MT9083 Series
ACCESS Master
Remote Control
Operation manual

Fourth 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 MT9083A2/B2/C2 ACCESS Master Operation Manual. Please also refer to this document before using the equipment.**
- **Keep this manual with the equipment.**

ANRITSU CORPORATION

Document No.: M-W3636AE-4.0

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.

MT9083 Series ACCESS Master
Remote Control
Operation Manual

29 June 2012 (First Edition)
30 September 2016 (Fourth Edition)

About This Manual

This operation manual describes the SCPI (Standard Commands for Programmable Instruments) commands for the MT9083A2/B2/C2 ACCESS Master.

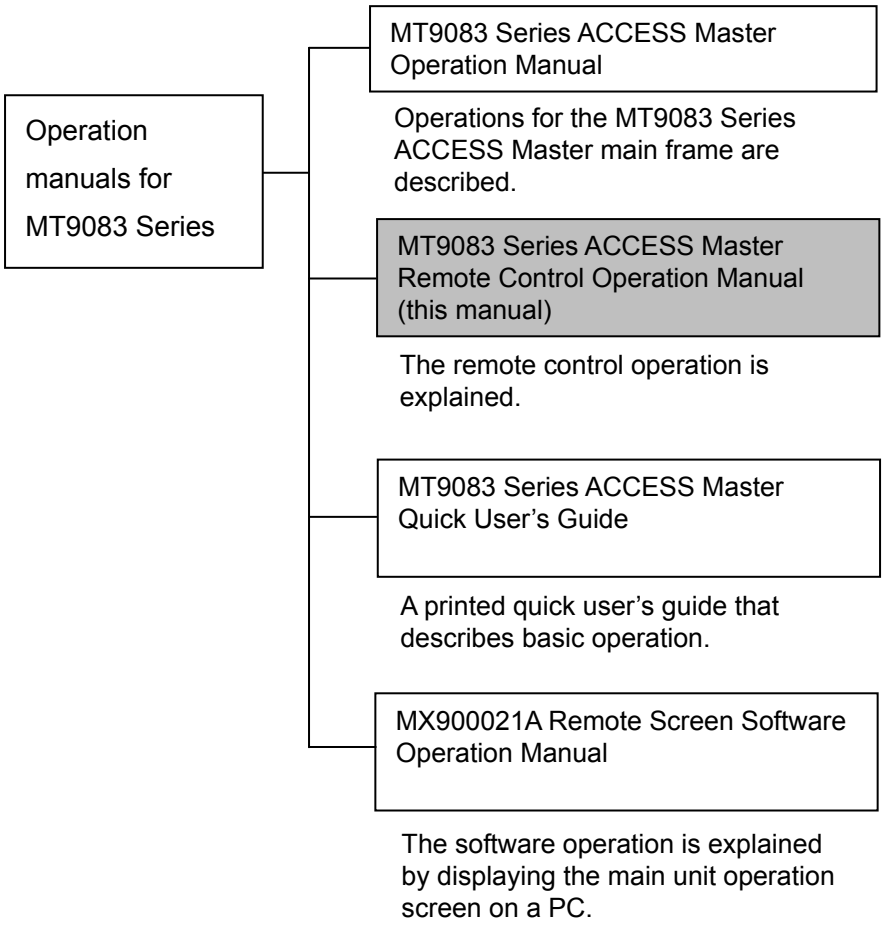

Table of Contents

Chapter 1 Overview

This chapter explains remote control of the MT9083A2/B2/C2 ACCESS Master.

1.1 About Remote Control

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

The MT9083A2/B2/C2 ACCESS Master (this instrument hereafter) uses the Ethernet interface for remote control.
Ethernet requires setting a unique IP address for this instrument so that it is recognized on the network.

The strings for controlling this instruments are called commands. Commands are comprised of character strings. For example, the following command sets the measurement wavelength to 1550 nm:

SOURce:WAVelength `1550`

A command for reading data from this instrument is called a query. A query command has the question symbol (?) appended to the string. For example, sending the following command queries the Distance Range set at the instrument.

```
SOURce:RANge?
```

The controller PC receives the following response from the instrument.

```
100
```

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

The front-panel displays and key operations are still enabled even when the instrument is being remotely controlled.

1.2 Applications

The main uses for remote control are listed below:

Automating Measurements

Instead of key-panel operations, measurement can be automated by controlling the instrument by executing programs.

Controlling Multiple Instruments

The characteristics of multiple DUTs 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 the preparations for using remote control.

2.1 Preparing Equipment

The following equipment is required to perform remote control.

- PC
- USB-Ethernet converter
- Ethernet cable
- Program development tools

PC

The PC must be able to run the program development tools.

USB-Ethernet converter

We recommend using the Anritsu USB-Ethernet converter, which is confirmed as working. Other commercial converters can be used but some may have compatibility problems.
Appendix A lists commercial USB-Ethernet converters that have been confirmed as compatible.

Ethernet cable

Depending on the connection type, use either straight through or crossover cables.

Program Development Tools

Prepare some tools for developing and running programs for performing remote control. Refer to the USB-Ethernet converter manual and SCPI Standard for the specifications required by the program development tools.

2.2 Connecting Equipment

2.2.1 Connecting USB-Ethernet converter

Connect the USB-Ethernet converter to the USB Down Port on the top edge of the ACCESS Master.

Figure 2.2.1-1 ACCESS Master and USB-Ethernet converter

If the converter is connected while the power is on, processing will be suspended and the Remote Setup screen displayed. Refer to section 2.3 “Setting Ethernet” to set the IP address, etc.
When the converter is connected while the power is off, press **Remote Setup (f1)** at the Top Menu displayed after power-on.

Settings such as the IP address, are saved at power-off. The previous settings are read at the next power-on.

2.2.2 Connecting External Equipment

Connect the external equipment with a LAN cable to the LAN port of the USB-Ethernet converter connected to the instrument. Use a crossover cable to connect one PC controller. When connecting multiple pieces of external equipment, use a network hub.

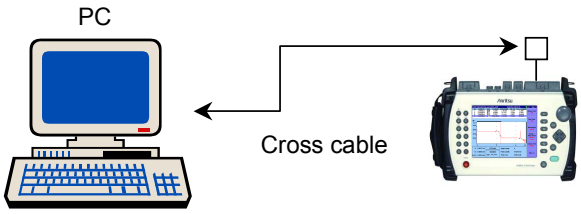

Figure 2.2.2-1 Connecting One External PC

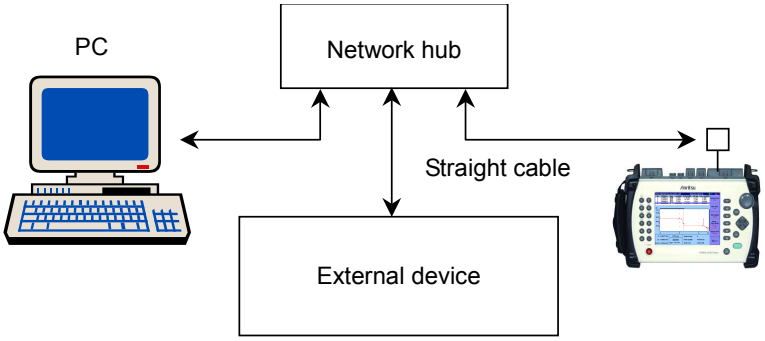

Figure 2.2.2-2 Connecting Multiple Pieces of External Equipment

Note:

Sometimes communications with the ACCESS Master are not established due to the external equipment communications status. For stable communications, connect directly using a crossover cable.

2.3 Setting Ethernet

Instrument settings such as IP address are set at the Remote Setup Screen.
The Remote screen is displayed by the following two methods.

- Press **Remote Setup (f1)** at the Top Menu.
- With the power on, connect the USB-Ethernet converter to the USB Down port.

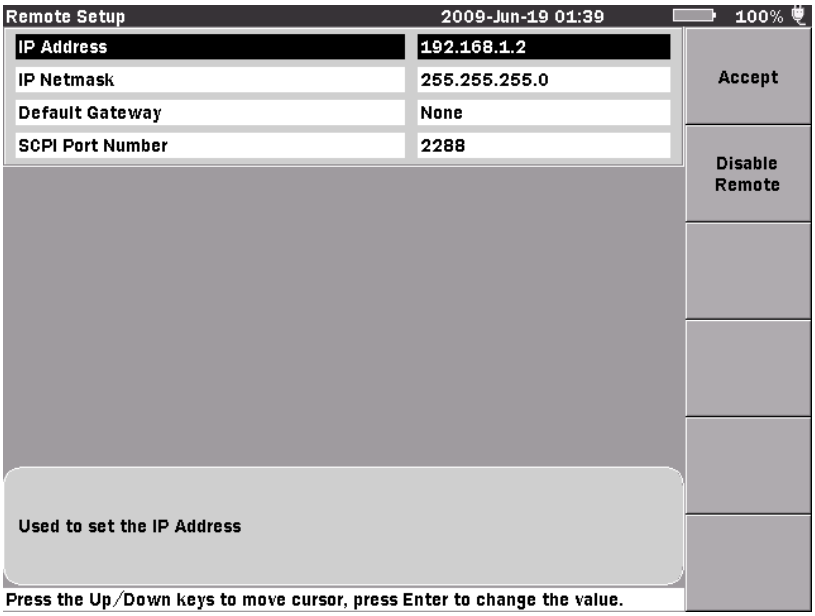

Figure 2.3-1 Remote Setup Screen

1. Set the IP Address. The input setting range is 0.0.0.1 to 255.255.255.254.
   Use the instrument's numeric keypad or the Left/Right/Up/Down Arrow keys to input the numeric values. The default setting is 192.168.1.2.
2. Set the IP Netmask.
   The input setting range is 0.0.0.0 to 255.255.255.255 and the default setting is 255.255.255.0.
3. Set the Default Gateway.
   The input setting range is 0.0.0.1 to 255.255.255.254 and the default is None.
4. Set the SCPI Port Number (TCP port).
   The input setting range is 1 to 65535 and the default is 2288.

