MT1000A Network Master Pro
OTDR Modules
Remote Scripting Operation Manual

Eighth Edition

ANRITSU CORPORATION

Document No.: M-W3859AE-8.0
For safety and warning information, please read this manual before attempting to use the equipment.

Additional safety and warning information is provided within the MT1000A Network Master Pro OTDR Modules Operation Manual (M-W3810AE). Please also refer to it before using the equipment.

Keep this manual with the equipment.

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

This operation manual describes the SCPI (Standard Commands for Programmable Instruments) commands for Network Master Pro OTDR modules.

Note: SCPI commands described in this manual are supported in Network Master Pro version 11.00.

Some commands or queries in this manual may require that specific hardware or software options are installed. These options must be purchased separately.

This operation manual uses the notations described in the following standards:

- IEEE: Std 488.2-1992
- SCPI: VERSION 1999.0 (SCPI Consortium)
Contents

1 Overview ................................................. 9
  1.1 Ethernet Based Remote Control .......................... 9
    1.1.1 Connecting Cable .................................. 9
    1.1.2 Ethernet Remote Control Settings .................. 10
    1.1.3 Communication Buffers .............................. 10
  1.2 Program Messages ..................................... 11
    1.2.1 Program Message Unit .............................. 11
    1.2.2 Program Headers .................................. 12
    1.2.3 Program Data ...................................... 13
    1.2.4 Program Message Terminator ....................... 14
    1.2.5 Compound Program Messages ....................... 14
    1.2.6 Sequential Execution .............................. 14
  1.3 Response Messages .................................... 15
    1.3.1 Response Data ..................................... 15
    1.3.2 Response Messages Terminator ..................... 17
    1.3.3 Prompt ........................................... 17
  1.4 Status ................................................ 18
    1.4.1 IEEE488.2 Standard Status and SCPI-defined Registers .................................................. 18
    1.4.2 Network Master Unique Status Registers ........... 22
    1.4.3 Reading, Writing and Clearing Status Registers ................................. 25
  1.5 Controller Example .................................... 27
    1.5.1 PuTTY ............................................. 27
  1.6 Definitions ........................................... 30
    1.6.1 NaN (Not a Number) ................................. 30
    1.6.2 Right Arrow ....................................... 30
    1.6.3 Data Bit (DB) ..................................... 30
    1.6.4 Port Number (Logical Port) ........................ 30

2 SCPI Conformance Information .................. 31
  2.1 SCPI Version .......................................... 31
  2.2 IEEE 488.2 Mandatory Commands .................. 32
    2.2.1 *CLS ............................................. 32
    2.2.2 *ESE ............................................. 32
    2.2.3 *ESR? ............................................. 33
    2.2.4 *IDN? ............................................. 33
    2.2.5 *OPC ............................................. 33
    2.2.6 *RST ............................................. 34
    2.2.7 *SRE ............................................. 34
    2.2.8 *STB? ............................................. 35
    2.2.9 *TST? ............................................. 35
    2.2.10 *WAI ............................................ 35
  2.3 SCPI System Subsystem Commands .................. 36
    2.3.1 SYSTem:VERSion? .................................. 36
    2.3.2 SYSTem:ERRor[:NEXT]? .................................. 36
    2.3.3 SYSTem:ERRor:ADDitional[:MESSage] .................. 36
    2.3.4 SYSTem:DATE ..................................... 37
    2.3.5 SYSTem:TIME ..................................... 37
    2.3.6 SYSTem:REBoot .................................... 37
    2.3.7 SYSTem:GPS:NSATellites? .......................... 38
| 2.3.8 | SYSTem:GPS:TIME? | 38 |
| 2.3.9 | SYSTem:GPS:LOCation? | 38 |
| 2.3.10 | SYSTem:GPS:POWER | 38 |
| 2.3.11 | SYSTem:GPS:POWER? | 38 |
| 2.3.12 | SYSTem:GPS:SYNCmode | 38 |
| 2.3.13 | SYSTem:GPS:SYNCmode? | 39 |
| 2.3.14 | SYSTem:GPS:ANTNpower | 39 |
| 2.3.15 | SYSTem:GPS:ANTNpower? | 39 |
| 2.3.16 | SYSTem:GPS:PWRElapsed? | 39 |
| 2.3.17 | SYSTem:GPS:LOCElapsed? | 39 |
| 2.3.18 | SYSTem:GPS:SYNElapsed? | 39 |
| 2.3.19 | SYSTem:COMMunicate:TERMinator | 40 |
| 2.3.20 | SYSTem:PROMpt | 40 |
| 2.3.21 | SYSTem:LOCal:CONTrol | 40 |
| 2.3.22 | SYSTem:NCONnectivity:COMunication:AUTH:TYPE | 41 |
| 2.3.23 | SYSTem:NCONnectivity:COMunication:BASic | 41 |
| 2.3.24 | SYSTem:NCONnectivity:COMunication:TURI | 41 |
| 2.3.25 | SYSTem:NCONnectivity:COMunication:SET | 42 |
| 2.3.26 | SYSTem:NCONnectivity:COMunication:LRETurn:RRESPonse? | 42 |
| 2.3.27 | SYSTem:NCONnectivity:COMunication:LRETurn:RSTatus? | 42 |
| 2.3.28 | SYSTem:NCONnectivity:COMunication:LRETurn:ERRor? | 42 |
| 2.3.29 | SYSTem:WAIT:[IDLE] | 43 |
| 2.3.30 | SYSTem:WAIT:DURation | 43 |

