESS Master has registers indicating the status, such as errors and command execution status. This section explains these registers.

2.7.1 Register Structure

Figure 2.7.1-1 shows the structure of the registers indicating the ACCESS Master status.

![Register Structure Diagram]

**Figure 2.7.1-1  Register Structure**
Each register uses 8-bit data. The register output values are the binary totals for each bit shown in Figure 2.7.1-1.

<table>
<thead>
<tr>
<th>Bit</th>
<th>Binary value</th>
<th>Bit</th>
<th>Binary value</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>4</td>
<td>16</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>5</td>
<td>32</td>
</tr>
<tr>
<td>2</td>
<td>4</td>
<td>6</td>
<td>64</td>
</tr>
<tr>
<td>3</td>
<td>8</td>
<td>7</td>
<td>128</td>
</tr>
</tbody>
</table>

To output the status byte register data, set the bit corresponding to the service request enable register to 1. To read the status byte register, set the service request enable register. The logical product of these two registers is read by \(^{*}\)STB.

Each standard event register has a corresponding enable register. The logical product of the event and enable registers is obtained and the logical sum of this result is output at bit 5 of the status byte register.

**Questionable Status Register**
This register is reserved for future use. It is not used.

**Message Queue**
The message queue is always empty, so the response message from the ACCESS Master can be sent immediately. Up to 12 messages from the PC can be spooled.

**Error Queue**
Up to 12 error messages from the ACCESS Master can be spooled.
2.7 Checking Instrument Status

2.7.2 Status Byte Register

The meaning of each bit of the status byte register is shown in the following table.

<table>
<thead>
<tr>
<th>Bit</th>
<th>Explanation</th>
</tr>
</thead>
</table>
| 7   | OSR (Operation Status Register)  
      Displays the ACCESS Master’s operation status.  
      Currently only 1 can be set during OTDR  
      measurement. This is the logical sum of each bit of the  
      logical product of the operation event register and  
      status operation enable register. |
| 6   | MSS (Master Summary Register)  
      This indicates whether the value read from the status  
      byte register is 0 or not.  
      It is the logical sum of bit 7 and bit 5 to 0 of the logical  
      product of the status byte register and the service  
      request enable register. |
| 5   | ESB (Event Status Bit)  
      This is the logical sum of each bit of the logical  
      product of the standard event status register and  
      standard event enable register. |
| 4   | MAV (Message Available summary)  
      This is always 0 in the ACCESS Master, because  
      messages are not spooled and are sent immediately. |
| 3   | Questionable Status Register  
      Not used; always 0 |
| 2   | Error / Event Queue  
      This is 1 when there is an error message in the Error  
      queue. |
| 1   | Not used; always 0 |
| 0   | Not used; always 0 |

Bit 7 to bit 0 of the status byte register can be read with \*STB.

The \*SRE and \*SRE? common commands can be used for setting and reading the service request enable register for setting reading of the status byte register. To output the status byte register data, set the bit corresponding to the service request enable register to 1.
Chapter 2  Before Use

Bits 5, 3, and 2 of the status byte register can be set to 0 using the *
CLS* common command.

When *
CLS* is sent after a command or when a query is sent after *
CLS*, the send queue is cleared and bit 4 is set to 0.

The service request enable register cannot be set to 0 using *
CLS*, so use *
SRE*.

2.7.3  Event Register

The meaning of each bit of the standard event status register is
listed in the table below.

<table>
<thead>
<tr>
<th>Bit</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>Power-on</td>
</tr>
<tr>
<td></td>
<td>Becomes 1 at power-on and 0 each time 1 is read.</td>
</tr>
<tr>
<td>6</td>
<td>User Request</td>
</tr>
<tr>
<td></td>
<td>Not used; always 0</td>
</tr>
<tr>
<td>5</td>
<td>Command Error</td>
</tr>
<tr>
<td></td>
<td>Becomes 1 when received undefined program message, message that cannot executed according to syntax, or message with spelling error.</td>
</tr>
<tr>
<td>4</td>
<td>Execution Error</td>
</tr>
<tr>
<td></td>
<td>Becomes 1 when received program message that cannot be executed because parameter specification is out of range.</td>
</tr>
<tr>
<td>3</td>
<td>Device Dependent Error</td>
</tr>
<tr>
<td></td>
<td>Becomes 1 at errors other than command, execution and query errors.</td>
</tr>
<tr>
<td>2</td>
<td>Query Error</td>
</tr>
<tr>
<td></td>
<td>Becomes 1 when no data to read in output queue or output queue data fails for some reason.</td>
</tr>
<tr>
<td>1</td>
<td>Request Control</td>
</tr>
<tr>
<td></td>
<td>Not used; always 0</td>
</tr>
<tr>
<td>0</td>
<td>Operation Complete</td>
</tr>
</tbody>
</table>
|     | Indicates whether or not device completely ended operations in event table. This command responds only to the *
OPC* command. |

Bit 7 to bit 0 of the standard event status register can be read by
the \texttt{*ESR?} command. The standard event status register returns to 0 when read.

The standard event Status enable register can be set and read using the \texttt{*ESE} and \texttt{*ESE?} commands. To output standard event register data, set the bit corresponding to the enable register to 1.

Bit 0 can be read using the \texttt{*OPC} common command.

The standard register can be set to 0 using the \texttt{*CLS} command.
2.8 Controlling Message Sync

The following messages (12 types max.) can be received during measurement by the ACCESS Master and analysis of measurement results. However, if a message is sent to change the measurement parameters before the previous processing is completed, the message is discarded and the correct measurement conditions will not be set.

The following program message changes the wavelength to 1550 nm while performing averaging measurements at a wavelength of 1310 nm.

SOUR:WAV 1310; INIT; SOUR:WAV 1550

Figure 2.8-1 shows the message execution sequence when this message is sent to the ACCESS Master. After setting the initial wavelength, sweeping starts when INIT is sent. Although a command to change the wavelength to 1550 nm is sent during sweeping, the command is ignored during measurement.

Figure 2.8-1  Message Execution Sequence

The control for processing the next command after completing processing of the message sent first is called sync control.

Sync control is performed by the following methods.

- Using *WAI command
- Using *OPC? query
- Using *OPC command and *ESR? query
- By querying execution end
The *WAI command, *OPC? query, *OPC command and *ESR? query can be used for all messages.

Using *WAI
The *WAI common command instructs processing to wait until processing of the message sent before the *WAI command is completed before executing the next command.

Example of Use: \texttt{INIT;*WAI;SOUR:WAV 1550;INIT}

![Diagram of message sync control using *WAI](image)

**Figure 2.8-2 Sync Control using *WAI**

Using *OPC? 
The *OPC? common command queries the OPC bit indicating the end of message processing.

Examples of Use:

\begin{itemize}
  \item \texttt{INIT} \hspace{1cm} \text{Executes averaging measurement}
  \item \texttt{*OPC?} \hspace{1cm} \text{Queries if operation is completed and waits until “1” returns.}
  \item \texttt{> 1} \hspace{1cm} \text{Measuring stopped when response is 1.}
  \item \texttt{SENS:TRAC:READY?} \hspace{1cm} \text{Queries presence/absence of measured waveform data.}
  \item \texttt{> 1} \hspace{1cm} \text{Waveform data is ready when response is 1.}
  \item \texttt{TRAC:LOAD:SOR?} \hspace{1cm} \text{Gets trace object from ACCESS Master.}
  \item \texttt{SOUR:WAV 1550} \hspace{1cm} \text{Changes wavelength}
  \item \texttt{INIT} \hspace{1cm} \text{Starts measurement with new measurement conditions.}
\end{itemize}

Using *OPC and *ESR?
The *OPC common command sets bit 0 (OPC bit) of the standard event status register to 1 when the message processing is
completed.

Examples of Use:

*CLS  Sets the OPC bit to 0.
*ESE 1 Sets standard event status register to 1.
INIT  Executes averaging measurement.
*OPC  Sets so as to change standard event status register OPC bit (bit 0) to 1 after measurement completed.
*ESR? Standard Event Status register query
> 0  Measuring when response is 0
*ESR? Standard Event Status register query
> 1  Measuring stopped when response is 1.
SENS:TRAC:READY? Queries presence/absence of measured waveform data.
> 1  Waveform data is ready when response is 1.
TRAC:LOAD:SOR? Gets trace object from ACCESS Master.
SOUR:WAV 1550 Changes wavelength.
INIT  Starts measurement with new measurement conditions.
2.9 Switching SM Port and MM Port

Two ports for single mode fiber (measurement wavelength: 1310/1550 nm) and multimode fiber (measurement wavelength: 850/1300 nm) are installed in ACCESS Master with option 063.

Switch between the SM and MM as described below.

Example of Use 1: with MT9085B-063 (MM 850/1300 nm, SM 1310/1550 nm)

```
```

Queries installed port.

At MM, 1, SM, 2 response: Multimode port installed as port 1 and single mode port installed as port 2.

```
TOPMenu:UNIT:NSElect?
>1
```

Queries current measurement port.

At 1 response: Multimode port selected.

```
TOPM:UNIT:NSEL 2
```

Switches to single mode port.

---

**Figure 2.9-1 Top Menu**

Switch between the SM and MM as described below.
Example of Use 2: with MT9085B-063 (MM 850/1300 nm, SM 1310/1550 nm)

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&gt;MM,SM</td>
<td>At MM, SM response: Multimode port and single mode port installed.</td>
</tr>
<tr>
<td>TOPMenu:UNIT?</td>
<td>Queries current measurement port.</td>
</tr>
<tr>
<td>&gt;MM</td>
<td>At MM response: Multimode port selected.</td>
</tr>
<tr>
<td>TOPM:UNIT:SEL SM</td>
<td>Switches to single mode port.</td>
</tr>
</tbody>
</table>
2.10 Moving to Another Measurement Mode from Top Menu

In addition to OTDR measurement, the ACCESS Master has other built-in measurement mode functions such as Light Source and Power Meter. To use these measurements, it is necessary to switch measurement mode.

The remote control modes supported by the current ACCESS Master are Top Menu and OTDR (Standard).

This example explains how to switch the OTDR (Standard) mode to the measurement mode from the Top Menu.

Switching between the Top Menu and OTDR (Standard) mode requires the following:

Example of Use 1: To perform measurement at 1310 nm with MT9085C-053

```
INStrument:CATalog:FULL?

>TOP_MENU, 1, OTDR_STD, 2

INSTrument:NSELect?

>1

INST:NSEL 2

INST:STAT 1

SOUR:WAV 1310
```

Queries measurement mode supporting remote control.

At TOP_MENU, 1, OTDR_STD, 2:
Supports selection of Mode 1 at Top Menu and Mode 2 at OTDR (Standard).

Queries current mode.

At 1 response:
Top Menu

Switches to OTDR (Standard).

Note: Measurement mode inactive

Enables OTDR (Standard) mode.