Press **Accept (f1)** after finishing the settings to close the Remote Setup screen. When a connection is established, an icon indicating successful connection is displayed in the screen Title Bar.

Figure 2.3-2 Remote control icon

After setting the IP Address of the ACCESS Master, also set the address of a remote control PC.

2.4 Checking Connection

Check that the link between the PC and instrument has been established.

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.

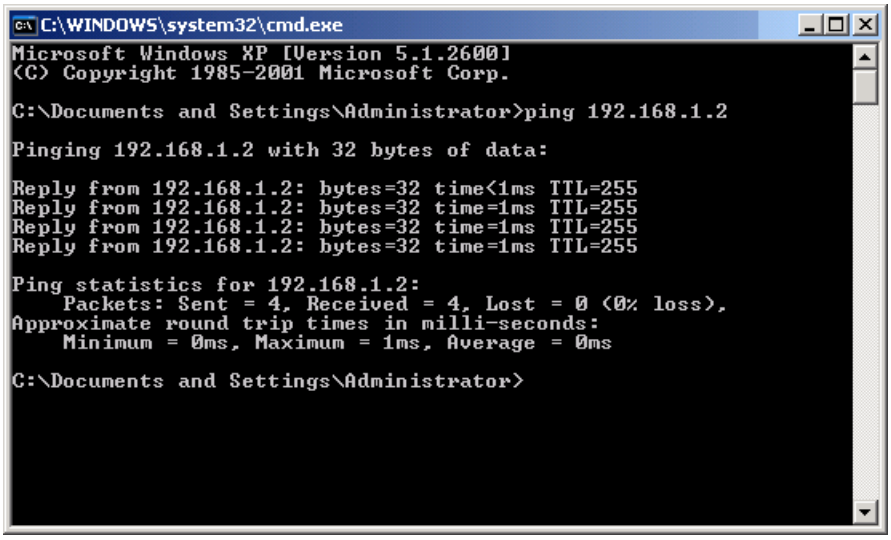

Figure 2.4-1 Executing Ping Command

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

2.5 Message Format

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

Messages are composed of the following types:

Program Messages:

Messages sent from PC to instrument
These are composed of commands to set the instrument and queries requesting sending of a response message.

Response Messages:

Messages sent from instrument to PC controller

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 must have appended data but may not include a header.
In this manual, white space is written as <wsp> in command syntax description.

The header is composed of alphanumeric characters and underbars while the head string is alphabetic characters. However, common commands defined by IEEE 488.2 have an asterisk (*) appended to the header string. Both upper and lower-case alphabetic characters are supported.