### 2.4 SCPI Instrument Subsystem Commands

| 2.4.1 | Connection to Application Server | 44 |
| 2.4.2 | Connection to multiple applications | 44 |
| 2.4.3 | Connection from multiple users | 44 |
| 2.4.4 | Force Termination of Application Server | 44 |
| 2.4.5 | INSTRument:START:[DEFault] | 45 |
| 2.4.6 | INSTRument:START:LAST | 46 |
| 2.4.7 | INSTRument:START:GUI | 46 |
| 2.4.8 | INSTRument:TERMinate | 46 |
| 2.4.9 | INSTRument:TERMinate:FORCe | 47 |
| 2.4.10 | INSTRument:COUNT? | 48 |
| 2.4.11 | INSTRument:CATalog? | 48 |
| 2.4.12 | INSTRument:STATe? | 48 |
| 2.4.13 | INSTRument:CONNect | 48 |
| 2.4.14 | INSTRument:CONNect:ALL | 49 |
| 2.4.15 | INSTRument:CONNect:CATalog[] | 49 |
| 2.4.16 | INSTRument:DISConnect | 49 |
| 2.4.17 | INSTRument:[SELect] | 49 |
| 2.4.18 | INSTRument:ERRor[:NEXT]? | 50 |
| 2.4.19 | INSTRument:PORT? | 50 |
| 2.4.20 | INSTRument:PORT:FREE? | 50 |
| 2.4.21 | INSTRument:PORT:CATalog? | 51 |
| 2.4.22 | INSTRument:MODule:CATalog? | 51 |
| 2.4.23 | INSTRument:CTRL:NAME? | 51 |
| 2.4.24 | INSTRument:CTRL:SN? | 51 |
| 2.4.25 | INSTRument:CTRL:TRT? | 51 |
| 2.4.26 | INSTRument:CTRL:OPTion:CATalog? | 51 |
| 2.4.27 | INSTRument:MODule<Md>:NAME? | 52 |
| 2.4.28 | INSTRument:MODule<Md>:SN? | 52 |
| 2.4.29 | INSTRument:MODule<Md>:TRT? | 52 |
| 2.4.30 | INSTRument:MODule<Md>:OPTion:CATalog? | 52 |

### 2.5 SCPI Status Subsystem Commands

| 2.5.1 | STATus:OPERation:[EVENt]? | 53 |
| 2.5.2 | STATus:OPERation:CONDition? | 53 |
| 2.5.3 | STATus:OPERation:ENABle | 53 |
| 2.5.4 | STATus:OPERation:PTRansition | 54 |
3 Standard OTDR Application Commands