Sets wavelength to 1310 nm.
Example of Use 2: To measure with MM unit after measuring with SM unit using MT9085B-063 (MM 850/1300 nm, SM 1310/1550 nm)

```plaintext
INSTrument:NSELect?
>2

INST:NSEL 1

TOPMenu:UNIT:CATalog:FULL?
>MM,1,SM,2

TOPM:UNIT:NSEL 1
```

Queries current measurement mode.

At 2 response:
OTDR (Standard) mode

Moves to Top Menu.

Queries installed OTDR.

At MM, 1, SM, 2 response:
Multimode port installed as port 1 and single mode port installed as port 2.

Switches to multimode port.
Chapter 3 Platform SCPI Commands

This chapter details the SCPI commands for the ACCESS Master platform.

3.1 IEEE 488.2 Common Commands ....................... 3-2
3.2 System Commands ........................................... 3-10
3.3 Status Subsystem Commands ......................... 3-12
3.4 Instrument Subsystem Commands .................. 3-29
3.5 TOPMenu Subsystem Commands ..................... 3-33
3.1 IEEE 488.2 Common Commands

*CLS

**Function**
The Clear Status command *CLS clears all the event registers summarized in the Status Byte register. Except for the output queue, all queues summarized in the Status Byte register are emptied. The error queue is also emptied. Neither the Standard Event Status Enable register, nor the Service Request Enable register are affected by this command.

**Syntax**
*CLS

**Parameters**
None

**Response Data**
None

**Example of Use**
*CLS

*ESE

**Function**
The standard Event Status Enable command (*ESE) sets bits in the Standard Event Status Enable register. A 1 in a bit in the enable register enables the corresponding bit in the Standard Event Status register. The register is cleared at power-on. The *RST and *CLS commands do not affect the register.

**Syntax**
*ESE<wsp><value>

**Parameters**
The bit value for the register (a short value):

- 7: (MSB) Power On  128
- 6: User Request  64
- 5: Command Error  32
3.1 IEEE 488.2 Common Commands

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Description</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>Execution Error</td>
<td>16</td>
</tr>
<tr>
<td>3</td>
<td>Device Dependent Error</td>
<td>8</td>
</tr>
<tr>
<td>2</td>
<td>Query Error</td>
<td>4</td>
</tr>
<tr>
<td>1</td>
<td>Request Control</td>
<td>2</td>
</tr>
<tr>
<td>0</td>
<td>(LSB) Operation Complete</td>
<td>1</td>
</tr>
</tbody>
</table>

Response Data
None

Example of Use
*ESE 21

*ESE?

Function
The standard Event Status Enable query *ESE? returns the contents of the Standard Event Status Enable register (see **ESE for information on this register).

Syntax
*ESE?

Parameters
None

Response Data
The bit value for the register (a short value).

Example of Use
*ESE?
> 21<END>

*ESR?

Function
The standard Event Status Register query *ESR? returns the contents of the Standard Event Status register. The register is cleared after being read.

Syntax
*ESR?
Parameters
None

Response Data
The bit value for the register (a short value):
7: (MSB) Power On 128
6: User Request 64
5: Command Error 32
4: Execution Error 16
3: Device Dependent Error 8
2: Query Error 4
1: Request Control 2
0: (LSB) Operation Complete 1

Example of Use
*ESR?
> 22<END>

*IDN?

Function
The Identification query *IDN? gets the instrument identification over the interface.

Syntax
*IDN?

Parameters
None

Response Data
The identification string terminated by <END>.

Example of Use
*IDN?
> ANRITSU,MT9085C,6260123456<END>

*OPC

Function
A device is in the Operation Complete Command Active State (OCAS) after it has executed *OPC.
The device returns to the Operation Complete Command Idle State (OCIS) whenever the No Operation Pending flag is TRUE, and at the same time the OPC bit of the SESR is set to “1”. The following events force the device into OCIS. In these cases, the No Operation Pending flag is FALSE and the OPC bit of the SESR is not set to “1”:
. Power-on
. *CLS
. *RST

**Syntax**

*OPC

**Parameters**

None

**Response Data**

None

**Example of Use**

`*OPC`

*OPC?*

**Function**

A device is in the Operation Complete Query Active State (OQAS) after it has executed *OPC?. The device returns to the Operation Complete Query Idle State (OQIS) whenever the No Operation Pending flag is TRUE, at the same time placing a “1” in the Output Queue. The following actions cancel the *OPC? query (and put the instrument into Operation Complete, Command Idle State):
. Power-on

**Syntax**

*OPC?

**Parameters**

None
Response Data
1 <END>.

Example of Use
*OPC?
> 1<END>

*RST
Function
The ReSeT command *RST sets the ACCESS Master to Top Menu. Pending *OPC actions are cancelled. The ACCESS Master is placed in the idle state awaiting a command. The *RST command clears the error queue.
The following are not changed:
. Output queue
. Service Request Enable register (SRE)
. Standard Event Status Enable register (ESE)

Syntax
*RST

Parameters
None

Response Data
None

Example of Use
*RST

*SRE
Function
The standard Service Request Enable command (*SRE) sets bits in the Service Request Enable register.
A 1 in a bit in the enable register enables the corresponding bit in the Service Request Enable register.
The register is cleared at power-on. The *RST and *CLS commands do not affect the register.
3.1 IEEE 488.2 Common Commands

Syntax
*SRE\(\text{wsp}\)<value>

Parameters
The bit value for the register (a **short** value):

- 7: Operation Status Summary \(128\)
- 6: Master Summary Status (MSS) / Request Service (RQS) \(64\)
- 5: Standard Event Status Summary (ESB) \(32\)
- 4: Message Available (MAV) \(16\)
- 3: Questionable Status Summary \(8\)
- 2: Error/Event Queue Summary \(4\)
- 1: Available \(2\)
- 0: Available \(1\)

Response Data
None

Example of Use
*SRE 64

*SRE?

Function
The Service Request Enable query *SRE? returns the contents of the Service Request Enable register (see *SRE for information on this register).

Syntax
*SRE?

Parameters
None

Response Data
The bit value for the register (a **short** value):

- 7: Operation Status Summary \(128\)
- 6: Master Summary Status (MSS) / Request Service (RQS) \(64\)
- 5: Standard Event Status Summary (ESB) \(32\)
- 4: Message Available (MAV) \(16\)
- 3: Questionable Status Summary \(8\)
- 2: Error/Event Queue Summary \(4\)
Example of Use

*STB?

Function

The Status Byte query *STB? returns the contents of the Status Byte register. The Master Summary Status (MSS) bit is true when any enabled bit of the STB register is set (excluding Bit 6). The Status Byte register including, the master summary bit, MSS, is not directly altered because of an *STB? query.

Syntax

*STB?

Parameters

None

Response Data

The bit value for the register (a short value):

<table>
<thead>
<tr>
<th>Bit Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>Operation Status Summary</td>
</tr>
<tr>
<td>6</td>
<td>Master Summary Status (MSS) / Request Service (RQS)</td>
</tr>
<tr>
<td>5</td>
<td>Standard Event Status Summary (ESB)</td>
</tr>
<tr>
<td>4</td>
<td>Message Available (MAV)</td>
</tr>
<tr>
<td>3</td>
<td>Questionable Status Summary</td>
</tr>
<tr>
<td>2</td>
<td>Error/Event Queue Summary</td>
</tr>
<tr>
<td>1</td>
<td>Available</td>
</tr>
<tr>
<td>0</td>
<td>Available</td>
</tr>
</tbody>
</table>

Example of Use

*TST?

Function

This query is not used with the ACCESS Master now.
The self-test query *TST? makes the currently selected logical instrument to perform a self-test and place the results of the test in the output 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.

**Syntax**

* TST?

**Parameters**

None

**Response Data**

The sum of the results for the individual tests (a 32-bit signed integer value)

**Example of Use**

* TST?
  > 0<END>

*WAI

**Function**

The Wait command *WAI prevents the instrument from executing any further commands until the current command has finished executing. All pending operations are completed during the wait period.

**Syntax**

*WAI

**Parameters**

None

**Response Data**

None

**Example of Use**

*WAI
3.2 System Commands

SYSTem:ERRor?

Function
Queries the contents of the SCPI error queue.
Removes the returned entry from the queue.

Syntax
SYSTem:ERRor?

Parameters
None

Response Data
The number of the latest error, sorted by the error commands
sending order, and its meaning.

Example of Use
SYST:ERR?
> -100,"std_command,Command Parse Error"<END>

SYSTem:VERSion?

Function
Queries the SCPI revision to which the system complies.

Syntax
SYSTem:VERSion?

Parameters
None

Response Data
The revision year and number string.

Example of Use
SYST:VERS?
> 1990.0<END>
3.2 System Commands

SYS TEM:LIGHT?

Function
Queries if the backlight is turned ON or OFF.

Syntax
SYST em:LIGHT?

Parameters
None

Response Data
Possible responses are:
0: The backlight is OFF.
1: The backlight is ON.

Example of Use
SYST:LIGHT?
> 0<END>

SYS TEM:LIGHT

Function
Sets the backlight on the system ON or OFF.

Syntax
SYST em:LIGHT<value>

Parameters
<value>
Boolean format
Range: 0|1
0: Turn the backlight OFF on the unit.
1: Turn the backlight ON on the unit.

Response Data
None

Example of Use
SYST:LIGHT 1<END>
3.3 Status Subsystem Commands

STATus:OPERation[:EVENt]?

Function
Queries the operation event register.

Syntax
STATus:OPERation[:EVENt]?

Parameters
None

Response Data
The bit value for the operation event register as a short value (0 .. +32767)

Example of Use
STAT:OPER?
> 0<END>

STATus:OPERation:CONDition?

Function
Queries the operation condition register.

Syntax
STATus:OPERation:CONDition?

Parameters
None

Response Data
The bit value for the operation condition register as a short value (0 .. +32767)

Example of Use
STAT:OPER:COND?
> 16<END>
3.3 Status Subsystem Commands

STATus:OPERation:BIT<n>:CONDition?

**Function**
This command accesses the user-definable bits in the OPERation register set. The value of \(<n>\) is restricted from 8 to 12 and represents bits 8 through 12 in the operation status register.

**Syntax**
STATus:OPERation:BIT<n>:CONDition?

**Parameters**
None

**Response Data**
The bit value for the operation condition register as a *short* value (0 .. 1)

**Example of Use**
STAT:OPER:BIT8:COND?
> 1<END>

STATus:OPERation:BIT<n>:ENABle

**Function**
Sets the operation enable mask for the event register specified bit. The value of \(<n>\) is restricted from 8 to 12 and represents bits 8 through 12.

**Syntax**
STATus:OPERation:BIT<n>:ENABle<wsp><value>

**Parameters**
The bit value for the operation enable mask as a *short* value (0 .. 1)

**Response Data**
None

**Example of Use**
STAT:OPER:BIT11:ENAB 1
STATus:OPERation:BIT<n>:ENABle?

**Function**
Queries the operation enable mask for the event register specified bit.
The value of <n> is restricted from 8 to 12 and represents bits 8 through 12 in the operation status register.