Command with only header:

```
*RST
*CLS
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

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

2.6 Checking Instrument Status

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

2.6.1 Register Structure

Figure 2.6.1-1 shows the structure of the registers indicating the instrument status.

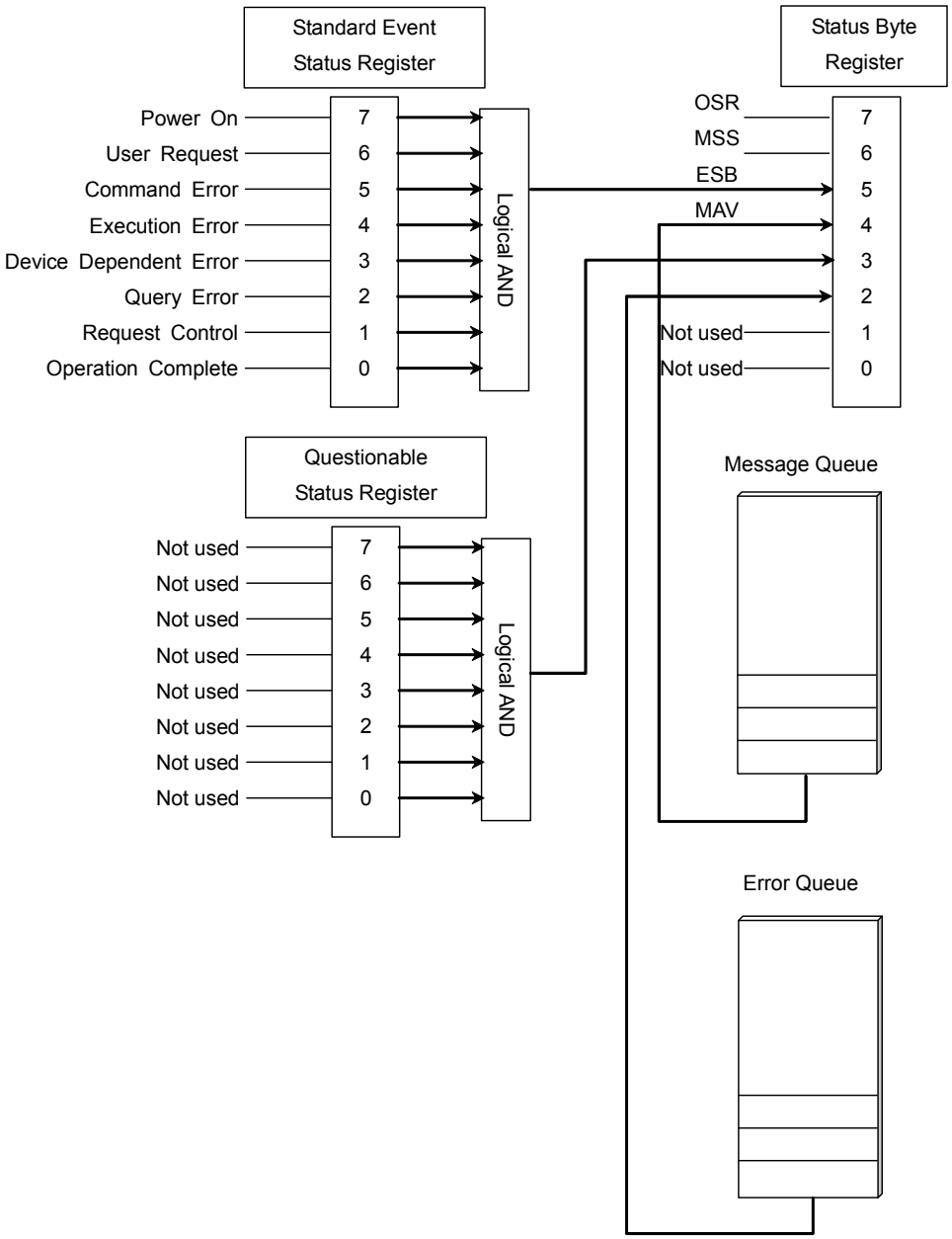

Figure 2.6.1-1 Register Structure

Each register uses 8-bit data. The register output values are the binary totals for each bit shown in Figure 2.6.1-1.

Table 2.6.1-1 Register Bit Binary Conversion Values

Bit	Binary value	Bit	Binary value
0	1	4	16
1	2	5	32
2	4	6	64
3	8	7	128

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 + 1 bit is output at bit 5.

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 instrument can be sent immediately.
Up to 12 messages from the PC can be spooled.

Error Queue

Up to 12 error messages from the instrument can be spooled.

2.6.2 Status Byte Register

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

Table 2.6.2-1 Meaning of Status Byte Register

Bit	Explanation
7	OSR (Operation Status Register) Displays the equipment'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 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 this instrument, 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.

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`.

2.6.3 Event Register

Standard Event Status Register

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

Table 2.6.3-1 Meaning of Standard Event Status Register

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

Bit 7 to bit 1 of the standard event register can be read by the `*ESR?` command. The standard event register returns to 0 when read.

The standard event register enable register can be set and read using the `*ESE` and `*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 `*OPC` common command.

The standard event register can be set to 0 using the `*CLS` command.

2.7 Controlling Message Sync

The following messages (12 types max.) can be received during measurement by this instrument 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 2.7-1 shows the message execution sequence when this message is sent to the instrument. 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.

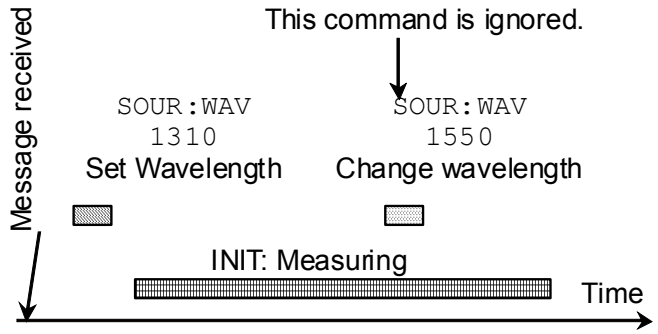

Figure 2.7-1 Message Execution Sequence

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 and *OPC command 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: INIT ; *WAI ; SOUR:WAV 1550; INIT

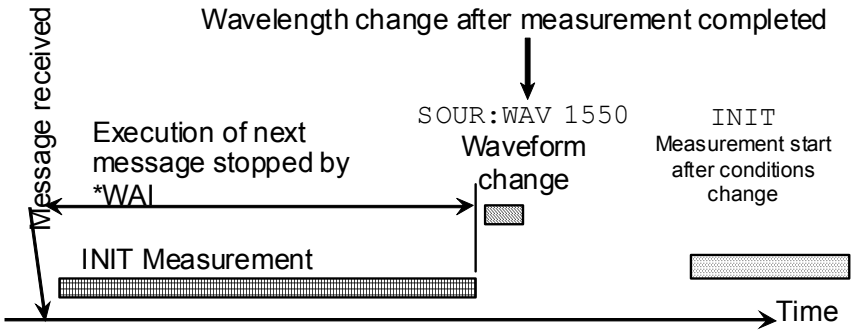

Figure 2.7-2 Sync Control using *WAI

Using *OPC?

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

Examples of Use:

INIT	Executes averaging measurement
*OPC?	Queries if operation is completed and waits until "1" returns.
> 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 OTDR
SOUR:WAV 1550	Changes wavelength
INIT	Starts measurement with new measurement conditions

Using *OPC and *ESR?

The *OPC common command sets the standard event status register bit to 0 and displays the OPC bit.

Examples of Use:

*CLS	Sets the OPC bit to 0.
*ESE 1	Sets standard event status enable bit 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 OTDR
SOUR:WAV 1550	Changes wavelength
INIT	Starts measurement with new measurement conditions

Querying Measurement End

The instrument program messages query the end of processing execution. These queries send the following messages after confirming the processing end.

Example of Use, 1: When averaging measurement is 60 s

`INIT`	Executes averaging measurement
`SENS:AVER:TIM?`	Queries elapsed averaging time
`> 20`	Measuring when response is 20
`SENS:AVER:TIM?`	Queries elapsed averaging time
`> 60`	Measurement ended when response is 60
`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 OTDR
`SOUR:WAV 1550`	Changes wavelength
`INIT`	Starts measurement with new measurement conditions

Example of Use, 2: When averaging measurement is 60 s

`INIT`	Starts averaging measurement.
`SENS:TRAC:READY?`	Queries if measurement is finished and waits until “0” returns.
`> 0`	Measuring when response is 1.
`SENS:TRAC:READY?`	Queries elapsed averaging time
`> 1`	Waveform data is ready when response is 1.
`TRAC:LOAD:SOR?`	Gets trace object from OTDR
`SOUR:WAV 1550`	Changes wavelength
`INIT`	Starts measurement with new measurement conditions

2.8 Switching SM Unit and MM unit

Two OTDR units for single mode fiber (measurement wavelength: 1310/1550 nm) and multimode fiber (measurement wavelength: 850/1300 nm) can be installed in this instrument according to the number of the option unit (e.g. MT9083B2 Option 063).

Switch between the SM unit and MM units as described below.

Example of Use 1: with MT9083B2-063 (MM 850/1300 nm/SM 1310/1550 nm)

`TOPMenu:UNIT:CATalog:FULL?`	Queries installed unit
`>MM, 1, SM, 2`	At MM, 1, SM, 2 response: Multimode unit installed as unit 1 and single mode unit installed as unit 2
`TOPMenu:UNIT:NSELect?`	Queries current measurement unit
>1	At 1 response: Multimode unit selected
`TOPM:UNIT:NSEL 2`	Switches to single mode unit

Example of Use 2: with MT9083B2-063 (MM 850/1300 nm/SM 1310/1550 nm)

`TOPMenu:UNIT:CATalog?`	Queries installed unit
`>MM, SM`	At MM, SM response: Multimode unit and single mode units installed
`TOPMenu:UNIT?`	Queries current measurement unit
`>MM`	At MM response: Multimode unit selected
`TOPM:UNIT:SEL SM`	Switches to single mode unit

2.9 Moving to Another Measurement Mode from Top Menu

In addition to OTDR measurement, this instrument 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 MT9083A2/B2/C2 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 MT9083C2-053

`INSTrument:CATalog:FULL?`	Queries measurement mode supporting remote
`>TOP_MENU, 1, OTDR_STD, 2`	At TOP_MENU, 1, OTDR_STD, 2: Supports selection of Mode 1 at Top Menu and Mode 2 at OTDR (Standard)
`INSTrument:NSELect?`	Queries current mode
`>1`	At 1 response: Top Menu
`INST:NSEL 2`	Switches to OTDR (Standard) Note: Measurement mode inactive
`INST:STAT 1`	Enables OTDR (Standard) mode
`SOUR:WAV 1310`	Sets wavelength to 1310 nm

Example of Use 2: To measure with MM unit after measuring with SM unit using MT9083B2-063 (MM 850/1300 nm/SM 1310/1550 nm)

INSTrument:NSELect?	Queries current measurement mode
>2	At 2 response: OTDR (Standard) mode
INST:NSEL 1	Moves to Top Menu
TOPMenu:UNIT:CATalog:FULL?	Queries installed OTDR
>MM, 1, SM, 2	Multimode unit installed a unit 1; single mode unit installed as unit 2
TOPM:UNIT:NSEL 1	Switches to multimode unit

Chapter 3 Platform SCPI Commands

This chapter details the SCPI commands for the MT9083A2/B2/C2 platform.

3.1 Star (IEEE 488.2) Subsystem Commands

3.1.1 *CLS Clear Status Command

Syntax:	*CLS
Description:	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 emptied. Neither the Standard Event Status Enable register, nor the Service Request Enable register are affected by this command. The command's effect on individual logical instrument depends on the instrument implementation and is described in the instrument's documentation.
Parameters:	None
Response:	None
Example:	*CLS

3.1.2 *ESE Standard Event Status Enable Command

Syntax:	*ESE<wsp><value>
Description:	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.
Parameters:	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
Response:	None
Example:	`*ESE 21`

3.1.3 *ESE? Standard Event Status Enable Query

Syntax:	*ESE?
Description:	The standard Event Status Enable query *ESE? returns the contents of the Standard Event Status Enable register (see *ESE for information on this register).
Parameters:	None
Response:	The bit value for the register (a **short** value).
Example:	`*ESE? –> 21<END>`

3.1.4 *ESR? Standard Event Status Register Query

Syntax:	*ESR?
Description:	The standard Event Status Register query *ESR? returns the contents of the Standard Event Status register. The register is cleared after being read.
Parameters:	None
Response:	The bit value for the register (a **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:	`*ESR? –> 22<END>`

3.1.5 *IDN? Identification Query

Syntax:	*IDN?
Description:	The Identification query *IDN? gets the instrument identification over the interface.
Parameters:	None
Response:	The identification string terminated by <END>
Example:	`*IDN? –> ANRITSU, MT9083C2-053, 6200123456<END>`

3.1.6 *OPC Operation Complete Command

Syntax:	*OPC
Description:	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, at the same time setting the OPC bit of the SESR TRUE. The following events force the device into OCIS without setting the No Operation Pending flag TRUE and without setting the OPC bit of the SESR: . Power-on . *CLS . *RST
Parameters:	None
Response:	None
Example:	`*OPC`

3.1.7 *OPC? Operation Complete Query

Syntax:	*OPC?
Description:	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
Parameters:	None
Response:	1<END>
Example:	*OPC? –> 1<END>

3.1.8 *RST Reset Command

Syntax:	*RST
Description:	The ReSeT command *RST sets the instrument to Top Menu. Pending *OPC actions are cancelled. The instrument 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)
Parameters:	None
Response:	None
Example:	*RST

3.1.9 *SRE Service Request Enable Command

Syntax:	*SRE<wsp><value>
Description:	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.
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:	None