3.1 Measurement Conditions
3.1.1 OTDR:SOURce:PORT
3.1.2 OTDR:SOURce:TEST
3.1.3 OTDR:SOURce:WAVElength:AVAILable?
3.1.4 OTDR:SOURce:WAVElength
3.1.5 OTDR:SOURce:RANGE:AVAILable?
3.1.6 OTDR:SOURce:RANGE
3.1.7 OTDR:SOURce:RESo:AVAILable?
3.1.8 OTDR:SOURce:RESo
3.1.9 OTDR:SOURce:PILot:AVAILable?
3.1.10 OTDR:SOURce:PILot
3.1.11 OTDR:SOURce:AVERages:TIMe
3.1.12 OTDR:SOURce:AVERages
3.2 IOR/BSC
3.2.1 OTDR:SENSE:FIBer:IOR
3.2.2 OTDR:SENSE:FIBer:BSC
3.3 Splitter
3.3.1 OTDR:SOURce:SPLITter
3.4 Status
3.4.1 OTDR:SENSE:AVERages:TIMe?
3.4.2 OTDR:SENSE:TRACe:READY?
3.5 Measurement Functions
3.5.1 OTDR:SENSE:CONCheck
3.5.2 OTDR:SENSE:CONState?
3.5.3 OTDR:CONTinue
3.5.4 OTDR:SENSE:LIVCheck
3.5.5 OTDR:SENSE:FIACheck
3.6 Analysis
3.6.1 OTDR:SENSE:PATCH:LAUNCH
3.6.2 OTDR:SENSE:PATCH:LAUNCH:CORrelation
3.6.3 OTDR:SENSE:PATCH:LOAD
3.6.4 OTDR:SENSE:PATCH:RECeive:CORrelation
3.6.5 OTDR:SENSe:ACURsor 73
3.6.6 OTDR:SENSe:BCURsor 73
3.6.7 OTDR:SENSe:LSAleft 74
3.6.8 OTDR:SENSe:LSARight 74
3.6.9 OTDR:SENSe:LOSS:MODE 74
3.6.10 OTDR:SENSe:ORL:MODE 75
3.6.11 OTDR:SENSe:ANALyze:PARameters 75

3.7 TRACE 77
3.7.1 OTDR:TRACe:PARameters? 77
3.7.2 OTDR:TRACe:ANALyze 77
3.7.3 OTDR:TRACe:ANALyze:ORL 77
3.7.4 OTDR:TRACe:MDLOss? 77
3.7.5 OTDR:TRACe:EELOss? 77
3.7.6 OTDR:TRACe:LOAD:TEXT? 78
3.7.7 OTDR:TRACe:HOPFset? 78

4 OLTS Application Commands 79
4.1 Loss Test Set 79
4.1.1 OLTS:SOURce:POWer:WAVelength:AVAilable? 79
4.1.2 OLTS:SOURce:POWer:WAVelength 79
4.1.3 OLTS:SOURce:AM:FREQuency:AVAilable? 79
4.1.4 OLTS:SOURce:AM:FREQuency 80
4.1.5 OLTS:SOURce:POWer:STATe 80
4.1.6 OLTS:SENSe:POWer:WAVelength:AVAilable? 80
4.1.7 OLTS:SENSe:POWer:WAVelength 80
4.1.8 OLTS:SENSe:CORRection:COLLect:ZERO 81
4.1.9 OLTS:SENSe:AVERage:COUNt 81
4.1.10 OLTS:SENSe:POWer:REFerence 81
4.1.11 OLTS:SENSe:POWer:THReshold 82
4.1.12 OLTS:FETCh:POWer 82

5 Measurement 83
5.1 Measurement Start and Stop 83
5.1.1 MEASurement:APPLication? 83
5.1.2 MEASurement:STARt 83
5.1.3 MEASurement:STOP 83
5.1.4 MEASurement:RESult:SUMMary? 83