**Syntax**
STATus:OPERation:BIT<n>:ENABle?

**Parameters**
None

**Response Data**
The bit value for the operation enable mask as a short value (0 .. 1)

**Example of Use**
STAT:OPER:BIT9:ENAB?
> 0<END>

STATus:OPERation:BIT<n>[:EVENt]?

**Function**
Queries the operation event register specified bit.
The value of <n> is restricted from 8 to 12 and represents bits 8 through 12 in the operation status register.

**Syntax**
STATus:OPERation:BIT<n>[:EVENt]?

**Parameters**
None

**Response Data**
The bit value for the operation event register as a short value (0 .. 1)

**Example of Use**
STAT:OPER:BIT10:EVEN?
> 0<END>
3.3 Status Subsystem Commands

STATus:OPERation:ENABle

Function
Sets the operation enable mask register for the event register.

Syntax
STATus:OPERation:ENABle<wsp><value>

Parameters
The bit value for the operation enable mask as a short value (0 .. +32767)

Response Data
None

Example of Use
STAT:OPER:ENAB 128

STATus:OPERation:ENABle?

Function
Queries the operation enable mask register for the event register.

Syntax
STATus:OPERation:ENABle?

Parameters
None

Response Data
The bit value for the operation enable mask as a short value (0 .. +32767)

Example of Use
STAT:OPER:ENAB?
> 128<END>

STATus:OPERation:INStrument:CONDition?

Function
Queries the instrument operation condition register.
Syntax
STATus:OPERation:INStrument:CONDition?

Parameters
None

Response Data
The bit value for the operation condition register as a short value (0 .. +32767)

Example of Use
STAT:OPER:INST:COND?
> 16<END>

STATus:OPERation:INStrument:ENABle
Function
Sets the instrument operation enable mask for the event register.

Syntax
STATus:OPERation:INStrument:ENABle<ws><value>

Parameters
The bit value for the operation enable mask as a short value (0 .. +32767)

Response Data
None

Example of Use
STAT:OPER:INST:ENAB 128

STATus:OPERation:INStrument:ENABle?
Function
Queries the instrument operation enable mask for the event register.

Syntax
STATus:OPERation:INStrument:ENABle?
3.3 *Status Subsystem Commands*

**Parameters**
None

**Response Data**
The bit value for the operation enable mask as a `short` value (0 .. +32767)

**Example of Use**
```
STAT:OPER:INST:ENAB?
> 128<END>
```

**STATus:OPERation:INStrument[:EVENt]?**

**Function**
Queries the instrument operation event register.

**Syntax**
```
STATus:OPERation:INStrument[:EVENt]?
```

**Parameters**
None

**Response Data**
The bit value for the operation event register as a `short` value (0 .. +32767)

**Example of Use**
```
STAT:OPER:INST?
> 0<END>
```

**STATus:OPERation:INStrument:ISUMmary<n>:CONDition?**

**Function**
Queries the instrument operation condition register of the specified instrument.
The value of `<n>` is restricted from 1 to 14 and represents the logical instruments id assigned to the SCPI controlled instrument by the INStrument subsystem.

**Syntax**
```
STATus:OPERation:INStrument:ISUMmary<n>:CONDition?
```
Parameters
None

Response Data
The bit value for the operation condition register as a **short** value (0 .. +32767)

Example of Use
STAT:OPER:INST:ISUM4:COND?
> 16<END>

STATus:OPERation:INStrument:ISUMmary<n>:ENABle
Function
Sets the instrument operation enable mask for the event register of the specified instrument. The value of <n> is restricted from 1 to 14 and represents the logical instruments id assigned to the SCPI controlled instrument by the INSTRument subsystem.

Syntax
STATus:OPERation:INStrument:ISUMmary<n>:ENABle<wsp><value>

Parameters
The bit value for the operation enable mask as a **short** value (0 .. +32767)

Response Data
None

Example of Use
STAT:OPER:INST:ISUM1:ENAB 128

STATus:OPERation:INStrument:ISUMmary<n>:ENABle?
Function
Queries the instrument operation enable mask for the event register of the specified instrument. The value of <n> is restricted from 1 to 14 and represents the logical instruments id assigned to the SCPI controlled instrument by the INSTRument subsystem.
3.3 Status Subsystem Commands

Syntax
STATus:OPERation:INSTrument:ISUMmary<n>ENABle?

Parameters
None

Response Data
The bit value for the operation enable mask as a short value (0 .. +32767)

Example of Use
STAT:OPER:INST:ISUM4:ENAB?
> 128<END>

STATus:OPERation:INSTrument:ISUMmary<n>[:EVENt]?
Function
Queries the instrument operation event register of the specified instrument. The value of <n> is restricted from 1 to 14 and represents the logical instruments id assigned to the SCPI controlled instrument by the INSTrument subsystem.

Syntax
STATus:OPERation:INSTrument:ISUMmary<n>[:EVENt]?

Parameters
None

Response Data
The bit value for the operation event register as a short value (0 .. +32767)

Example of Use
STAT:OPER:INST:ISUM3?
> 0<END>

STATus:QUESTionable[:EVENt]?
Function
Queries the questionable event register.
Syntax
STATus:QUEStionable[:EVENt]?  

Parameters
None

Response Data
The bit value for the questionable event register as a short value (0 .. +32767)

Example of Use
STAT:QUES?
> 0<END>

STATus:QUEStionable:CONDition?
Function
Queries the questionable condition register.

Syntax
STATus:QUEStionable:CONDition?

Parameters
None

Response Data
The bit value for the questionable condition register as a short value (0 .. +32767)

Example of Use
STAT:QUES:COND?
> 8<END>

STATus:QUEStionable:BIT<n>:CONDition?
Function
Queries the questionable condition register specified bit. The value of <n> is restricted from 9 to 12 and represents bits 9 through 12 in the operation status register.

Syntax
STATus:QUEStionable:BIT<n>:CONDition?
3.3 Status Subsystem Commands

Parameters
None

Response Data
The bit value for the questionable condition register as a short value (0 .. 1)

Example of Use
STAT:QUES:BIT9:COND?
> 0<END>

STATus:QUEStionable:BIT<n>:ENABle

Function
Sets the questionable enable mask for the event register specified bit.
The value of <n> is restricted from 9 to 12 and represents bits 9 through 12 in the questionable status register.

Syntax
STATus:QUEStionable:BIT<n>:ENABle<wsp><value>

Parameters
The bit value for the questionable enable mask as a short value (0 .. 1)

Response Data
None

Example of Use
STAT:QUES:BIT11:ENAB 1

STATus:QUEStionable:BIT<n>:ENABle?

Function
Queries the questionable enable mask for the event register specified bit.
The value of <n> is restricted from 9 to 12 and represents bits 9 through 12 in the questionable status register.

Syntax
STATus:QUEStionable:BIT<n>:ENABle?
Parameters
None

Response Data
The bit value for the questionable enable mask as a short value (0 .. 1)

Example of Use
STAT:QUES:BIT10:ENAB?
> 1<END>

STATus:QUEStionable:BIT<n>[:EVENt]?
Function
Queries the questionable event register specified bit. The value of <n> is restricted from 9 to 12 and represents bits 9 through 12 in the questionable status register.

Syntax
STATus:QUEStionable:BIT<n>[:EVENt]?

Parameters
None

Response Data
The bit value for the questionable event register as a short value (0 .. 1)

Example of Use
STAT:QUES:BIT9:EVEN?
> 0<END>

STATus:QUEStionable:ENABle
Function
Sets the questionable enable mask for the event register.

Syntax
STATus:QUEStionable:ENABle<wsp><value>
3.3  Status Subsystem Commands

Parameters
The bit value for the questionable enable mask as a `short` value (0 .. +32767)

Response Data
None

Example of Use
```
STAT:QUES:ENAB 128
```
value (0 .. +32767)

**Example of Use**

```
STAT:QUES:INST:COND?
> 8<END>
```

**STATus:QUEStionable:INSTrument:ENABle**

**Function**
Sets the questionable instrument enable mask for the event register.

**Syntax**

```
STATus:QUEStionable:INSTrument:ENABle <value>
```

**Parameters**
The bit value for the questionable enable mask as a **short** value (0 .. +32767)

**Response Data**
None

**Example of Use**

```
STAT:QUES:INST:ENAB 128
```

**STATus:QUEStionable:INSTrument:ENABle?**

**Function**
Queries the questionable instrument enable mask for the event register.

**Syntax**

```
STATus:QUEStionable:INSTrument:ENABle?
```

**Parameters**
None

**Response Data**
The bit value for the questionable enable mask as a **short** value (0 .. +32767)
Example of Use
STAT:QUES:INST:ENAB?
> 128<END>

STATus:QUESTionable:INSTRument:EVENt?

Function
Queries the questionable instrument event register.

Syntax
STATus:QUESTionable:INSTRument:EVENt?

Parameters
None

Response Data
The bit value for the questionable event register as a short value (0 .. +32767)

Example of Use
STAT:QUES:INST:EVEN?
> 0<END>

STATus:QUESTionable:INSTRument:ISUMmary<n>:CONDition?

Function
Queries the specified logical instrument questionable instrument condition register.
The value of <n> is restricted from 1 to 14 and represents the logical instruments id assigned to the SCPI controlled instrument by the INSTRument subsystem.

Syntax
STATus:QUESTionable:INSTRument:ISUMmary<n>:CONDition?

Parameters
None

Response Data
The bit value for the questionable condition register as a short value (0 .. +32767)
Example of Use
STAT:QUES:INST:ISUM2:COND?
> 0<END>

STATus:QUEStionable:INStrument:ISUMmary<n>:ENABle

Function
Sets the questionable instrument enable mask for the event register of the specified logical instrument.
The value of <n> is restricted from 1 to 14 and represents the logical instruments id assigned to the SCPI controlled instrument by the INSTrument subsystem.

Syntax
STATus:QUEStionable:INStrument:ISUMmary<n>:ENABle<wsp><value>

Parameters
The bit value for the questionable enable mask as a short value (0 .. +32767)

Response Data
None

Example of Use
STAT:QUES:INST:ISUM3:ENAB 128

STATus:QUEStionable:INStrument:ISUMmary<n>:ENABle?

Function
Queries the questionable instrument enable mask for the event register of the specified logical instrument.
The value of <n> is restricted from 1 to 14 and represents the logical instruments id assigned to the SCPI controlled instrument by the INSTrument subsystem.

Syntax
STATus:QUEStionable:INStrument:ISUMmary<n>:ENABle?

Parameters
None
3.3 Status Subsystem Commands

Response Data
The bit value for the questionable enable mask as a **short** value (0 .. +32767)

Example of Use
STAT:QUES:INST:ISUM3:ENAB?
> 136<END>

STATus:QUEStionable:INStrument:ISUMmary<n>[:EVENt]?
Function
Queries the questionable instrument event register of the specified logical instrument.
The value of <n> is restricted from 1 to 14 and represents the logical instruments id assigned to the SCPI controlled instrument by the INSTRument subsystem.