Example:	`*SRE 64`

3.1.10 *SRE? Service Request Enable Query

Syntax:	*SRE?
Description:	The Service Request Enable query *SRE? returns the contents of the Service Request Enable register (see *SRE for information on this register).
Parameters:	None
Response:	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
Example:	`*SRE? –> 21<END>`

3.1.11 *STB? Read Status Byte Query

Syntax:	*STB?
Description:	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.
Parameters:	None
Response:	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
Example:	`*STB? –> 78<END>`

3.1.12 *TST? Self-Test Query

Syntax:	*TST?
Description:	This query is not used with the MT9083A2/B2/C2 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.
Parameters:	None
Response:	The sum of the results for the individual tests (a **32-bit signed integer** value)
Example:	`*TST? –> 0<END>`

3.1.13 *WAI Wait-to-Continue Command

Syntax:	*WAI
Description:	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.
Parameters:	None
Response:	None
Example:	`*WAI`

3.2 System Subsystem Commands

3.2.1 SYSTem:ERRor?

Syntax:	SYSTem:ERRor?
Description:	Returns the contents of the SCPI error queue. Removes the returned entry from the queue.
Parameters:	None
Response:	The number of the latest error, sorted by the error commands sending order, and its meaning.
Example:	SYST:ERR? -> -100, "std_command, Command Parse Error"<END>

3.2.2 SYSTem:VERSion?

Syntax:	SYSTem:VERSion?
Description:	Returns the SCPI revision to which the system complies.
Parameters:	None
Response:	The revision year and number string.
Example:	SYST:VERS? -> 1990.0<END>

3.2.3 SYSTem:LOCK?

Syntax:	SYSTem:LOCK?
Description:	Checks if the user interface of the remote unit is locked.
Parameters:	None
Response:	Possible responses are: 0 = User interface of the remote unit is not locked. 1 = User interface of the remote unit is locked.
Example:	SYST:LOCK? -> 0<END>

3.2.4 SYSTem:LOCK

Syntax:	SYSTem:LOCK<wsp><value>
Description:	Locks the user interface of the remote unit.
Parameters:	<value> Boolean format Range: 0|1 0 = Unlock the user interface of the remote unit. 1 = Lock the user interface of the remote unit.
Response:	None
Example:	SYST:LOCK 1<END>

3.2.5 SYSTem:LIGHt?

Syntax:	SYSTem:LIGHt?
Description:	Checks if the backlight is turned ON or OFF.
Parameters:	None
Response:	Possible responses are: 0 = The backlight is OFF. 1 = The backlight is ON.
Example:	`SYST:LIGH? -> 0<END>`

3.2.6 SYSTem:LIGHt

Syntax:	SYSTem:LIGHt<wsp><value>
Description:	Sets the backlight on the system ON or OFF.
Parameters:	<value> Boolean format Range: 0|1 0 = Turn the backlight OFF on the unit. 1 = Turn the backlight ON on the unit.
Response:	None
Example:	`SYST:LIGH 1<END>`

3.3 Status Subsystem Commands

All commands in this section are prepared for future functions. All the registers regarding these commands are set to 0 now.

3.3.1 STATus:OPERation[:EVENt]?

Syntax:	STATus:OPERation[:EVENt]?
Description:	Queries the operation event register.
Parameters:	None
Response:	The bit value for the operation event register as a **short** value (0 .. +32767)
Example:	`STAT:OPER? –> 0<END>`

3.3.2 STATus:OPERation:CONDition?

Syntax:	STATus:OPERation:CONDition?
Description:	Queries the operation condition register.
Parameters:	None
Response:	The bit value for the operation condition register as a **short** value (0 .. +32767)
Example:	`STAT:OPER:COND? –> 16<END>`

3.3.3 STATus:OPERation:BIT#:CONDition?

Syntax:	STATus:OPERation:BIT<n>:CONDition?
Description:	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 :STATus:OPERation status register.
Parameters:	None
Response:	The bit value for the operation condition register as a **short** value (0 .. 1)
Example:	`STAT:OPER:BIT8:COND? –> 1<END>`

3.3.4 STATus:OPERation:BIT#:ENABle

Syntax:	STATus:OPERation:BIT<n>:ENABle<wsp><value>
Description:	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
Parameters:	The bit value for the operation enable mask as a **short** value (0 .. 1)
Response:	None
Example:	`STAT:OPER:BIT11:ENAB 1`

3.3.5 STATus:OPERation:BIT#:ENABle?

Syntax:	STATus:OPERation:BIT<n>:ENABle?
Description:	Returns 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
Parameters:	None
Response:	The bit value for the operation enable mask as a **short** value (0 .. 1)
Example:	`STAT:OPER:BIT9:ENAB? –> 0<END>`

3.3.6 STATus:OPERation:BIT#[:EVENt]?

Syntax:	STATus:OPERation:BIT<n>[:EVENt]?
Description:	Queries the operation event register specified bit. The value of <n> is restricted from 8 to 12 and represents bits 8 through 12
Parameters:	None
Response:	The bit value for the operation event register as a **short** value (0 .. 1)
Example:	`STAT:OPER:BIT10:EVEN? –> 0<END>`

3.3.7 STATus:OPERation:ENABle

Syntax:	STATus:OPERation:ENABle<wsp><value>
Description:	Sets the operation enable mask for the event register.
Parameters:	The bit value for the operation enable mask as a **short** value (0 .. +32767)
Response:	None
Example:	`STAT:OPER:ENAB 128`

3.3.8 STATus:OPERation:ENABle?

Syntax:	STATus:OPERation:ENABle?
Description:	Returns the operation enable mask for the event register.
Parameters:	None
Response:	The bit value for the operation enable mask as a **short** value (0 .. +32767)
Example:	`STAT:OPER:ENAB? –> 128<END>`

3.3.9 STATus:OPERation:INSTrument:CONDition?

Syntax:	STATus:OPERation:INSTrument:CONDition?
Description:	Queries the instrument operation condition register.
Parameters:	None
Response:	The bit value for the operation condition register as a **short** value (0 .. +32767)
Example:	`STAT:OPER:INST:COND? –> 16<END>`

3.3.10 STATus:OPERation:INSTrument:ENABle

Syntax:	STATus:OPERation:INSTrument:ENABle<wsp><value>
Description:	Sets the instrument operation enable mask for the event register.
Parameters:	The bit value for the operation enable mask as a **short** value (0 .. +32767)
Response:	None
Example:	`STAT:OPER:INST:ENAB 128`

3.3.11 STATus:OPERation:INSTrument:ENABle?

Syntax:	STATus:OPERation:INSTrument:ENABle?
Description:	Returns the instrument operation enable mask for the event register.
Parameters:	None
Response:	The bit value for the operation enable mask as a **short** value (0 .. +32767)
Example:	`STAT:OPER:INST:ENAB? –> 128<END>`

3.3.12 STATus:OPERation:INSTrument[:EVENt]?

Syntax:	STATus:OPERation:INSTrument[:EVENt]?
Description:	Queries the instrument operation event register.
Parameters:	None
Response:	The bit value for the operation event register as a **short** value (0 .. +32767)
Example:	`STAT:OPER:INST? –> 0<END>`

3.3.13 STATus:OPERation:INSTrument:ISUMmary#:CONDition?

Syntax:	STATus:OPERation:INSTrument:ISUMmary<n>:CONDition?
Description:	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.
Parameters:	None
Response:	The bit value for the operation condition register as a **short** value (0 .. +32767)
Example:	`STAT:OPER:INST:ISUM4:COND? –> 16<END>`

3.3.14 STATus:OPERation:INSTrument:ISUMmary#:ENABle

Syntax:	STATus:OPERation:INSTrument:ISUMmary<n>:ENABle<wsp><value>
Description:	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.
Parameters:	The bit value for the operation enable mask as a **short** value (0 .. +32767)
Response:	None
Example:	`STAT:OPER:INST:ISUM1:ENAB 128`

3.3.15 STATus:OPERation:INSTrument:ISUMmary#:ENABle?

Syntax:	STATus:OPERation:INSTrument:ISUMmary<n>ENABle?
Description:	Returns 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.
Parameters:	None
Response:	The bit value for the operation enable mask as a **short** value (0 .. +32767)
Example:	`STAT:OPER:INST:ISUM4:ENAB? –> 128<END>`

3.3.16 STATus:OPERation:INSTrument:ISUMmary#[:EVENt]?

Syntax:	STATus:OPERation:INSTrument:ISUMmary<n>[:EVENt]?
Description:	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.
Parameters:	None
Response:	The bit value for the operation event register as a **short** value (0 .. +32767)
Example:	`STAT:OPER:INST:ISUM3? –> 0<END>`

3.3.17 STATus:QUEStionable[:EVENt]?

Syntax:	STATus:QUEStionable[:EVENt]?
Description:	Queries the questionable event register
Parameters:	None
Response:	The bit value for the questionable event register as a **short** value (0 .. +32767)
Example:	`STAT:QUES? –> 0<END>`

3.3.18 STATus:QUEStionable:CONDition?

Syntax:	STATus:QUEStionable:CONDition?
Description:	Queries the questionable condition register
Parameters:	None
Response:	The bit value for the questionable condition register as a **short** value (0 .. +32767)
Example:	`STAT:QUES:COND? –> 8<END>`

3.3.19 STATus:QUEStionable:BIT#:CONDition?

Syntax:	STATus:QUEStionable:BIT<n>:CONDition?
Description:	Queries the questionable condition register specified bit. The value of <n> is restricted from 9 to 12 and represents bits 9 through 12
Parameters:	None
Response:	The bit value for the questionable condition register as a **short** value (0 .. 1)
Example:	`STAT:QUES:BIT9:COND? –> 0<END>`

3.3.20 STATus:QUEStionable:BIT#:ENABle

Syntax:	STATus:QUEStionable:BIT<n>:ENABle<wsp><value>
Description:	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
Parameters:	The bit value for the questionable enable mask as a **short** value (0 .. 1)
Response:	None
Example:	`STAT:QUES:BIT11:ENAB 1`

3.3.21 STATus:QUEStionable:BIT#:ENABle?

Syntax:	STATus:QUEStionable:BIT<n>:ENABle?
Description:	Returns 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
Parameters:	None
Response:	The bit value for the questionable enable mask as a **short** value (0 .. 1)