A Example Scripts 85
A.1 Hints 85
A.2 OTDR Measurement 86
A.3 Sending an HTTP GET request 87
A.4 Setting parameters for uploading scenario results 88

List of Figures

1.1 System setup using Ethernet 9
1.2 Connector panel 9
1.3 Configure TCP Port for Remote Control 10
1.4 Program message structure 11
1.5 Program message unit 11
1.6 Response message structure 15
1.7 IEEE488.2 standard status and SCPI-defined registers/queues 18
1.8 The Network Master Unique Status registers for some of the supported interfaces 22
<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.9</td>
<td>The general structure for the Alarms and Errors status register for the interfaces</td>
<td>23</td>
</tr>
<tr>
<td>1.10</td>
<td>The structure for the Port Status register</td>
<td>23</td>
</tr>
<tr>
<td>1.11</td>
<td>The register model for the Network Master Unique Status registers</td>
<td>24</td>
</tr>
<tr>
<td>1.12</td>
<td>Enable <strong>Implicit CR in every LF in PuTTY</strong></td>
<td>27</td>
</tr>
<tr>
<td>1.13</td>
<td><strong>IP Address</strong> on the instrument</td>
<td>28</td>
</tr>
<tr>
<td>1.14</td>
<td>Specify the destination and <strong>Open</strong> the connection in PuTTY</td>
<td>28</td>
</tr>
<tr>
<td>1.15</td>
<td>Connection established with PuTTY</td>
<td>29</td>
</tr>
<tr>
<td>1.16</td>
<td><strong>Data bit</strong></td>
<td>30</td>
</tr>
<tr>
<td>1.17</td>
<td>The physical and logical port number of the OTDR module</td>
<td>30</td>
</tr>
<tr>
<td>2.1</td>
<td><strong>INST:STAR</strong> automatically <strong>connects</strong> and selects started application server</td>
<td>47</td>
</tr>
<tr>
<td>2.2</td>
<td>Original client session has to <strong>disconnect</strong>, before another client session can <strong>connect</strong></td>
<td>47</td>
</tr>
<tr>
<td>2.3</td>
<td>When connected to multiple application servers, client session <strong>selects</strong> to which application server the application specific commands are dispatched.</td>
<td>47</td>
</tr>
</tbody>
</table>
Chapter 1

Overview

The Network Master command based remote control functions support the built-in Ethernet service interface. Software specifications are in conformity with the IEEE488.2 standard based on SCPI version 1999 (Standard Commands for Programmable Instruments). Network Master becomes an automated measurement instrument when it is connected to an external controller.

1.1 Ethernet Based Remote Control

1.1.1 Connecting Cable

To use remote control via the Ethernet service interface, connect an Ethernet cable to the Ethernet connector next to the power socket.

Figure 1.1: System setup using Ethernet

Figure 1.2: Connector panel
1.1.2 Ethernet Remote Control Settings

Port Number

To change a TCP port number (for a valid range, see Table 1.1) type the number in the TCP Port field (see Figure 1.3).

![Figure 1.3: Configure TCP Port for Remote Control](image)

<table>
<thead>
<tr>
<th>Setup item</th>
<th>Description</th>
<th>Allowable range</th>
</tr>
</thead>
<tbody>
<tr>
<td>Port Number</td>
<td>TCP Port Number</td>
<td>1024 to 65535 (default: 56001)</td>
</tr>
</tbody>
</table>

Table 1.1: Allowable TCP port range

1.1.3 Communication Buffers

The input- and output streams are buffered. Besides the TCP receive buffer (87380 bytes) and the TCP transmit buffer (16384 bytes), the two streams share a common command/response buffer of 32 entries. Each buffer entry can hold a compound program message of maximum 4 KB or a response message of maximum 64 KB.