Syntax
STATus:QUEStionable:INStrument:ISUMmary<n>[:EVENt]?

Parameters
None

Response Data
The bit value for the questionable event register as a **short** value (0 .. +32767)

Example of Use
STAT:QUES:INST:ISUM4:EVEN?
> 0<END>

STATus:PRESet
Function
Resets both the operation event mask and questionable event mask to 0.

Syntax
STATus:PRESet
Parameters
None

Response Data
None

Example of Use
STAT:PRES
3.4 Instrument Subsystem Commands

**INSTRument:CATalog?**

**Function**
Queries the list of instruments on the ACCESS Master that are identified as SCPI controllable instruments.

**Syntax**
INSTRument:CATalog?

**Parameters**
None

**Response Data**
Comma-separated list of string identifiers of all SCPI controllable logical instruments
TOP_MENU    Top Menu
OTDR_STD    OTDR (Standard)

**Example of Use**
INST:CAT?
> TOP_MENU,OTDR_STD<END>

**INSTRument:CATalog:FULL?**

**Function**
Queries the list of instruments on the ACCESS Master that are identified as SCPI controllable instruments.

**Syntax**
INSTRument:CATalog:FULL?

**Parameters**
None

**Response Data**
Comma-separated list of string identifiers and numbers of all SCPI controllable logical instruments.
The string contains the name identifier of the logical instrument. The immediately following short value is an associated logical instrument number of the port. All response data elements are
comma separated.

**Example of Use**

```
INStrument:CATalog:FULL?
> TOP_MENU,1,OTDR_STD,2<END>
```

**INStrument:NSELect**

**Function**
Sets a numeric value identifier for the logical instrument specified as control target.

**Syntax**

```
INStrument:NSELect<wsp><num_id>
```

**Parameters**
Numeric value identifier for the logical instrument specified as control target (a short value)

**Response Data**
None

**Example of Use**

```
INST:NSEL 2
```

**INStrument:NSELect?**

**Function**
Queries the numeric value identifier of the logical instrument specified as control target.

**Syntax**

```
INStrument:NSELect?
```

**Parameters**
None

**Response Data**
Numeric value identifier for the logical instrument specified as control target (a short value)
Example of Use
INST:NSEL?
> 2<END>

INSTRument[:SELect]

Function
Selects the specified logical instrument as control target.

Syntax
INSTRument:SELect<wsp><string_id>

Parameters
String instrument identifier assigned by the Instrument subsystem for selecting the instrument

Response Data
None

Example of Use
INST:SEL OTDR_STD

INSTRument[:SELect]?

Function
Queries the string value identifier of the currently selected logical instrument.

Syntax
INSTRument:SELect?

Parameters
None

Response Data
String value identifier of the currently selected logical instrument.

Example of Use
INST:SEL?
> OTDR_STD<END>
INSTRument:STATe

Function
Turns the currently selected logical instrument state ON or OFF.

Syntax
INSTRument:STATe<boolean>

Parameters
Boolean format
On: 1 or ON
Off: 0 or OFF

Response Data
None

Example of Use
INSTR:STAT ON

INSTRument:STATe?

Function
Queries the state of the currently selected logical instrument.

Syntax
INSTRument:STATe?

Parameters
None

Response Data
State of logical instrument
Boolean format
1: On
0: Off

Example of Use
INSTR:STAT?
> 1<END>
3.5 TOPMenu Subsystem Commands

**TOPMenu:UNIT:CATalog?**

**Function**
Queries a list of ports installed on the ACCESS Master.

**Syntax**
TOPMenu:UNIT:CATalog?

**Parameters**
None

**Response Data**
Comma-separated list of string identifiers of all installed Ports

**Example of Use**
TOPM:UNIT:CAT?
> MM,SM<END>

**TOPMenu:UNIT:CATalog:FULL?**

**Function**
Queries a list of ports installed on the ACCESS Master.

**Syntax**
TOPMenu:UNIT:CATalog:FULL?

**Parameters**
None

**Response Data**
List of string identifiers and numbers of ports
The string contains the name identifier of the installed Ports. The immediately following NR1-formatted number is its associated logical port number. All response data elements are comma separated.

**Example of Use**
TOPM:UNIT:CAT:FULL?
> MM,1,SM,2<END>
Chapter 3  Platform SCPI Commands

TOPMenu:UNIT[:SELect]

Function
Sets the specified Port to be currently selected Port.

Syntax
TOPMenu:UNIT:SELect<wsp><string_id>

Parameters
The string instrument identifier assigned by the TOPMenu subsystem for the Port to be selected as a string value

Response Data
None

Example of Use
TOPM:UNIT:SEL SM

TOPMenu:UNIT[:SELect]?

Function
Queries the string value identifier of the currently selected Port.

Syntax
TOPMenu:UNIT:SELect?

Parameters
None

Response Data
String value identifier of the currently selected port.

Example of Use
TOPM:UNIT:SEL?
> MM<END>

TOPMenu:UNIT:NSELect

Function
Sets the specified Port to be currently selected to run with OTDR instrument.
3.5 TOPMenu Subsystem Commands

Syntax
TOPMenu:NSELect\<wsp>\<num_id>

Parameters
The numeric value identifier assigned by the TOPMenu subsystem for the Port to be selected as a short value

Response Data
None

Example of Use
TOPM:UNIT:NSEL 2

TOPMenu:UNIT:NSELect?

Function
Queries the numeric value identifier of the currently selected Port.

Syntax
TOPMenu:UNIT:NSELect?

Parameters
None

Response Data
Numeric value identifier of the currently selected port.

Example of Use
TOPM:UNIT:NSEL?
> 1<END>
Chapter 4   OTDR Commands

This chapter details the SCPI commands for the ACCESS Master Standard OTDR application. The Command Summary section presents a brief summary of each command, while each command is detailed in the subsequent sections.

4.1 Command Summary ...........................................4-2
   4.1.1 Root level commands .............................4-2
   4.1.2 Source Subsystem Commands ..............4-3
   4.1.3 Sense Subsystem Commands ...............4-5
   4.1.4 Trace Subsystem Commands ..............4-9
   4.1.5 Display Subsystem Commands ..........4-12
4.2 Root Level Commands .......................................4-13
4.3 SOURce Subsystem Commands ..........................4-17
4.4 SENSE Subsystem Commands ..........................4-27
4.5 TRACe Subsystem Commands ......................4-45
4.6 DISPlay Subsystem Commands .....................4-55
4.1 Command Summary

The Command Summary section provides a list of the SCPI commands for the ACCESS Master Standard OTDR application, and a brief description for each command.

Commands and queries in this section can be received only in the OTDR (Standard) mode.

4.1.1 Root level commands

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ABORt</td>
<td>Aborts active test, data is lost.</td>
</tr>
<tr>
<td>STOP</td>
<td>Stops the test, keeps the trace on screen.</td>
</tr>
<tr>
<td>INITiate</td>
<td>Start test with manual test settings.</td>
</tr>
<tr>
<td>INITiate:AUTo</td>
<td>Start auto-test.</td>
</tr>
<tr>
<td>INITiate:RTIMe</td>
<td>Start real time test with manual settings.</td>
</tr>
<tr>
<td>INITiate?</td>
<td>Queries if test is active.</td>
</tr>
</tbody>
</table>

Figure 4.1.1-1  Commands Related to Measurements
4.1.2 Source Subsystem Commands

SOURce:WAVelength:AVAilable? Queries available wavelength list.
SOURce:WAVelength Sets current selected wavelength.
SOURce:WAVelength? Queries current selected wavelength.
SOURce:RANge:AVAilable? Queries available ranges list.
SOURce:RANge Sets current selected range.
SOURce:RANge? Queries current selected range.
SOURce:RESo:AVAilable? Queries available resolutions list.
SOURce:RESo Sets current selected resolution.
SOURce:RESo? Queries current selected resolution.
SOURce:PULSe:AVAilable? Queries available pulse width list.
SOURce:PULSe Sets current selected pulse width.
SOURce:PULSe? Queries current selected pulse width.
SOURce:PULSe:ENHanced:AVAilable? Queries if Enhanced Range is available for the dead zone when using the selected pulse width.
SOURce:PULSe:ENHanced Sets Enhanced Range for the dead zone.
SOURce:PULSe:ENHanced? Queries if Enhanced Range is set for the dead zone.
SOURce:AVERages:TIMe Sets averaging time for the next test.
SOURce:AVERages:TIMe? Queries averaging time for the next test.

Figure 4.1.2-1 Commands Related to Wavelength
Chapter 4  
OTDR Commands

SOURce:RANge:AVAilable?  SOURce:RESo:AVAilable?  SOURce:PULSe:AVAilable?
SOURce:RANge  SOURce:RESo  SOURce:PULSe?
SOURce:PULSe:ENHanced:AVAilable?
SOURce:PULSe:ENHanced
SOURce:PULSe:ENHanced?

Figure 4.1.2-2  Commands Related to Measurement Condition

SOURce:AVERages:TIMe
SOURce:AVERages:TIMe?

Figure 4.1.2-3  Commands Related to Average Time
4.1.3 Sense Subsystem Commands

SENSe:AVERages? Queries averages count since test started.
SENSe:AVERages:TIMe? Queries averaging time since test started.
SENSe:TRACe:READY? Queries if the trace data is ready.
SENSe:CONCheck Sets connection check option ON or OFF.
SENSe:CONCheck? Queries if the connection check option enabled.
SENSe:LIVCheck Sets live fiber check option ON or OFF.
SENSe:LIVCheck? Queries if the live fiber check option enabled.
SENSe:FIBer:IOR Sets fiber IOR value.
SENSe:FIBer:IOR? Queries fiber IOR value.
SENSe:FIBer:BSC Sets fiber BSC value.
SENSe:FIBer:BSC? Queries fiber BSC value.
SENSe:Hoff set Sets horizontal offset value.
SENSe:Hoff set? Queries horizontal offset value.
SENSe:VOFFset Sets vertical offset value.
SENSe:VOFFset? Queries vertical offset value.
SENSe:LSALeft Sets left LSA maker position values.
SENSe:LSALeft? Queries left LSA maker position values.
SENSe:LSARight Sets right LSA maker position values.
SENSe:LSARight? Queries right LSA maker values.
SENSe:ACURsor Sets A cursor position.
SENSe:ACURsor? Queries A cursor position value.
SENSe:BCURsor Sets B cursor position.
SENSe:BCURsor? Queries B cursor position value.
SENSe:LOSS:MODE Sets current loss mode.
SENSe:LOSS:MODE? Queries currently selected loss mode.
SENSe:ORL:MODE Sets current ORL Mode.
SENSe:ORL:MODE? Queries current ORL Mode.
SENSe:ANALyze:PARameters Sets auto detection parameters values.
SENSe:ANALyze:PARameters? Queries auto detection parameters values.
SENSe:ANALyze:AUTO Sets trace auto analysis option ON/OFF.
SENSe:ANALyze:AUTO? Queries if the trace auto analysis option is on.
Chapter 4  OTDR Commands