Example:	`STAT:QUES:BIT10:ENAB? –> 1<END>`

3.3.22 STATus:QUEStionable:BIT#[:EVENt]?

Syntax:	STATus:QUEStionable:BIT<n>[:EVENt]?
Description:	Queries the questionable event register specified bit. The value of <n> is restricted from 9 to 12 and represents bits 9 through 12
Parameters:	None
Response:	The bit value for the questionable event register as a **short** value (0 .. 1)
Example:	`STAT:QUES:BIT9:EVEN? –> 0<END>`

3.3.23 STATus:QUEStionable:ENABle

Syntax:	STATus:QUEStionable:ENABle<wsp><value>
Description:	Sets the questionable enable mask for the event register
Parameters:	The bit value for the questionable enable mask as a **short** value (0 .. +32767)
Response:	None
Example:	`STAT:QUES:ENAB 128`

3.3.24 STATus:QUEStionable:ENABle?

Syntax:	STATus:QUEStionable:ENABle?
Description:	Returns the questionable enable mask for the event register.
Parameters:	None
Response:	The bit value for the questionable enable mask as a **short** value (0 .. +32767)
Example:	`STAT:QUES:ENAB? –> 128<END>`

3.3.25 STATus:QUEStionable:INSTrument:CONDition?

Syntax:	STATus:QUEStionable:INSTrument:CONDition?
Description:	Queries the questionable instrument condition register.
Parameters:	None
Response:	The bit value for the questionable condition register as a **short** value (0 .. +32767)
Example:	`STAT:QUES:INST:COND? –> 8<END>`

3.3.26 STATus:QUEStionable:INSTrument:ENABle

Syntax:	STATus:QUEStionable:INSTrument:ENABle<wsp><value>
Description:	Sets the questionable instrument enable mask for the event register
Parameters:	The bit value for the questionable enable mask as a **short** value (0 .. +32767)
Response:	None
Example:	`STAT:QUES:INST:ENAB 128`

3.3.27 STATus:QUEStionable:INSTrument:ENABle?

Syntax:	STATus:QUEStionable:INSTrument:ENABle?
Description:	Returns the questionable instrument enable mask for the event register.
Parameters:	None
Response:	The bit value for the questionable enable mask as a **short** value (0 .. +32767)
Example:	`STAT:QUES:INST:ENAB? –> 128<END>`

3.3.28 STATus:QUEStionable:INSTrument[:EVENt]?

Syntax:	STATus:QUEStionable:INSTrument[:EVENt]?
Description:	Queries the questionable instrument event register
Parameters:	None
Response:	The bit value for the questionable event register as a **short** value (0 .. +32767)
Example:	`STAT:QUES:INST:EVEN? –> 0<END>`

3.3.29 STATus:QUEStionable:INSTrument:ISUMmary#:CONDition?

Syntax:	STATus:QUEStionable:INSTrument:ISUMmary<n>:CONDition?
Description:	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.
Parameters:	None
Response:	The bit value for the questionable condition register as a **short** value (0 .. +32767)
Example:	`STAT:QUES:INST:ISUM2:COND? –> 0<END>`

3.3.30 STATus:QUEStionable:INSTrument:ISUMmary#:ENABle

Syntax:	STATus:QUEStionable:INSTrument:ISUMmary<n>:ENABle<wsp><value>
Description:	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.
Parameters:	The bit value for the questionable enable mask as a **short** value (0 .. +32767)
Response:	None
Example:	`STAT:QUES:INST:ISUM3:ENAB 128`

3.3.31 STATus:QUEStionable:INSTrument:ISUMmary#:ENABle?

Syntax:	STATus:QUEStionable:INSTrument:ISUMmary<n>:ENABle?
Description:	Returns 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.
Parameters:	None
Response:	The bit value for the questionable enable mask as a **short** value (0 .. +32767)
Example:	`STAT:QUES:INST:ISUM3:ENAB? –> 136<END>`

3.3.32 STATus:QUEStionable:INSTrument:ISUMmary#[:EVENt]?

Syntax:	STATus:QUEStionable:INSTrument:ISUMmary[:EVENt]?
Description:	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.
Parameters:	None
Response:	The bit value for the questionable event register as a **short** value (0 .. +32767)
Example:	`STAT:QUES:INST:ISUM4:EVEN? –> 0<END>`

3.3.33 STATus:PRESet

Syntax:	STATus:PRESet
Description:	Resets both the operation enable mask and questionable enable mask to 0
Parameters:	None
Response:	None
Example:	`STAT:PRES`

3.4 Instrument Subsystem Commands

3.4.1 INSTrument:CATalog?

Syntax:	INSTrument:CATalog?
Description:	Returns the list of SCPI controllable instruments on MT9083A2/B2/C2 that are identified as SCPI controllable instruments
Parameters:	None
Response:	comma-separated list of string identifiers of all SCPI controllable logical instruments
Example:	`INST:CAT? –> TOP_MENU, OTDR_STD<END>`

3.4.2 INSTrument:CATalog:FULL?

Syntax:	INSTrument:CATalog:FULL?
Description:	Returns the list of SCPI controllable instruments on MT9083A2/B2/C2 that are identified as SCPI controllable
Parameters:	None
Response:	Returns a list of string - number pairs. The string contains the name identifier of the logical instrument. The immediately following NR1-formatted number is its associated logical instrument number. All response data elements are comma separated.
Example:	`INSTrument:CATalog:FULL? –> TOP_MENU, 1, OTDR_STD, 2<END>`

3.4.3 INSTrument:NSELect

Syntax:	INSTrument:NSELect<wsp><num_id>
Description:	Sets the specified logical instrument to be currently selected instrument.
Parameters:	The numeric value identifier assigned by the Instrument subsystem for the instrument to be selected as a **short** value
Response:	None
Example:	`INST:NSEL 2`

3.4.4 INSTrument:NSELect?

Syntax:	INSTrument:NSELect?
Description:	Returns the numeric value identifier of the currently selected logical instrument
Parameters:	None
Response:	Returns a numeric value of the currently selected logical instrument
Example:	`INST:NSEL? –> 2<END>`

3.4.5 INSTrument[:SELect]

Syntax:	INSTrument:SELect<wsp><string_id>
Description:	Sets the specified logical instrument to be currently selected instrument
Parameters:	The string instrument identifier assigned by the Instrument subsystem for the instrument to be selected as a **string** value
Response:	None
Example:	`INST:SEL OTDR_STD`

3.4.6 INSTrument[:SELect]?

Syntax:	INSTrument:SELect?
Description:	Returns the string value identifier of the currently selected logical instrument
Parameters:	None
Response:	Returns the string value identifier of the currently selected logical instrument
Example:	`INST:SEL? –> OTDR_STD<END>`

3.4.7 INSTrument:STATe

Syntax:	INSTrument:STATe<wsp><boolean>
Description:	Turns the currently selected logical instrument state ON or OFF
Parameters:	The Boolean data (ON or 1) to turn instrument on and (OFF or 0) to turn instrument off
Response:	None
Example:	`INST:STAT ON`

3.4.8 INSTrument:STATe?

Syntax:	INSTrument:STATe?
Description:	Returns the state of the currently selected logical instrument
Parameters:	None
Response:	Returns a boolean value of the state of the currently selected logical instrument (1 if the instrument is active, 0 if the instrument is inactive)
Example:	`INST:STAT? –> 1<END>`

3.5 TOPMenu Subsystem Commands

3.5.1 TOPMenu:UNIT:CATalog?

Syntax:	TOPMenu:UNIT:CATalog?
Description:	Returns the list of Units installed on MT9083A2/B2/C2 that can be selected prior to starting OTDR instrument
Parameters:	None
Response:	comma-separated list of string identifiers of all installed Units
Example:	`TOPM:UNIT:CAT? –> MM, SM<END>`

3.5.2 TOPMenu:UNIT:CATalog:FULL?

Syntax:	TOPMenu:UNIT:CATalog:FULL?
Description:	Returns the list of Units installed on MT9083A2/B2/C2 that can be selected prior to starting OTDR instrument
Parameters:	None
Response:	Returns a list of string - number pairs. The string contains the name identifier of the installed Units. The immediately following NR1-formatted number is its associated logical unit number. All response data elements are comma separated.
Example:	`TOPM:UNIT:CAT:FULL? –> MM, 1, SM, 2<END>`

3.5.3 TOPMenu:UNIT[:SELect]

Syntax:	TOPMenu:UNIT:SELect<wsp><string_id>
Description:	Sets the specified Unit to be currently selected Unit
Parameters:	The string instrument identifier assigned by the TOPMenu subsystem for the Unit to be selected as a **string** value
Response:	None
Example:	`TOPM:UNIT:SEL SM`

3.5.4 TOPMenu:UNIT[:SELect]?

Syntax:	TOPMenu:UNIT:SELect?
Description:	Returns the string value identifier of the currently selected Unit
Parameters:	None
Response:	Returns the string value identifier of the currently selected Unit
Example:	`TOPM:UNIT:SEL? –> MM<END>`

3.5.5 TOPMenu:UNIT:NSELect

Syntax:	TOPMenu:NSELect<wsp><num_id>
Description:	Sets the specified Unit to be currently selected to run with OTDR instrument.
Parameters:	The numeric value identifier assigned by the TOPMenu subsystem for the Unit to be selected as a **short** value
Response:	None
Example:	`TOPM:UNIT:NSEL 2`

3.5.6 TOPMenu:UNIT:NSELect?

Syntax:	TOPMenu:UNIT:NSELect?
Description:	Returns the numeric value identifier of the currently selected Unit
Parameters:	None
Response:	Returns a numeric value of the currently selected Unit
Example:	`TOPM:UNIT:NSEL? –> 1<END>`

Chapter 4 OTDR Commands

This chapter details the SCPI commands for the MT9083A2/B2/C2 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

The Command Summary section provides a list of the SCPI commands for the MT9083A2/B2/C2 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.

Root level commands

ABORt - Aborts active test, data is lost.
STOP – Stops the test, keeps the trace on screen.
INITiate - Start test with manual test settings.
INITiate:AUTo - Start auto-test.
INITiate:RTIMe – Start real time test with manual settings.
INITiate? - Queries if test is active.

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 mode is available for selected pulse width.