Program data transferred as <ARBITRARY BLOCK PROGRAM DATA> does not go through the internal buffer, but is streamed directly from the TCP receive buffer to the internal file system. Similar for response data of type <DEFINITE LENGTH ARBITRARY BLOCK RESPONSE DATA>; it is streamed directly from the internal file system to the TCP transmit buffer.
1.2 Program Messages

Program messages are the remote commands sent to Network Master as shown in Figure 1.4.

A program message consists of one or more program message units separated with a semicolon (;). Space(s) before or after a semicolon is ignored (space has no meaning). For more information on program message units, see section 1.2.1 Program Message Unit.

When a program message is sent to Network Master, a terminator is appended after it. Network Master receives the program message by detecting the terminator. For a description of the terminator, see section 1.2.4 Program Message Terminator.

The Network Master is able to handle program messages with a maximum length of 4096 characters including the message terminator.

1.2.1 Program Message Unit

A program message unit consists of a program header and a program data, see Figure 1.5.

There must be one or more spaces between a program header and a program data. Network Master recognizes the program header and program data using the space(s). One or more spaces before a program header are ignored.
1.2.2 Program Headers

The program header specifies the function of the command message unit sent from the controller to Network Master. There are two types of program headers:

- Program headers for command message units.
- Program headers for query message units. Similar to headers for command message unit, but are always followed by a question mark “?”.

The Network Master supports some of the common commands defined in the IEEE488.2 standard. These common commands are special in the way that they are always preceded by an asterisk “*”; e.g. *IDN?. All other commands are referred to as "device specific commands". Device specific commands consists of two or more `<program mnemonic>`’s (hereinafter called "mnemonic") separated with a colon "(:)".

```
[:]<program mnemonic>[:<program mnemonic>]|...
```

A mnemonic is a character string, which consists of capital and small letters. The capital part of the mnemonic is also referred to as the short form of the mnemonic.

- Long form program header: INStrument:STARt
- Short form program header: INST:STAR

The Network Master recognizes a mnemonic even if only the short form is sent. For example, mnemonic SYSTe is recognized as a normal mnemonic when SYST is sent.

In this way, capital and small letters are used for recognizing long and short forms of a mnemonic. The Network Master does not distinguish between capital and small letters when reading the program header. However, the Network Master only accepts the short form or the complete long form of a mnemonic. Hence SYSTe is not a valid mnemonic. The following program headers are all acceptable and assumed to be the same:

- SYSTEM:POWER:SOURCE?
- system:power:source?
- SysTeM:PoWeR:SoUr?
- syst:POW:sour?
1.2 Program Messages

1.2.3 Program Data

Program data is sent following the program header as parameters specified in the command message unit. This operation manual uses the notations given below in Table 1.2 for indicating the program data format. Most of them are defined in the IEEE488.2 standard.