### Figure 4.1.3-1  Commands for Preferences (1-2)

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>SENSe:CONCheck</td>
<td>Connection Check</td>
<td>Off</td>
</tr>
<tr>
<td>SENSe:CONCheck?</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SENSe:LIVCheck</td>
<td>Live Check</td>
<td>On</td>
</tr>
<tr>
<td>SENSe:LIVCheck?</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SENSe:ORL:MODE</td>
<td>OLR Mode</td>
<td>On</td>
</tr>
<tr>
<td>SENSe:ORL:MODE?</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SENSe:ANALyze:AUTO</td>
<td>Auto Analysis</td>
<td>On</td>
</tr>
<tr>
<td>SENSe:ANALyze:AUTO?</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Figure 4.1.3-2  Commands for Preferences (2-2)

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Marker Mode</td>
<td>Type of reflective result</td>
<td>Reflectance</td>
</tr>
<tr>
<td>Auto Patch-panel Removal</td>
<td>None/None</td>
<td></td>
</tr>
<tr>
<td>Force Total Loss</td>
<td>Off</td>
<td></td>
</tr>
<tr>
<td>End Event for OLR Calculation</td>
<td>OMIT</td>
<td></td>
</tr>
<tr>
<td>OTDR (Standard)</td>
<td>Auto Analysis</td>
<td>On</td>
</tr>
<tr>
<td>Bi-Directional Correlation</td>
<td>2.000 %</td>
<td></td>
</tr>
<tr>
<td>Continuous Pulse Emission</td>
<td>Off</td>
<td></td>
</tr>
</tbody>
</table>
4.1 Command Summary

**Figure 4.1.3-3** Commands for Thresholds

<table>
<thead>
<tr>
<th>Thresholds</th>
<th>2019-Jul-24 18:04</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Auto Detect</strong></td>
<td>0.05 dB</td>
</tr>
<tr>
<td>Splice Loss</td>
<td>-0.0 dB</td>
</tr>
<tr>
<td>Reflectance</td>
<td>3 dB</td>
</tr>
<tr>
<td>Fiber End</td>
<td>0.3 dB</td>
</tr>
<tr>
<td>Macro Bend</td>
<td>1 dB (0.0 dB)</td>
</tr>
</tbody>
</table>

**Pass/Fail Thresholds**
- Non-Reflective Event Loss (fusion): None
- Reflective Event Loss (connector/mechanical): None
- Reflectance: None
- Fiber Loss (dB/km): None
- Total Loss: None
- Splitter Loss: None

**SENSe:ANALyze:PARameters**
- **SENSe:ANALyze:PARameters?**

**Figure 4.1.3-4** Commands for IOR and BSC

**SENSe:FIBer:IOR**
- **SENSe:FIBer:IOR?**

**SENSe:FIBer:BSC**
- **SENSe:FIBer:BSC?**

**SENSe:HOFFset**
- **SENSe:HOFFset?**

**SENSe:VOFFset**
- **SENSe:VOFFset?**

**Figure 4.1.3-5** Commands for Offset
Chapter 4  OTDR Commands

Figure 4.1.3-6  Commands for Averaging

SENSe:AVERages?
SENSe:AVERages:TIMe?

Figure 4.1.3-7  Commands for Cursors

SENSe:LSALeft
SENSe:LSALeft?
SENSe:LSARight
SENSe:LSARight?
4.1 Command Summary

4.1.4 Trace Subsystem Commands

- **TRACe:PARameters?** Queries trace parameters summary in text format.
- **TRACe:ANALyze** Performs analysis on the trace.
- **TRACe:ANALyze?** Queries if the trace is analyzed.
- **TRACe:ANALyze:ORL** Performs ORL calculations on the trace.
- **TRACe:MDLOss?** Queries trace loss value for current loss mode.
- **TRACe:EELoss?** Queries trace end-to-end loss value.
- **TRACe:LOAD:SOR?** Queries trace data in SOR file format.
- **TRACe:LOAD:TEXT?** Queries trace data in ASCII text format.
- **TRACe:LOAD:DATA?** Queries trace data points in binary format.
- **TRACe:HEADER** Sets Trace Header.
- **TRACe:HEADER?** Queries Trace Header.
- **TRACe:STORe:SOR** Stores the SOR file in internal memory of the ACCESS Master.
Chapter 4  OTDR Commands

Figure 4.1.4-2  Commands for Analysis

```
TRACe:EELOss?
TRACe:ANALyze?
TRACe:ANALyze:ORL
TRACe:STORe:SOR
```

Figure 4.1.4-3  Command for Save Trace
Figure 4.1.4-4  Commands for Header
4.1.5 Display Subsystem Commands

**DISPlay:MODE** Selects the reference cursor A or B.

**DISPlay:MODE?** Queries which reference cursor is selected.

**DISPlay:ZOOM:FULL** Sets zoom to view full trace in current display mode.

**DISPlay:ZOOM:HORIzontal** Sets Horizontal display zoom to specified level.

**DISPlay:ZOOM:HORIzontal?** Queries Horizontal display zoom level.

**DISPlay:ZOOM:VERTical** Sets Vertical display zoom to specified level.

**DISPlay:ZOOM:VERTical?** Queries Vertical display zoom level.

**DISPlay:SCAlE:HORIzontal** Sets Horizontal scale range to specified value.

**DISPlay:SCAlE:HORIzontal?** Queries Horizontal display scale range value.

**DISPlay:SCAlE:VERTical** Sets Vertical scale range to specified value.

**DISPlay:SCAlE:VERTical?** Queries Vertical display scale range value.

---

**Figure 4.1.5-1  Commands for Display**
4.2 Root Level Commands

ABORt

Function
Aborts the active test. The trace data will be lost.

Syntax
ABORt

Parameters
None

Response Data
None

Errors
(–200, "std_execGen, Test is Inactive")
(–200, "std_execGen, Instrument is Busy")

Example of Use
Abor

STOP

Function
Stops the active test. The trace data will be preserved.

Syntax
STOP

Parameters
None

Response Data
None

Errors
(–200, "std_execGen, Test is Inactive")
(–200, "std_execGen, Instrument is Busy")
Example of Use
Stop

INITiate

Function
Starts OTDR test with currently selected settings. Manual test mode.
This command is an overlapped command.
When this command is executed with “Wavelength all” selected, the measurement starts with the minimum wavelength, and no measurement is done for other wavelengths.

Syntax
INITiate

Parameters
None

Response Data
None

Errors
(–200, "std_execGen, Test is Active")
(–200, "std_execGen, Instrument is Busy")
(–200, "std_execGen, Start Test Failed")
(–200, "std_execGen, Connection Check Failed")
(–200, "std_execGen, Live Fiber Check Failed")

Example of Use
init

INITiate:AUTo

Function
Starts OTDR auto-test. All test parameters are automatically obtained during the pre-scan. This command is an overlapped command.
When this command is executed with “Wavelength all” selected, the measurement starts with the minimum wavelength, and no measurement is done for other wavelengths.
4.2 Root Level Commands

OTDR Commands

4.2.1 Syntax

INITiate:AUTo

Parameters
None

Response Data
None

Errors
(–200, "std_execGen, Test is Active")
(–200, "std_execGen, Instrument is Busy")
(–200, "std_execGen, Start Test Failed")
(–200, "std_execGen, Connection Check Failed")
(–200, "std_execGen, Live Fiber Check Failed")

Example of Use
init:auto

INITiate:RTIMe

Function
Starts OTDR real time-test. Starts OTDR real time test with currently selected settings. This command is an overlapped command.
When this command is executed with “Wavelength all” selected, the measurement starts with the minimum wavelength, and no measurement is done for other wavelengths.

Syntax
INITiate:RTIMe

Parameters
None

Response Data
None

Errors
(–200, "std_execGen, Test is Active")
(–200, "std_execGen, Instrument is Busy")
Chapter 4  OTDR Commands

(--200, "std_execGen, Start Test Failed")
(--200, "std_execGen, Connection Check Failed")
(--200, "std_execGen, Live Fiber Check Failed")

Example of Use
init:rtim

INITiate?
Function
Queries if test is initiated.

Syntax
INITiate?

Parameter
None

Response Data
Possible response range: 0 | 1
0: Test is NOT active.
1: Test is currently active.

Errors
None

Example of Use
init?
> 0 <End>
4.3 SOURce Subsystem Commands

The SOURce subsystem controls/query OTDR’s optical source parameters.

SOURce:WAVelength:AVAvailable?

**Function**
Queries the list of available wavelengths.

**Syntax**
SOURce:WAVelength:AVAvailable?

**Parameter**
None

**Response Data**
List of available wavelength in nanometers (nm).

**Errors**
None

**Example of Use**
```plaintext
sour:wav:ava?
> 1310, 1550<END>
```

SOURce:WAVelength

**Function**
Sets current wavelength. Wavelength value units – nm. Wavelength will be set only if the new value matches the one in the list of the available wavelengths.

**Syntax**
SOURce:WAVelength< wsp><value>

**Parameters**
<value>
Integer value format
Range: Integer WL value returned by available wavelengths query command.
Wavelength “ALL” cannot be selected.
**Chapter 4  OTDR Commands**

Response Data
None

**Errors**

(–200, "std_execGen, Test is Active")
(–224, "std_illegalParmValue, Invalid Parameter Value")

**Example of Use**

sour:wave 1310

SOURce:WAVelength?

**Function**
Queries current wavelength.
Available first wavelength will be returned when “Wavelength ALL” is set.

**Syntax**

SOURce:WAVelength?

**Parameter**
None

**Response Data**
Current wavelength value.

**Errors**
None

**Example of Use**

sour:wav?
> 1310<END>

SOURce:RANge:AVAilable?

**Function**
Queries the list of available ranges for current wavelength settings.

**Syntax**

SOURce:RANge:AVAilable?
4.3 SOURce Subsystem Commands

Parameters
None

Response Data
List of available ranges in kilometers (km).

Errors
None

Example of Use
sour:ran:ava?
> 5.0, 10.0, 20.0, 50.0, 100.0, 200.0, 300.0<END>

SOURce:RANge
Function
Sets current range.

Syntax
SOURce:RANge<wsp><value>

Parameters
<value>
Numeric format
Range: Model dependent, numeric value returned by available ranges query command

Response Data
None

Errors
(–200, "std_execGen, Test is Active")
(–224, "std_illegalParmValue, Invalid Parameter Value")

Example of Use
sour:ran 100

SOURce:RANge?
Function
Queries current range.
Chapter 4  OTDR Commands

Syntax
SOURce:RANge?

Parameters
None

Response Data
The value is current range in kilometers (km).

Errors
None

Example of Use
sour:ran?
> 50.0<END>

SOURce:RESo:AVAilable?

Function
Queries the list of available resolution flags for current range settings.

Syntax
SOURce:RESo:AVAilable?

Parameter
None

Response Data
List of available resolutions:
0: Low Density
1: High Density
2: Very High Density

Errors
None

Example of Use
sour:res:ava?
> 0, 1, 2<END>
SOURce:RESo

Function
Sets current resolution.