SOURce:PULSe:ENHanced - Sets enhanced mode for current selected pulse width.
SOURce:PULSe:ENHanced? - Queries enhanced mode for current selected pulse width.
SOURce:AVERages:TIMe - Sets averaging time for the next test.
SOURce:AVERages:TIMe? - Queries averaging time for the next test.

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:HOFFset - Sets horizontal offset value.
SENSe:HOFFset? - 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 trace analysis parameters values.
SENSe:ANALyze:PARameters? - Queries trace analysis parameters values.
SENSe:ANALyze:AUTO - Sets trace auto analysis option ON/OFF.
SENSe:ANALyze:AUTO? - Queries if the trace auto analysis option is on.

Trace Subsystem Commands

TRACe:PARameters? - Queries trace parameters summary in text format.
TRACe:ANALyze - Sets trace to be analyzed.
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:LOADSTORe:SOR - Stores the SOR file in internal memory of MT9083.

Display Subsystem Commands

DISPlay:MODE – Sets Current display mode (From A, From B, From Origin).
DISPlay:MODE? – Queries current display mode.
DISPlay:ZOOM:FULL – Sets zoom to view full trace in current display mode.
DISPlay:ZOOM:HORIzontal – Set the display zoom level in current display mode.
DISPlay:ZOOM:HORIzontal? – Queries the display zoom level in current display mode.
DISPlay:ZOOM:VERTical– Set the display zoom level in current display mode.
DISPlay:ZOOM:VERTical? – Queries the display zoom level in current display mode.
DISPlay:SCALe:HORIzontal – Set the display scale range in current display mode.
DISPlay:SCALe:HORIzontal? – Queries the display scale range in current display mode.
DISPlay:SCALe:VERTical– Set the display scale range in current display mode.
DISPlay:SCALe:VERTical? – Queries the display scale range in current display mode.

4.2 Root Level Commands

These commands start/stop OTDR test sequence

4.2.1 ABORt Command

Syntax:	ABORt
Parameters:	None
Description:	Aborts the active test. The trace data will be lost.
Example:	abor
Response:	None
Errors:	(–200, "std_execGen, Test is Inactive")

4.2.2 STOP Command

Syntax:	STOP
Parameters:	None
Description:	Stops the active test. The trace data will be preserved.
Example:	stop
Response:	None
Error:	(–201, "std_execGen, Test is Inactive") (–200, "std_execGen, Instrument is Busy")

4.2.3 INITiate Command

Syntax:	INITiate
Parameters:	None
Description:	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.
Example:	init
Response:	None
Error:	(–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")

4.2.4 INITiate:AUTo Command

Syntax:	INITiate:AUTo
Parameters:	None
Description:	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.
Example:	init:aut
Response:	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")

4.2.5 INITiate:RTIMe Command

Syntax:	INITiate:RTIMe
Parameters:	None
Description:	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.
Example:	init:rtim
Response:	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")

4.2.6 INITiate? Command

Syntax:	INITiate?
Parameters:	None
Description:	Queries if test is initiated.
Example:	init? --- 0<END>
Response:	Possible response range: 0 | 1 0 = Test is NOT active. 1 = Test is currently active.
Errors:	None

4.3 SOURce Subsystem Commands

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

4.3.1 SOURce:WAVelength:AVAilable? Command

Syntax:	SOURce:WAVelength:AVAilable?
Parameters:	None
Description:	Queries the list of available wavelengths.
Example:	sour:wav:ava? --- 1310, 1550<END>
Response:	List of available wavelength in nanometers (NM).
Errors:	None

4.3.2 SOURce:WAVelength Command

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.
Description:	Set 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.
Example:	sour:wav 1310
Response:	None
Errors:	(–200, "std_execGen, Test is Active") (–224, "std_illegalParmValue, Invalid Parameter Value")

4.3.3 SOURce:WAVelength? Command

Syntax:	SOURce:WAVelength?
Parameters:	None
Description:	Queries current wavelength. Available first wavelength will be returned when "Wavelength ALL" is set.
Example:	sour:wav? --- 1310<END>
Response:	Current wavelength value.
Errors:	None

4.3.4 SOURce:RANge:AVAilable? Command

Syntax:	SOURce:RANge:AVAilable?
Parameters:	None
Description:	Queries the list of available ranges for current wavelength settings.
Example:	sour:ran:ava? --- 5.0, 10.0, 20.0, 50.0, 100.0, 200.0, 300.0<END>
Response:	List of available ranges in kilometers (KM).
Errors:	None

4.3.5 SOURce:RANge Command

Syntax:	SOURce:RANge<wsp><value>
Parameters:	<value> Numeric format Range: Model dependent, numeric value returned by available ranges query command
Description:	Set current range
Example:	sour:ran 100
Response:	None
Errors:	(–200, "std_execGen, Test is Active") (–224, "std_illegalParmValue, Invalid Parameter Value")

4.3.6 SOURce:RANge? Command

Syntax:	SOURce:RANge?
Parameters:	None
Description:	Query current range.
Example:	sour:ran? --- 50.0<END>
Response:	The value is current range in kilometers (KM).
Errors:	None

4.3.7 SOURce:RESo:AVAilable? Command

Syntax:	SOURce:RESo:AVAilable?
Parameters:	None
Description:	Queries the list of available resolution flags for current range settings.
Example:	sour:res:ava? --- 0, 1, 2<END>
Response:	List of available resolutions: 0-Low Density, 1-High Density, 2-Very High Density
Errors:	None

4.3.8 SOURce:RESo Command

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.
Description:	Set current resolution.
Example:	sour:res 1
Response:	None
Errors:	(–200, "std_execGen, Test is Active") (–224, "std_illegalParmValue, Invalid Parameter Value")

4.3.9 SOURce:RESo? Command

Syntax:	SOURce:RESo?
Parameters:	None
Description:	Query current resolution settings flag.
Example:	sour:res? --- 1<END>
Response:	The value is current resolution: 0-Low Density, 1-High Density, 2-Very High Density
Errors:	None

4.3.10 SOURce:PULSe:AVAilable? Command

Syntax:	SOURce:PULSe:AVAilable?
Parameters:	None
Description:	Queries the list of available pulse width for current range/resolution settings.
Example:	sour:puls:ava? --- 10,20,50,100<END>
Response:	List of available pulse width in nanoseconds (NS).
Errors:	None

4.3.11 SOURce:PULSe Command

Syntax:	SOURce:PULSe<wsp><value>
Parameters:	<value> Numeric format Range: Integer PW value returned by querying available pulse width command. Available pulse width values are dependant on current range/resolution settings.
Description:	Set current pulse width.
Example:	sour:puls 100
Response:	None
Errors:	(–200, "std_execGen, Test is Active") (–224, "std_illegalParmValue, Invalid Parameter Value")

4.3.12 SOURce:PULSe? Command

Syntax:	SOURce:PULSe?
Parameters:	None
Description:	Query current pulse width.
Example:	sour:puls? --- 100<END>
Response:	value is current pulse width in nanoseconds (NS)
Errors:	None

4.3.13 SOURce:PULSe:ENHanced:AVAilable? Command

Syntax:	SOURce:PULSe:ENHanced:AVAilable?
Parameters:	None
Description:	Queries if the enhance mode is available for currently set pulse width.
Example:	sour:puls:enh:ava? --- 1<END>
Response:	Boolean value 1 or 0.
Errors:	None

4.3.14 SOURce:PULSe:ENHanced Command

Syntax:	SOURce:PULSe:ENHanced<wsp><value>
Parameters:	<value> Boolean format Range: 1 | 0.
Description:	Set current pulse width's enhanced mode.
Example:	sour:puls:enh 0
Response:	None
Errors:	(–200, "std_execGen, Test is Active") (–104, "std_wrongParamType, Data Type Error") (–224, "std_illegalParmValue, Invalid Parameter Value")

4.3.15 SOURce:PULSe:ENHanced? Command

Syntax:	SOURce:PULSe:ENHanced?
Parameters:	None
Description:	Query current pulse width's enhanced mode flag.
Example:	sour:puls:enh? --- 1<END>
Response:	Boolean value of the enhanced mode flag
Errors:	None

4.3.16 SOURce:AVERages:TIMe? Command

Syntax:	SOURce:AVERages:TIMe?
Parameters:	None
Description:	Queries number of seconds that have been set for the next test in manual mode.
Example:	sour:aver:tim? --- 120<END>
Response:	Number of seconds set to run the test.
Errors:	None

4.3.17 SOURce:AVERages:TIMe Command

Syntax:	SOURce:AVERages:TIMe<wsp><value>
Parameters:	<value> Integer format Range: 1 – 3600.
Description:	Sets number of seconds for the test duration for the next test in manual mode.
Example:	sour:aver:tim 120<END>
Response:	None
Errors:	(–200, "std_execGen, Test is Active") (–224, "std_illegalParmValue, Invalid Parameter Value")

4.4 SENSe Subsystem Commands

The SENSe subsystem lets you control/query OTDR's measurement parameters.

4.4.1 SENSe:AVERages? Command

Syntax:	SENSe:AVERages?
Parameters:	None
Description:	Queries number of averages that have been completed on the trace or since the test started
Example:	sens:aver? --- 4096<END>
Response:	Number of averages
Errors:	(–400, "std_queryGen, Trace Not Ready")

4.4.2 SENSe:AVERages:TIMe? Command

Syntax:	SENSe:AVERages:TIMe?
Parameters:	None
Description:	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.
Example:	sens:aver:tim? --- 28<END>
Response:	Number of seconds
Errors:	(–400, "std_queryGen, Trace Not Ready")

4.4.3 SENSe:TRACe:READY? Command

Syntax:	SENSe:TRACe:READY?
Parameters:	None
Description:	Queries if trace data is ready.
Example:	sens:trac:ready? --- 1<END>
Response:	Possible response values: 1 = trace data is ready and can be transferred. 0 = no trace data available in the memory.
Errors:	None

4.4.4 SENSe:CONCheck? Command

Syntax:	SENSe:CONCheck?
Parameters:	None
Description:	Queries if connection check option is ON.