Syntax
SOURce:RESo<wsp><value>

Parameters
[value]
Integer format
Range: 0 | 1 | 2
0: Low Density
1: High Density
2: Very High Density
Available resolution values are dependant on current range settings.

Response Data
None

Errors
(–200, "std_execGen, Test is Active")
(–224, "std_illegalParmValue, Invalid Parameter Value")

Example of Use
sour:res 1

SOURce:RESo?

Function
Queries current resolution settings flag.

Syntax
SOURce:RESo?

Parameters
None

Response Data
The value is current resolution:
0: Low Density
Chapter 4  OTDR Commands

1: High Density
2: Very High Density

Errors
None

Example of Use
sour:res?
> 1<END>

SOURce:PULSe:AVAilable?
Function
Queries the list of available pulse width for current range/resolution settings.

Syntax
SOURce:PULSe:AVAilable?

Parameters
None

Response Data
List of available pulse width in nanoseconds (ns).

Errors
None

Example of Use
sour:puls:ava?
> 10,20,50,100<END>

SOURce:PULSe
Function
Sets current pulse width.

Syntax
SOURce:PULSe<ws><value>

Parameters
<value>
Numeric format
Range: Integer PW value returned by querying available pulse width query command.
Available pulse width values are dependant on current range/resolution settings.

Response Data
None

Errors
(–200, "std_execGen, Test is Active")
(–224, "std_illegalParmValue, Invalid Parameter Value")

Example of Use
sour:puls 100

SOURce:PULSe?
Function
Queries current pulse width.

Syntax
SOURce:PULSe?

Parameter
None

Response Data
The value is current pulse width in nanoseconds (ns).

Errors
None

Example of Use
sour:puls?
> 100<END>

SOURce:PULSe:ENHanced:AVAilable?
Function
Queries if Enhanced Range is available for the dead zone when using the selected pulse width.
**Syntax**

SOURce:PULSe:ENHanced:AVAilable?

**Parameters**

None

**Response Data**

Boolean format

1 or 0.

0: Unavailable

1: Available

**Errors**

None

**Example of Use**

sour:puls:enh:ava?

> 1<END>

**SOURce:PULSe:ENHanced**

**Function**

Sets current pulse width’s Dead-zone setting.

**Syntax**

SOURce:PULSe:ENHanced<wsp><value>

**Parameter**

/value>

Boolean format

Range: 1|0.

0: Standard (HR)

1: Enhanced Range (ER)

**Response Data**

None

**Errors**

(–200, "std_execGen, Test is Active")

(–104, "std_wrongParamType, Data Type Error")
4.3 SOURce Subsystem Commands

(–224, "std_illegalParmValue, Invalid Parameter Value")

Example of Use
sour:puls:enh 0

SOURce:PULSe:ENHanced?
Function
Queries current pulse width’s Dead-zone setting.

Syntax
SOURce:PULSe:ENHanced?

Parameters
None

Response Data
Boolean format
0: Standard (HR)
1: Enhanced Range (ER)

Errors
None

Example of Use
sour:puls:enh?
> 1<END>

SOURce:AVERages:TIMe
Function
Sets number of seconds for the test duration for the next test in manual mode.

Syntax
SOURce:AVERages:TIMe<wsp><value>

Parameters
<value>
Integer format
Range: 1 to 3600.

Response Data
None

Errors
(–200, "std_execGen, Test is Active")
(–224, "std_illegalParmValue, Invalid Parameter Value")

Example of Use
sour:aver:tim 120<END>

SOURce:AVERages:TIMe?

Function
Queries number of seconds that have been set for the next test in manual mode.

Syntax
SOURce:AVERages:TIMe?

Parameters
None

Response Data
Number of seconds set to run the test.

Errors
None

Example of Use
sour:aver:tim?
> 120<END>
4.4 SENSe Subsystem Commands

SENSe:AVERages?

Function
Queries the averages count since the test started.

Syntax
SENSe:AVERages?

Parameters
None

Response Data
Number of averages

Errors
(–400, "std_queryGen, Trace Not Ready")

Example of Use
sens:aver?
> 4096<END>

SENSe:AVERages:TIMe?

Function
Queries number of seconds that have been completed on the trace or since the test started.
Actual average time may be longer than the set average time depending on the resolution and distance range setting.

Syntax
SENSe:AVERages:TIMe?

Parameters
None

Response Data
Number of seconds

Errors
(–400, "std_queryGen, Trace Not Ready")
Example of Use
sens:aver:tim?
> 28<END>

SENSe:TRACe:READY?

Function
Queries if trace data is ready.

Syntax
SENSe:TRACe:READY?

Parameters
None

Response Data
Possible response values:
1: Trace data is ready and can be transferred.
0: No trace data available in the memory.

Errors
None

Example of Use
sens:trac:ready?
> 1<END>

SENSe:CONCheck?

Function
Queries if connection check option is ON.

Syntax
SENSe:CONCheck?

Parameters
None

Response Data
Possible response values:
1: Connection check is ON.
0: Connection check is OFF.

**Errors**
None

**Example of Use**
```
sens:conc?
> 1<END>
```

**SENSe:CONCheck**

**Function**
Sets Connection Check option ON or OFF.

**Syntax**
```
SENSe:CONCheck<wsp><value>
```

**Parameters**

<value>
Boolean format
1 or on: Connection check is ON.
0 or off: Connection check is OFF.

**Response Data**
None

**Errors**

(–200, "std_execGen, Test is Active")
(–104, "std_wrongParamType, Data Type Error")

**Example of Use**
```
sens:conc ON<END>
```

**SENSe:LIVCheck?**

**Function**
Queries if live fiber check option is ON.

**Syntax**
```
SENSe:LIVCheck?
```
Parameters
None

Response Data
Possible response values:
1: Live fiber check is ON.
0: Live fiber check is OFF.

Errors
None

Example of Use
sens:livc?
> 1<END>Abor

SENSe:LIVCheck
Function
Sets live fiber check option ON or OFF.

Syntax
SENSe:LIVCheck<wsp><value>

Parameters
<value>
Boolean format
1 or on: Live fiber check is ON.
0 or off: Live fiber check is OFF.

Response Data
None

Errors
(–200, "std_execGen, Test is Active")
(–104, "std_wrongParamType, Data Type Error")

Example of Use
sens:livc ON<END>
SENSe:FIBer:IOR

**Function**
Sets IOR (index of refraction). This value will be used for the next test.

**Syntax**
SENSe:FIBer:IOR<sp><value>

**Parameters**

<value>
Floating point format
Range: 1.3 to 1.7

**Response Data**
None

**Errors**

(–200, "std_execGen, Test is Active")
(–224, "std_illegalParmValue, Invalid Parameter Value")

**Example of Use**
sens:fib:ior 1.45

SENSe:FIBer:IOR?

**Function**
Queries IOR.

**Syntax**
SENSe:FIBer:IOR?

**Parameters**
None

**Response Data**
Possible response is value in range:
1.3 to 1.7

**Errors**

(–200, "std_execGen, Test is Inactive")
Example of Use
sens:fib:ior?
> 1.450000<END>

SENSe:FIBer:BSC
Function
Sets BSC (backscatter coefficient). This value will be used for the next test.

Syntax
SENSe:FIBer:BSC<wsp><value>

Parameters
<value>
Floating point format
Range: –90.0 to –40.0

Response Data
None

Errors
(–200, "std_execGen, Test is Active")
(–224, "std_illegalParmValue, Invalid Parameter Value")

Example of Use
sens:fib:bsc –83.0

SENSe:FIBer:BSC?
Function
Queries BSC.

Syntax
SENSe:FIBer:BSC?

Parameters
None

Response Data
Possible response is value in range:
–90.0 to –40.0
4.4 SENSE Subsystem Commands

Errors
None

Example of Use
sens:fib:bsc?
> -83.0<END>

SENSe:HOFFset
Function
Set horizontal offset for the displayed trace(s). Offset value units – km.

Syntax
SENSe:HOFFset<wsp><value>

Parameters
,value>
Floating point format
Range: Offset value can be set plus/minus maximum distance range.

Response Data
None

Errors
(–224, "std_illegalParmValue, Invalid Parameter Value")

Example of Use
sens:hoff 10.0

SENSe:HOFFset?
Function
Queries horizontal offset for the displayed trace(s).

Syntax
SENSe:HOFFset?

Parameters
None
Response Data
Current horizontal offset value.

Errors
None

Example of Use
sens:hoff?
> 0<END>

SENSe:VOFFset
Function
Sets vertical offset for the displayed trace(s). Offset value units – dB.

Syntax
SENSe:VOFFset<wsp><value>

Parameters
<value>
Floating point format
Range: Offset value can be set plus/minus current dynamic range.
(–64.0 to 64.0)

Response Data
None

Errors
(–224, "std_illegalParmValue, Invalid Parameter Value")

Example of Use
sens:voff -5.0

SENSe:VOFFset?
Function
Queries vertical offset for the displayed trace(s).

Syntax
SENSe:VOFFset?
4.4 SENSE Subsystem Commands

Parameters
None

Response Data
Current vertical offset value.

Errors
None

Example of Use
sens:voff?
> 0<END>

SENSe:ACURsor
Function
Sets a cursor position. Cursor position units – km.
Note:
If remote-controlled by SCPI command, the Marker Mode is forcibly changed to Movement.

Syntax
SENSe:ACURsor<wsp><value>

Parameters
<value>
Floating point format
Range: 0.0 to Current distance range.

Response Data
None

Errors
(–224, "std_illegalParmValue, Invalid Parameter Value")

Example of Use
sens:acur 20.5
Chapter 4  OTDR Commands

SENSe:ACURsor?

Function
Queries current A cursor position.

Syntax
SENSe:ACURsor?

Parameters
None

Response Data
Current A cursor position value.

Errors
None

Example of Use
sens:acur?
> 20.5<END>

SENSe:BCURsor

Function
Sets B cursor position. Cursor position units – km.

Note:
If remote-controlled by SCPI command, the Marker Mode is forcibly changed to Movement.

Syntax
SENSe:BCURsor<wsp><value>

Parameters
<value>
Floating point format
Range: 0.0 to Current distance range.

Response Data
None

Errors
(–224, "std_illegalParmValue, Invalid Parameter Value")
Example of Use
sens:bcur 20.5

SENSe:BCURsor?
  **Function**
  Queries current B cursor position.

  **Syntax**
  SENSe:BCURsor?

  **Parameters**
  None

  **Response Data**
  Current B cursor position value.

  **Errors**
  None

  **Example of Use**
  sens:bcur?
  > 20.5<END>

SENSe:LSALeft
  **Function**
  Sets start and stop for left LSA marker. Start and stop units – km.
  **Note:**
  If remote-controlled by SCPI command, the Marker Mode is forcibly changed to Movement.

  **Syntax**
  SENSe:LSALeft<wsp><start>,<stop>

  **Parameters**
  <start>
  Floating point format
  Range: –100.0 to 400.0.
  <stop>
  Floating point format
Range: –100.0 to 400.0.
Start value must be less or equal to stop value.

**Response Data**
None

**Errors**
(–200, "std_execGen, LSA Inactive State")
(–224, "std_illegalParmValue, Invalid Parameter Value")

**Example of Use**
sens:lsal 0.0,0.5

**SENSe:LSALeft?**

**Function**
Queries left LSA values.

**Syntax**
SENSe:LSALeft?

**Parameters**
None

**Response Data**
Start and stop for left LSA marker. Start and stop units – km.

**Errors**
(–400, "std_queryGen, LSA Inactive State")

**Example of Use**
sens:lsal?
> 0.0,0.5<END>

**SENSe:LSARight**

**Function**
Sets start and stop for right LSA marker. Start and stop units – km.

**Note:**
If remote-controlled by SCPI command, the Marker Mode is forcibly changed to **Movement**.
### 4.4 SENSE Subsystem Commands

#### SENSE:LSARight \(<\text{start}>,\text{<stop}>\)

**Syntax**

SENSe:LSARight\(<\text{start}>,\text{<stop}>\)

**Parameters**

- **\(<\text{start}>\)**
  - Floating point format
  - Range: –100.0 to 400.0.
- **\(<\text{stop}>\)**
  - Floating point format
  - Range: –100.0 to 400.0.
  - Start value must be less or equal to stop value.

**Response Data**

None

**Errors**

- (–200, "std_execGen, LSA Inactive State")
- (–224, "std_illegalParmValue, Invalid Parameter Value")

**Example of Use**

sens:lsal 0.0,0.5

SENSe:LSARight?

**Function**

Queries right LSA values.

**Syntax**

SENSe:LSARight?

**Parameters**

None

**Response Data**

Start and stop for right LSA marker. Start and stop units – km.

**Errors**

(–400, "std_queryGen, LSA Inactive State")
Example of Use
sens:lsar?
> 0.0, 0.5<END>

SENSe:LOSS:MODE
Function
Sets current Loss Mode.

Syntax
SENSe:LOSS:MODE<wsp><value>

Parameters
[value>
Integer format
Range: 0|1|2|3|4|5|6
0: Splice Loss
1: 2-Pt Loss
2: 2-Pt LSA
3: dB/km Loss
4: dB/km LSA
5: 2-Pt, dB/km
6: ORL

Response Data
None

Errors
(–224, "std_illegalParmValue, Invalid Parameter Value")

Example of Use
sens:loss:mode 0

SENSe:LOSS:MODE?
Function
Queries current Loss Mode.

Syntax
SENSe:LOSS:MODE?
4.4 SENSe Subsystem Commands

Parameters
None

Response Data
Possible responses are:
0: Splice Loss
1: 2-Pt Loss
2: 2-Pt LSA
3: dB/km Loss
4: dB/km LSA
5: 2-Pt, dB/km
6: ORL

Errors
None

Example of Use
sens:loss:mode?
> 0<END>

SENSe:ORL:MODE
Function
Sets current ORL Mode.

Syntax
SENSe:ORL:MODE<wsp><value>

Parameters
<value>
Integer format
Range: 0|1|2
0: A Cursor
1: Origin
2: Full Trace

Response Data
None

Errors
(–224, "std_illegalParmValue, Invalid Parameter Value")
Example of Use
 sensible:loss:mode 0

SENSe:ORL:MODE?
 Function
 Queries current ORL Mode.

Syntax
 SENSe:ORL:MODE?

Parameters
 None

Response Data
 Possible responses are:
 0: A Cursor
 1: Origin
 2: Full Trace

Errors
 None

Example of Use
 sensible:orl:mode?
 > 0<END>

SENSe:ANALyze:PARameters
 Function
 Sets Auto detection Parameters for the next test.

Syntax
 SENSe:ANALyze:PARameters<spliceloss>,<reflectance>,<endloss>,<pon loss>

Parameters
 Double values:
 <spliceloss> Range: 0.01 to 9.99
 <reflectance> Range: –70.0 to –20.0
 <endloss> Range: 1 to 99
4.4 SENSE Subsystem Commands

<pon loss>  Range: 1.0 to 30.0

Response Data
None

Errors
(–224, "std_illegalParmValue, Invalid Parameter Value")

Example of Use
sens:anal:par 0.05,–60.0,3.0,10.0

SENSe:ANALyze:PARameters?
Function
Queries current Auto detection parameters.

Syntax
SENSe:ANALyze:PARameters?

Parameters
None

Response Data
<event loss>,<reflectance>,<end loss>,<pon loss><END>

Errors
None

Example of Use
sens:anal:par?
> 0.050000,–60.000000,3.000000,10.000000<END>

SENSe:ANALyze:AUTO
Function
Sets ON/OFF analysis to be performed automatically after the test is complete.

Syntax
SENSe:ANALyze:AUTO<wsp><value>
**Parameters**

<value>

Boolean format

Range: 0|1

0: Auto-analysis is OFF.
1: Auto-analysis is ON.

**Response Data**

None

**Errors**

(–104, "std_wrongParamType, Data Type Error")

**Example of Use**

sens:anal:auto 0

SENSe:ANALyze:AUTO?

**Function**

Queries if auto-analysis after test is set to ON.

**Syntax**

SENSe:ANALyze:AUTO?

**Parameters**

None

**Response Data**

Possible responses are:

0: Auto-analysis is OFF.
1: Auto-analysis is ON.

**Errors**

None

**Example of Use**

sens:anal:auto?
> 0<END>
4.5 TRACe Subsystem Commands

The TRACe subsystem provides access to the trace analysis and trace data.

TRACe:PARameters?

Function
Queries the trace parameters by text format.

Syntax
TRACe:PARameters?

Parameters
None

Response Data
Trace parameters as comma separated numeric values.
<wave>,<range>,<pulse>,<avg>,<reso>,<ior>,<bsc>,<enh><END>

Errors
(–400, "std_queryGen, Trace Not Ready")

Example of Use
trac:par?
> 1310, 16.415554, 50, 6144, 0.656621, 1.467700, –78.500000, 1<END>

TRACe:ANALyze

Function
Performs analysis on the trace.
To be used if analysis parameters are changed or auto analysis is set to OFF.
This command is an overlapped command.

Syntax
TRACe:ANALyze

Parameters
None
Response Data
None

Errors
(–200, "std_execGen, Test is Active")
(–200, "std_execGen, Trace Not Ready")

Example of Use
trac:anal

TRACe:ANALyze?
Function
Queries if analysis is done on the trace.

Syntax
TRACe:ANALyze?

Parameters
None

Response Data
Possible responses are:
0: Trace not analyzed.
1: Trace is analyzed.

Errors
(–400, "std_queryGen, Trace Not Ready")

Example of Use
trac:anal?
> 0<END>

TRACe:ANALyze:ORL
Function
Performs ORL calculations on the trace.
This command is an overlapped command.

Syntax
TRACe:ANALyze:ORL
Parameters
None

Response Data
None

Errors
(–200, "std_execGen, Test is Active")
(–200, "std_execGen, Trace Not Ready")
(–200, "std_execGen, Invalid Loss Mode")

Example of Use
trac:anal:orl

TRACe:MDLOss?
Function
Get trace loss values. The returned values depend on current loss mode.
For single loss modes only first value is valid. If loss mode is not calculable the returned value will be –99.99.

Syntax
TRACe:MDLOss?

Parameters
None

Response Data
Calculated loss values. Two comma separated numeric values.

Errors
(–200, "std_execGen, Test is Active")
(–400, "std_queryGen, Trace Not Ready")

Example of Use
trac:mdlo?
> –4.610,—99.99<END>
**Function**
Get trace end-to-end loss. If end-to-end loss is not calculable the returned value will be –99.99.

**Syntax**
```
TRACe:EELOss?
```

**Parameters**
None

**Response Data**
Calculated end-to-end loss value.

**Errors**

(-200, "std_execGen, Test is Active")
(-400, "std_queryGen, Trace Not Ready")

**Example of Use**
```
trac:eelo?
> -4.610<END>
```

**Function**
Get SOR trace object.
Refer to “Binary Data” in Section 2.6.3 “Data Formats” for further details.

**Syntax**
```
TRACe:LOAD:SOR?
```

**Parameters**
None

**Response Data**
SOR trace file as an array of bytes per SCPI binary transfer specifications.

**Errors**

(-200, "std_execGen, Test is Active")
4.5 TRACe Subsystem Commands

Example of Use

```
trac:load:sor?
> “BINARY ARRAY”
```

#524047 //SCPI data size message for binary data transfer followed by array of SOR file bytes.

**TRACe:LOAD:TEXT?**

**Function**

TRACe:LOAD:TEXT? command has three modes.

- **TRACe:LOAD:TEXT?**
  - Get SOR trace information in text format and full trace data.

- **TRACe:LOAD:TEXT? <start>**
  - **<start>:** Start position of the trace data. Distance unit is km.
  - Get the trace data from specified start point to the end of distance range.

- **TRACe:LOAD:TEXT? <start>,<end>**
  - **<start>:** Start position of the trace data. Distance unit is km.
  - **<end>:** End position of the trace data. Distance unit is km.
  - Get the trace data from specified start to end position.

Refer to “Binary Data” in Section 2.6.3 “Data Formats” for further details.

**Syntax**

```
TRACe:LOAD:TEXT?
TRACe:LOAD:TEXT?<wsp><start>
TRACe:LOAD:TEXT?<wsp><start>,<end>
```

**Parameters**

- **<start>**
  - Floating point format
  - Range: 0.000000 to Current distance range.

- **<end>**
  - Floating point format
  - Range: 0.000000 to Current distance range.

- **<start>** value must be less or equal to **<end>** value.
Response Data
SOR trace data as an array of bytes per SCPI binary transfer specifications.

Errors
(–108, "std_tooManyParameters, Parameter not Allowed")
(–200, "std_execGen, Test is Active")
(–224, "std_illegalParmValue, Invalid Parameter Value")
(–400, "std_queryGen, Trace Not Ready")

Example of Use
trac:load:text?
#6140479 //SCPI data size message for binary data transfer
WL   = 1310 nm //Wavelength
FBR  = SM //Fiber Type
DR   = 5 km //
PW   = 50 ns [ER] //Pulse Width and resolution type
AVG  = 6144 //Number of hardware averages
IOR  = 1.467700 //IOR value
BSC  = –78.50 //BSC value
DATE = 08/13/09 //Date of test
TIME = 10:19 PM //Time of test
MXDB = 64 dB //dB Range
RESO = 0.200 m //Resolution value
DX   = 0.20440100621721 m //Point spacing
PTS  = 25001 //Number of data points in the trace

//Start of trace data points
1000
9741
41291
41923
.....

9741
9741
0
//End of trace data points
4.5 TRACe Subsystem Commands

Events 1 //Number of events found by analysis

Dist 1.0505 km //Event distance
Type E //Event type
Loss >3.00 dB //Event loss value
Reflectance N/A //Event reflectance value
dB / km 59.420 dB //dB/km Loss value
Cumulative Loss 1.81 dB //Cumulative loss value

TRACe:LOAD:DATA?