Example:	sens:conc? --- 1<END>
Response:	Possible response values: 1 = Connection check is ON. 0 = Connection check is OFF.
Errors:	None

4.4.5 SENSe:CONCheck Command

Syntax:	SENSe:CONCheck<wsp><value>
Parameters:	<value> Boolean value 1 or on = Connection check is ON. 0 or off = Connection check is OFF.
Description:	Sets Connection Check option ON or OFF.
Example:	sens:conc ON<END>
Response:	NONE
Errors:	(–200, "std_execGen, Test is Active") (–104, "std_wrongParamType, Data Type Error")

4.4.6 SENSe:LIVCheck? Command

Syntax:	SENSe:LIVCheck?
Parameters:	None
Description:	Queries if live fiber check option is ON.
Example:	sens:livc? --- 1<END>
Response:	Possible response values: 1 = Live fiber check is ON. 0 = Live fiber check is OFF.
Errors:	None

4.4.7 SENSe:LIVCheck:Command

Syntax:	SENSe:LIVCheck<wsp><value>
Parameters:	<value> Boolean value 1 or on = Live fiber check is ON. 0 or off = Live fiber check is OFF.
Description:	Sets live fiber check option ON or OFF.
Example:	sens:livc ON<END>
Response:	NONE
Errors:	(–200, "std_execGen, Test is Active") (–104, "std_wrongParamType, Data Type Error")

4.4.8 SENSe:FIBer:IOR Command

Syntax:	SENSe:FIBer:IOR<wsp><value>
Parameters:	<value> Floating point format Range: 1.400000 – 1.699999
Description:	Sets index of refraction. This value will be used for the next test.
Example:	sens:fib:ior 1.45
Response:	None
Errors:	(–200, "std_execGen, Test is Active") (–224, "std_illegalParmValue, Invalid Parameter Value")

4.4.9 SENSe:FIBer:IOR? Command

Syntax:	SENSe:FIBer:IOR?
Parameters:	None
Description:	Queries index of refraction.
Example:	sens:fib:ior? --- 1.450000<END>
Response:	Possible response is value in range: 1.400000 – 1.699999
Errors:	None

4.4.10 SENSe:FIBer:BSC Command

Syntax:	SENSe:FIBer:BSC<wsp><value>
Parameters:	<value> Floating point format Range: –90.0 - –40.0
Description:	Sets backscatter coefficient. This value will be used for the next test.
Example:	sens:fib:bsc –83.0
Response:	None
Errors:	(–200, "std_execGen, Test is Active") (–224, "std_illegalParmValue, Invalid Parameter Value")

4.4.11 SENSe:FIBer:BSC? Command

Syntax:	SENSe:FIBer:BSC?
Parameters:	None
Description:	Queries backscatter coefficient.
Example:	sens:fib:bsc? --- –83.0<END>
Response:	Possible response is value in range: –90.0 - –40.0
Errors:	None

4.4.12 SENSe:HOFFset Command

Syntax:	SENSe:HOFFset<wsp><value>
Parameters:	<value> Floating point format Range: Offset value can be set plus/minus maximum distance range. (-300 to 300km)
Description:	Set horizontal offset for the displayed trace(s). Offset value units – KM.
Example:	Sens:hoff 10.0
Response:	None
Errors:	(–224, "std_illegalParmValue, Invalid Parameter Value")

4.4.13 SENSe:HOFFset? Command

Syntax:	SENSe:HOFFset?
Parameters:	None
Description:	Queries horizontal offset for the displayed trace(s).
Example:	sens:hoff? --- 0<END>
Response:	Current horizontal offset value.
Errors:	None

4.4.14 SENSe:VOFFset Command

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)
Description:	Set vertical offset for the displayed trace(s). Offset value units – DB.
Example:	sens:voff –5.0
Response:	None
Errors:	(–224, "std_illegalParmValue, Invalid Parameter Value")

4.4.15 SENSe:VOFFset? Command

Syntax:	SENSe:VOFFset?
Parameters:	None
Description:	Queries vertical offset for the displayed trace(s).
Example:	sens:voff? --- 0<END>
Response:	Current vertical offset value.
Errors:	None

4.4.16 SENSe:ACURsor Command

Syntax:	SENSe:ACURsor<wsp><value>
Parameters:	<value> Floating point format Range: 0.0 – Current distance range.
Description:	Set A cursor position. Cursor position units – KM.
Example:	sens:acur 20.5
Response:	None
Errors:	(–224, "std_illegalParmValue, Invalid Parameter Value")

4.4.17 SENSe:ACURsor? Command

Syntax:	SENSe:ACURsor?
Parameters:	None.
Description:	Queries current A cursor position.
Example:	sens:acur? --- 20.5<END>
Response:	Current A cursor position value.
Errors:	None

4.4.18 SENSe:BCURsor Command

Syntax:	SENSe:BCURsor<wsp><value>
Parameters:	<value> Floating point format Range: 0.0 – Current distance range.
Description:	Set B cursor position. Cursor position units – Km.
Example:	sens:bcur 20.5
Response:	None
Errors:	(–224, "std_illegalParmValue, Invalid Parameter Value")

4.4.19 SENSe:BCURsor? Command

Syntax:	SENSe:BCURsor?
Parameters:	None.
Description:	Queries current B cursor position.
Example:	sens:bcur? --- 20.5<END>
Response:	Current B cursor position value in Km.
Errors:	None

4.4.20 SENSe:LSALeft Command

Syntax:	SENSe:LSALeft<wsp><start>,<stop>
Parameters:	<start> Floating point format Range: –100.0 – 400.0. <stop> Floating point format Range: –100.0 – 400.0. Start value must be less or equal to stop value.
Description:	Set start and stop for left LSA marker. Start and stop units – kilometers (KM).
Example:	sens:lsal 0.0, 0.5
Response:	None
Errors:	(–224, "std_illegalParmValue, Invalid Parameter Value") (–200, "std_execGen, LSA Inactive State")

4.4.21 SENSe:LSALeft? Command

Syntax:	SENSe:LSALeft?
Parameters:	None
Description:	Query left LSA values.
Example:	sens:lsal? --- 0.0, 0.5<END>
Response:	Start and stop for left LSA marker. Start and stop units – kilometers (KM).
Errors:	(–400, "std_queryGen, LSA Inactive State")

4.4.22 SENSe:LSARight Command

Syntax:	SENSe:LSARight<wsp><start>,<stop>
Parameters:	<start> Floating point format Range: –100.0 – 400.0. <stop> Floating point format Range: –100.0 – 400.0. Start value must be less or equal to stop value.
Description:	Set start and stop for right LSA marker. Start and stop units – kilometers (KM).
Example:	sens:lsar 0.0, 0.5
Response:	None
Errors:	(–224, "std_illegalParmValue, Invalid Parameter Value") (–200, "std_execGen, LSA Inactive State")

4.4.23 SENSe:LSARight? Command

Syntax:	SENSe:LSARight?
Parameters:	None
Description:	Query right LSA values.
Example:	sens:lsar? --- 0.0,0.5<END>
Response:	Start and stop for right LSA marker. Start and stop units – kilometers (KM).
Errors:	(–400, "std_queryGen, LSA Inactive State")

4.4.24 SENSe:LOSS:MODE Command

Syntax:	SENSe:LOSS:MODE<wsp><value>
Parameters:	<value> Integer format Range: 0 | 1 | 2 | 3 | 4 | 5 | 6 0 = Splice Loss Mode 1 = Two-point Loss Mode 2 = Two-point LSA 3 = dB/km Loss Mode 4 = dB/km LSA Loss Mode 5 = Two-point Attenuation Corrected Loss Mode 6 = ORL Loss Mode
Description:	Set current Loss Mode.
Example:	sens:loss:mode 0
Response:	None
Errors:	(–224, "std_illegalParmValue, Invalid Parameter Value")

4.4.25 SENSe:LOSS:MODE? Command

Syntax:	SENSe:LOSS:MODE?
Parameters:	none
Description:	Query current Loss Mode.
Example:	sens:loss:mode? --- 0<END>
Response:	Possible responses are: 0 = Splice Loss Mode 1 = Two-point Loss Mode 2 = Two-point LSA 3 = dB/km Loss Mode 4 = dB/km LSA Loss Mode 5 = Two-point Attenuation Corrected Loss Mode 6 = ORL Loss Mode
Errors:	None

4.4.26 SENSe:ORL:MODE Command

Syntax:	SENSe:ORL:MODE<wsp><value>
Parameters:	<value> Integer format Range: 0 | 1 | 2 0 = A Cursor 1 = Origin 2 = Full Trace
Description:	Set current ORL Mode.
Example:	sens:orl:mode 0
Response:	None
Errors:	(–224, "std_illegalParmValue, Invalid Parameter Value")

4.4.27 SENSe:ORL:MODE? Command

Syntax:	SENSe:ORL:MODE?
Parameters:	None
Description:	Query current ORL Mode.
Example:	sens:orl:mode? --- 0<END>
Response:	Possible responses are: 0 = A Cursor 1 = Origin 2 = Full Trace
Errors:	None

4.4.28 SENSe:ANALyze:PARameters Command

Syntax:	SENSe:ANALyze:PARameters<wsp><spliceloss>,<reflectance>,<endloss>,<pon loss>
Parameters:	Double values: <splice loss> Range: 0.01 – 9.99 <reflectance> Range: –70.0 - –20.0 <end loss> Range: 1 - 99 <pon loss> Range: 1.0 – 30.0
Description:	Set Analysis Parameters for the next test.
Example:	sens:anal:par 0.05,–60.0,3.0,10.0
Response:	None
Errors:	(–224, "std_illegalParmValue, Invalid Parameter Value")

4.4.29 SENSe:ANALyze:PARameters? Command

Syntax:	SENSe:ANALyze:PARameters?
Parameters:	none
Description:	Query current Analysis parameters.
Example:	sens:anal:par? --- 0.050000, –60.000000, 3.000000,10.000000<END>
Response:	<event loss>,<reflectance>,<end loss>,<pon loss><END>
Errors:	None

4.4.30 SENSe:ANALyze:AUTO Command

Syntax:	SENSe:ANALyze:AUTO<wsp><value>
Parameters:	<value> Boolean format Range: 0 | 1 0 = Auto-analysis is OFF. 1 = Auto-analysis is ON.
Description:	Set ON/OFF analysis to be performed automatically after the test is complete.
Example:	sens:anal:auto 0
Response:	None
Errors:	(–104, "std_wrongParamType, Data Type Error")

4.4.31 SENSe:ANALyze:AUTO? Command

Syntax:	SENSe:ANALyze:AUTO?
Parameters:	None
Description:	Query if auto-analysis after test is set to ON.
Example:	sens:anal:auto? --- 0<END>
Response:	Possible responses are: 0 = Auto-analysis is OFF. 1 = Auto-analysis is ON.
Errors:	None

4.5 TRACe Subsystem Commands

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

4.5.1 TRACe:PARameters? Command

Syntax:	TRACe:PARameters?
Parameters:	None
Description:	Get main OTDR parameters used to collect the trace data. <wave>, <range>, <pulse>, <avg>, <reso>, <ior>, <bsc>, <enh><END>
Example:	trac:par? --- 1310, 16.415554, 50, 6144, 0.656621, 1.467700, –78.500000, 1<END>
Response:	Trace parameters as comma separated numeric values.
Errors:	(–400, "std_queryGen, Trace Not Ready")

4.5.2 TRACe:ANALyze Command

Syntax:	TRACe:ANALyze
Parameters:	<none>
Description:	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.
Example:	trac:anal
Response:	None
Errors:	(–200, "std_execGen, Test is Active") (–200, "std_execGen, Trace Not Ready")

4.5.3 TRACe:ANALyze? Command

Syntax:	TRACe:ANALyze?
Parameters:	None
Description:	Query if analysis is done on the trace
Example:	trac:anal? --- 0<END>
Response:	Possible responses are: 0 = Trace not analyzed. 1 = Trace is analyzed.
Errors:	(–400, "std_queryGen, Trace Not Ready")

4.5.4 TRACe:ANALyze:ORL Command

Syntax:	TRACe:ANALyze:ORL
Parameters:	<none>
Description:	Performs ORL calculations on the trace. This command is an overlapped command.
Example:	trac:anal:orl
Response:	None
Errors:	(–200, "std_execGen, Test is Active") (–200, "std_execGen, Trace Not Ready") (–200, "std_execGen, Invalid Loss Mode")

4.5.5 TRACe:MDLOss? Command

Syntax:	TRACe:MDLOss?
Parameters:	None
Description:	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.
Example:	trac:mdlo? --- –4.610,–99.99<END>
Response:	Calculated loss values. Two comma separated numeric values.
Errors:	(–200, "std_execGen, Test is Active") (–400, "std_queryGen, Trace Not Ready")

4.5.6 TRACe:EELOss? Command

Syntax:	TRACe:EELOss?
Parameters:	None
Description:	Get trace end-to-end loss. If loss mode is not calculable the returned value will be –99.99.
Example:	trac:eelo? --- –4.610<END>
Response:	Calculated end-to-end loss value.
Errors:	(–200, "std_execGen, Test is Active") (–400, "std_queryGen, Trace Not Ready")

4.5.7 TRACe:LOAD:SOR? Command

Syntax:	TRACe:LOAD:SOR?
Parameters:	None
Description:	Get SOR trace object. Refer to Section 2.5 "Message Format – Binary Data" for further details.
Example:	trac:load:sor? --- "BINARY ARRAY" #524047 //SCPI data size message for binary data transfer followed by array of SOR file bytes.
Response:	SOR trace file as an array of bytes per SCPI binary transfer specifications
Errors:	(–200, "std_execGen, Test is Active") (–400, "std_queryGen, Trace Not Ready")

4.5.8 TRACe:LOAD:TEXT? Command

Syntax:	TRACe:LOAD:TEXT? TRACe:LOAD:TEXT?<wsp><start> TRACe:LOAD:TEXT?<wsp><start>,<end>
Parameters:	<start> Floating point format Range: 0.000000 – Current distance range. <end> Floating point format Range: 0.000000 – Current distance range. Start value must be less or equal to end value.
Description:	TRACe:LOAD:TEXT? command has 3 mode. 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 form 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 form specified start to end position. Refer to Section 2.5 "Message Format – Binary Data" for further details.

(Cont'd)

Example:	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
	
	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

Response:	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")

4.5.9 TRACe:LOAD:DATA? Command

Syntax:	TRACe:LOAD:DATA?<wsp><start>,<end>,<space>
Parameters:	<start> Starting distance (km) Floating point format Range: 0.0 — (Current distance range - <space>*resolution) <end> Ending distance (km) Floating point format Range: (<start>+<space>*resolution) — (Current distance range). <space> data point spacing in terms of resolution Numeric format Range: 1 — ((Max number of data points between <start> and <end) distance) <start> Starting distance (km) Numeric format Range: 0.0 — (Current distance range - <space>*resolution) Start value must be less or equal to end value.
Description:	TRACe:LOAD:DATA? command has 4 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 Section 2.5 “Message Format – Binary Data” for further details.
Example:	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
Response:	Requested data points as an array of bytes per SCPI binary transfer specifications.
Errors:	(–225, "std_illegalState, Trace Not Ready") (–224, "std_illegalValue, Invalid Parameter Value")

4.5.10 TRACe:HEADer Command

Syntax:	TRACe:HEADer<wsp><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
Description:	Set 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.
Example:	trac:head OT,1,1,ABC,Tokyo,Yokohama,1,Anritsu,TEST trac:head BC,,,,,,0,,,
Response:	None.
Errors:	(-108, "std_tooManyParameters, Parameter not Allowed") (-109, "std_tooFewParameters, Missing Parameter") (-224, "std_illegalParmValue, Invalid Parameter Value")

4.5.11 TRACe:HEADer? Command

Syntax:	TRACe:HEADer?
Parameters:	None.
Description:	Get Trace Header
Example:	trac:head? --- OT, 1, 1, ABC, Tokyo, Yokohama, 1, Anritsu, TEST<END>
Response:	Nine header.
Errors:	(-108, "std_tooManyParameters, Parameter not Allowed")

4.5.12 TRACe:STORe:SOR Command

Syntax:	TRACe:STORe:SOR<wsp><filepath&name>
Parameters:	<filepath&name> File path and file name to save the SOR file
Description:	Store the SOR file in internal memory of MT9083.
Example:	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.
Response:	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")

4.6 DISPlay Subsystem Commands

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

4.6.1 DISPlay:MODE Command

Syntax:	DISPlay:MODE<wsp><mode>
Parameters:	<mode> Display mode Numeric format Range: 0 | 1 | 2 0 – From A, 1 – From B, 2 – From Origin
Description:	Set display mode (From Origin, From A, From B).
Example:	disp:mode 2
Response:	None
Errors:	(–224, "std_illegalValue, Invalid Parameter Value")

4.6.2 DISPlay:MODE? Command

Syntax:	DISPlay:MODE?
Parameters:	None
Description:	Get display mode (From Origin, From A, From B).
Example:	disp:mode? --- 1<END>
Response:	Display mode as numeric values.
Errors:	None

4.6.3 DISPlay:ZOOM:FULL Command

Syntax:	DISPlay:ZOOM:FULL
Parameters:	None
Description:	Set display zoom to full trace in current display mode
Example:	disp:zoom:full
Response:	None
Errors:	None

4.6.4 DISPlay:ZOOM:HORIzontal Command

Syntax:	DISPlay:ZOOM:HORIzontal<wsp><level>
Parameters:	<level> Zoom level Numeric format Range: 0 – (6 - 16) Depends on current distance range 0 – Full zoom in, (6 – 16) – Full Zoom out *Note:* The maximum value for each distance is as follows. 0.5 km : 6 1 km : 7 2.5 km : 8 5 km : 9 10 km : 10 25 km : 11 50 km : 12 100 km : 13 200 km : 14 300 km : 16
Description:	Set Horizontal display zoom to specified level
Example:	disp:zoom:hori 2
Response:	None
Errors:	(–224, "std_illegalValue, Invalid Parameter Value")

4.6.5 DISPlay:ZOOM:HORIzontal? Command

Syntax:	DISPlay:ZOOM:HORIzontal?
Parameters:	None
Description:	Get Horizontal display zoom level
Example:	disp:zoom:hori? --- 5<END>
Response:	Horizontal zoom level as numeric value.
Errors:	None

4.6.6 DISPlay:ZOOM:VERTical Command

Syntax:	DISPlay:ZOOM:VERTical<wsp><level>
Parameters:	<level> Zoom level Numeric format Range: 0 – 7 0 – Full zoom in, 7 – Full Zoom out
Description:	Set Vertical display zoom to specified level
Example:	disp:zoom:vert 2
Response:	None
Errors:	(–224, "std_illegalValue, Invalid Parameter Value")

4.6.7 DISPlay:ZOOM:VERTical? Command

Syntax:	DISPlay:ZOOM:VERTical?
Parameters:	None
Description:	Get Vertical display zoom level
Example:	disp:zoom:vert? --- 5<END>
Response:	Vertical zoom level as numeric value.
Errors:	None

4.6.8 DISPlay:SCALe:HORIzontal Command

Syntax:	DISPlay:SCALe:HORIzontal<wsp><scale>
Parameters:	<scale> Horizontal scale range value in km Floating point format Range: 0.0096 – (0.5040 - 300.0000) Depends on current distance range (0.5 – 300) km 0.0096 – Full zoom in, (0.5040 - 300.0000) – Full Zoom out Note: There are 17 predefined scale ranges (0.0096, 0.0144, 0.0288, 0.0528, 0.1008, 0.2544, 0.5040, 1.0032, 2.5008, 5.0016, 10.0032, 25.0032, 50.0016, 100.0032, 200.0016, 250.0032, 300.0000), the specified <scale> value will snap to the nearest highest predefined scale value. The maximum value for each distance is as follows. 0.5 km : 0.5040 1 km : 1.0032 2.5 km : 2.5008 5 km : 5.0016 10 km : 10.0032 25 km : 25.0032 50 km : 50.0016 100 km : 100.0032 200 km : 200.0016 300 km : 300.0000
Description:	Set Horizontal scale range to specified value The distance specified with Horizontal scale is displayed on screen.
Example:	disp:scal:hori 5.0
Response:	None
Errors:	(–224, "std_illegalValue, Invalid Parameter Value")

4.6.9 DISPlay:SCALe:HORIzontal? Command

Syntax:	DISPlay:SCALe:HORIzontal?
Parameters:	None
Description:	Get Horizontal display scale range value in km
Example:	disp:scal:hori? --- 5.0016<END>
Response:	Horizontal scale range value in km as numeric value.
Errors:	None

4.6.10 DISPlay:SCALe:VERTical Command

Syntax:	DISPlay:SCALe:VERTical<wsp><scale>
Parameters:	<scale> Vertical scale range value in dB Floating point format Range: 0.5 – 65 0.5 – Full zoom in, 65 – Full Zoom out Note: There are 8 predefined scale ranges (0.5, 1.25, 2.5, 5, 12.5, 25, 50, 65), the specified <scale> value will snap to the nearest highest predefined scale value.
Description:	Set Vertical display scale range to specified value The distance specified with Vertical display scale is displayed on screen.
Example:	disp:scal:vert 0.5
Response:	None
Errors:	(–224, "std_illegalValue, Invalid Parameter Value")

4.6.11 DISPlay:SCALe:VERTical? Command

Syntax:	DISPlay:SCALe:VERTical?
Parameters:	None
Description:	Get Vertical display scale range value in dB
Example:	disp:scal:vert? --- 12.5<END>
Response:	Vertical scale range in dB as numeric value.
Errors:	None

Appendix A Recommended USB-Ethernet converter

These USB-Ethernet converters manufactured by other companies are confirmed to work.

Manufacturer	Model name
Planex	UE-200TX-G
Buffalo	LUA-U2-KTX
Buffalo	LUA2-U2-ATX
D-Link	DUB-E100
Linksys	USB200M
Linksys	USB300M