Function
TRACe:LOAD:DATA? command has four mode.

TRACe:LOAD:DATA?
Get full trace data.

TRACe:LOAD:DATA? <start>
<start>: Start position of the trace data. Distance unit is km.
Get all the trace data form specified start point to the end of distance range.

TRACe:LOAD:DATA? <start>,<end>
<start>: Start position of the trace data. Distance unit is km.
<end>: End position of the trace data. Distance unit is km.
Get all the trace data form specified start to end position.

TRACe:LOAD:DATA? <start>,<end>,<space>
<start>: Start position of the trace data. Distance unit is km.
<end>: End position of the trace data. Distance unit is km.
<space>: Data point spacing in terms of resolution.
Get the trace data form specified start to end position at the interval specified by space.

Refer to “Binary Data” in Section 2.6.3 “Data Formats” for further details.

Syntax
TRACe:LOAD:DATA?<wsp><start>,<end>,<space>
Parameters

<start> Starting distance (km)
Floating point format
Range: 0.0 to (Current distance range – <space>*resolution)

<end> Ending distance (km)
Floating point format
Range: (<start>+<space>*resolution) to (Current distance range)

<space> data point spacing in terms of resolution
Numeric format
Range: 1 to (Max number of data points between <start> and <end> distance)
<start> Starting distance (km)
Numeric format
Range: 0.0 to (Current distance range – <space>*resolution)

Start value must be less or equal to end value.

Response Data
Requested data points as an array of bytes per SCPI binary transfer specifications.

Errors
(–225, "std_illegalState, Trace Not Ready")
(–224, "std_illegalArgumentException, Invalid Parameter Value")

Example of Use
trac:load:data? 0.0,10.0,1
> "BINARY ARRAY"

#510006 //SCPI data size message for binary data transfer followed by array of data point bytes. First four bytes – unsigned long (big endian) for number of data points to follow, Every next two bytes – unsigned short for each data point requested

TRACe:HEADer

Function
Sets Trace Header. If set to blank, parameter is no characters. But <Data Flag> and <Direction> cannot be blank (Always set).
If you want to set blank in <Comment>, add a comma at the last argument.

**Syntax**

```
TRACe:HEADer<Data Flag>,<Cable ID>,<Fiber ID>,
<Cable Code>,<Start location>,<Terminal Location>,
<Direction>,<Operator>,<Comment>
```

**Parameters**

- `<Data Flag>`: BC, RC, OT
- `<Cable ID>`: Up to 30 characters
- `<Fiber ID>`: Up to 30 characters
- `<Cable Code>`: Up to 30 characters
- `<Start location>`: Up to 30 characters
- `<Terminal Location>`: Up to 30 characters
- `<Direction>`: 0: A->B, 1: B->A
- `<Operator>`: Up to 30 characters
- `<Comment>`: Up to 30 characters

**Response Data**

None

**Errors**

- –108, "std_tooManyParameters, Parameter not Allowed"
- –109, "std_tooFewParameters, Missing Parameter"
- –224, "std_illegalParmValue, Invalid Parameter Value"

**Example of Use**

```
trac:head OT,1,1,ABC,Tokyo,Yokohama,1,Anritsu,TEST
trac:head BC,,,,,,0,,
```

**TRACe:HEADer?**

**Function**

Queries Trace Header.

**Syntax**

```
TRACe:HEADer?
```

**Parameters**

None
Response Data
Nine header.

Errors
(–108, "std_tooManyParameters, Parameter not Allowed")

Example of Use
trac:head?
> OT, 1,1,ABC, Tokyo,Yokohama,1,Anritsu,TEST<END>

TRACe:STORe:SOR
Function
Stores the SOR file in internal memory of the ACCESS Master.

Syntax
TRACe:STORe:SOR<wsp><filepath&name>

Parameters
<filepath&name> File path and file name to save the SOR file

Response Data
None

Errors
(–108, "std_tooManyParameters, Parameter not Allowed")
(–109, "std_tooFewParameters, Missing Parameter")
(–224, "std_illegalParmValue, Invalid Parameter Value")
(–400, "std_queryGen, Trace Not Ready")

Example of Use
trac:stor:sor test.sor
test.sor file is stored in root folder of internal memory.
trac:stor:sor TEST/test.sor
test.sor file is stored in TEST folder of internal memory.
4.6 DISPlay Subsystem Commands

The DISPlay subsystem provides access to the display mode and display zooming/scaling.

DISPlay:MODE

Function
Selects the reference cursor A or B.

Syntax
DISPlay:MODE<wsp><mode>

Parameters
<mode> Reference cursor
Numeric format
Range: 0|1
0: Cursor A
1: Cursor B

Response Data
None

Errors
(–224, "std_illegalValue, Invalid Parameter Value")

Example of Use
disp:mode 1

DISPlay:MODE?

Function
Queries which reference cursor is selected.

Syntax
DISPlay:MODE?

Parameters
None

Response Data
Reference cursor as numeric values.
Errors
None

Example of Use
    disp:mode?
    > 1

DISPlay:ZOOM:FULL
Function
Sets display zoom to full trace in current display mode.

Syntax
    DISPlay:ZOOM:FULL

Parameters
None

Response Data
None

Errors
None

Example of Use
    disp:zoom:full

DISPlay:ZOOM:HORIzontal
Function
Sets Horizontal display zoom to specified level.

Syntax
    DISPlay:ZOOM:HORIzontal<wsp><level>

Parameters
    <level> Zoom level
    Numeric format
    Range: 0 to (6 to 16) Depends on current distance range
    0: Full zoom in,
    (6 to 16): Full Zoom out
4.6 DISPlay Subsystem Commands

**Note:**
The maximum value for each distance is as follows.

<table>
<thead>
<tr>
<th>Distance</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.5 km</td>
<td>6</td>
</tr>
<tr>
<td>1 km</td>
<td>7</td>
</tr>
<tr>
<td>2.5 km</td>
<td>8</td>
</tr>
<tr>
<td>5 km</td>
<td>9</td>
</tr>
<tr>
<td>10 km</td>
<td>10</td>
</tr>
<tr>
<td>25 km</td>
<td>11</td>
</tr>
<tr>
<td>50 km</td>
<td>12</td>
</tr>
<tr>
<td>100 km</td>
<td>13</td>
</tr>
<tr>
<td>200 km</td>
<td>14</td>
</tr>
<tr>
<td>300 km</td>
<td>16</td>
</tr>
</tbody>
</table>

**Response Data**
None

**Errors**
(–224, "std_illegalValue, Invalid Parameter Value")

**Example of Use**
disp:zoom:hori 2

**DISPlay:ZOOM:HORIzontal?**

**Function**
Queries Horizontal display zoom level.

**Syntax**
DISPlay:ZOOM:HORIzontal?

**Parameters**
None

**Response Data**
Horizontal zoom level as numeric value.

**Errors**
None

**Example of Use**
disp:zoom:hori?
> 5<END>

**DISPlay:ZOOM:VERTical**

**Function**
Sets Vertical display zoom to specified level.
Chapter 4  OTDR Commands

Syntax
DISPlay:ZOOM:VERTical <wsp><level>

Parameters
<level> Zoom level
Numeric format
Range: 0 to 6
0: Full zoom in,
6: Full Zoom out

Response Data
None

Errors
(–224, "std_illegalValue, Invalid Parameter Value")

Example of Use
disp:zoom:vert 2

DISPlay:ZOOM:VERTical?

Function
Queries Vertical display zoom level.

Syntax
DISPlay:ZOOM:VERTical?

Parameters
None

Response Data
Vertical zoom level as numeric value.

Errors
None

Example of Use
disp:zoom:vert?
> 5<END>
4.6 DISPlay Subsystem Commands

DISPlay:SCALE:HORizontal

Function
Sets Horizontal scale range to specified value.
The distance specified with Horizontal scale is displayed on screen.

Syntax
DISPlay:SCALE:HORizontal<wsp><scale>

Parameters
<scale> Horizontal scale range value in km
Floating point format
Range: 0.0124 to (0.5022 to 300.0056)
0.0124: Full zoom in
0.5022 to 300.0056: Full Zoom out
Depends on current distance range (0.5 to 300) km

Note:
There are 17 predefined scale ranges (0.0124, 0.0186, 0.0310, 0.0558, 0.1054, 0.2542, 0.5022, 1.0044, 2.5048, 5.0034, 10.0006, 25.0046, 50.0030, 100.0060, 200.0058, 250.0026, 300.0056), the specified <scale> value will snap to the nearest highest predefined scale value.
The maximum value for each distance is as follows.

<table>
<thead>
<tr>
<th>Distance</th>
<th>Scale Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.5 km</td>
<td>0.5022</td>
</tr>
<tr>
<td>1 km</td>
<td>1.0054</td>
</tr>
<tr>
<td>2.5 km</td>
<td>2.5048</td>
</tr>
<tr>
<td>5 km</td>
<td>5.0034</td>
</tr>
<tr>
<td>10 km</td>
<td>10.0006</td>
</tr>
<tr>
<td>25 km</td>
<td>25.0046</td>
</tr>
<tr>
<td>50 km</td>
<td>50.0030</td>
</tr>
<tr>
<td>100 km</td>
<td>100.0060</td>
</tr>
<tr>
<td>200 km</td>
<td>200.0058</td>
</tr>
<tr>
<td>300 km</td>
<td>300.0056</td>
</tr>
</tbody>
</table>

Response Data
None

Errors
(–224, "std_illegalValue, Invalid Parameter Value")

Example of Use
disp:scal:hori 5.0
DISPlay:SCALE:HORIzontal?

**Function**
Queries Horizontal display scale range value in km.

**Syntax**
DISPlay:SCALE:HORIzontal?

**Parameters**
None

**Response Data**
Horizontal scale range value in km as numeric value.

**Errors**
None

**Example of Use**
```plaintext
disp:scal:hori?
> 5.0034<END>
```

DISPlay:SCALE:VERTical

**Function**
Set Vertical display scale range to specified value.
The value specified with Vertical display scale is displayed on screen.

**Syntax**
DISPlay:SCALE:VERTical<wsp><scale>

**Parameters**

- `<scale>` Vertical scale range value in dB
- Floating point format
- Range: 0.5 to 65
- 0.5: Full zoom in,
- 65: Full Zoom out

**Note:**
There are 7 predefined scale ranges (0.5, 1.0, 2.5, 5.0, 10.0, 25.0, 65.0), the specified `<scale>` value will snap to the nearest highest predefined scale value.
Response Data
None

Errors
(–224, "std_illegalValue, Invalid Parameter Value")

Example of Use
disp:scal:vert 0.5

DISPlay:SCALe:VERTical?
Function
Queries Vertical display scale range value in dB.

Syntax
DISPlay:SCALe:VERTical?

Parameters
None

Response Data
Vertical scale range in dB as numeric value.

Errors
None

Example of Use
disp:scal:vert?
> 10.0<END>