<table>
<thead>
<tr>
<th>Program data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;BOOLEAN PROGRAM DATA&gt;</td>
<td>Defined in IEEE488.2&lt;/br&gt;Indicates On/Off, Enable/Disable, or Yes/No. &lt;br&gt;  &lt;br&gt;To specify On/Enable state, set {ON</td>
</tr>
<tr>
<td>&lt;NUMERIC PROGRAM DATA&gt;</td>
<td>Comprises &lt;DECIMAL NUMERIC PROGRAM DATA&gt; and &lt;NON-DECIMAL NUMERIC PROGRAM DATA&gt; as defined in IEEE488.2 &lt;br&gt;The Network Master accepts both decimal and non-decimal entries for the &lt;NUMERIC PROGRAM DATA&gt;.</td>
</tr>
<tr>
<td>&lt;DECIMAL NUMERIC PROGRAM DATA&gt;</td>
<td>Defined in IEEE488.2 &lt;br&gt;Comprises &lt;NR1&gt;, &lt;NR2&gt; and &lt;NR3&gt; decimal values, where &lt;br&gt;&lt;NR1&gt; indicates an integer value. &lt;br&gt;&lt;NR2&gt; indicates a numeric value in fixed point format. &lt;br&gt;&lt;NR3&gt; indicates a numeric value in floating point format. Examples:  &lt;br&gt;&lt;NR1&gt;: 123  &lt;br&gt;&lt;NR2&gt;: -123.456  &lt;br&gt;&lt;NR3&gt;: 1.23E-3</td>
</tr>
<tr>
<td>&lt;NON-DECIMAL NUMERIC PROGRAM DATA&gt;</td>
<td>Defined in IEEE488.2 &lt;br&gt;Comprises &lt;HEXADECIMAL&gt;, &lt;OCTAL&gt; or &lt;BINARY&gt; program data. &lt;br&gt;See below for further details.</td>
</tr>
<tr>
<td>&lt;HEXADECIMAL&gt;</td>
<td>Conforms to the hexadecimal format defined in IEEE488.2 as follows:  &lt;br&gt;#[H</td>
</tr>
<tr>
<td>&lt;OCTAL&gt;</td>
<td>Conforms to the octal format defined in IEEE488.2 as follows:  &lt;br&gt;#[Q</td>
</tr>
<tr>
<td>&lt;BINARY&gt;</td>
<td>Conforms to the binary format defined in IEEE488.2 as follows:  &lt;br&gt;#[B</td>
</tr>
<tr>
<td>&lt;STRING PROGRAM DATA&gt;</td>
<td>Defined in IEEE488.2 &lt;br&gt;A character string in a pair of single quotation marks (') or double quotation marks (&quot;). Examples:  &lt;br&gt;&quot;Network Master&quot;  &lt;br&gt;'Testing the network'</td>
</tr>
</tbody>
</table>

---

continued on next page...
... continued from previous page

<table>
<thead>
<tr>
<th>Program data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;CHARACTER PROGRAM DATA&gt;</td>
<td>Defined in IEEE488.2</td>
</tr>
<tr>
<td></td>
<td>Indicates two or more mnemonics for selections. Like program header mnemonics, &lt;CHARACTER PROGRAM DATA&gt; mnemonics can have a short and a long form. The syntax used in the Network Master additionally allows a digit as the first character of a mnemonic and also allows a dash (-) inside a mnemonic.</td>
</tr>
</tbody>
</table>

Table 1.2: Acceptable program data

1.2.4 Program Message Terminator

A program message terminator indicates the end of the program message. Upon reception of a terminator, the Network Master assumes that the program message is complete and starts processing the message. A terminator must always be added to the end of a program message. For Network Master the program message terminator is:

\[ \text{\{WHITE SPACE\}\{}\{NL\}\] for Ethernet based remote control

<WHITE SPACE> is one or more ASCII characters with a value in the range of 0x00 to 0x09 or 0x0B to 0x20 (0 to 9 or 11 to 32 in decimal). These ranges include the ASCII control characters and space, except NL (newline). Since <WHITE SPACE> includes CR (0x0D) (13 in decimal), \{CR\}\{NL\} is also interpreted as a terminator by Network Master in Ethernet based remote control - to keep compatibility with conventional models.

1.2.5 Compound Program Messages

Compound headers are supported by the Network Master. Examples of the use of the compound headers are shown below.

The three program message units:

SYSTem:TIME?
SYSTem:DATE?
SYSTem:POWer:SOURce?

can be combined in one program message as follows:

SYSTem:TIME?; :SYSTem:DATE?; :SYSTem:POWer:SOURce?

or just:

SYSTem:TIME?; DATE?; POWer:SOURce?
(SYST: mnemonic can be omitted in the second and third program data units)

For further information on compound headers, see Appendix A of the IEEE488.2 standard.

1.2.6 Sequential Execution

The Network Master processes one program message unit at a time and in the same order in which they are arranged within the program message. The Network Master will not start processing a new program message until the processing of the current program message is finished.
1.3 Response Messages

Response messages are messages sent from a Network Master to a controller as reply to queries, see Figure 1.6.

Figure 1.6: Response message structure

A response message consists of one or more response data separated with a semicolon (;). The response message is terminated with the response message terminator.

1.3.1 Response Data

Response data is a data returned by Network Master as reply to a query received from the controller. Table 1.3 shows examples of the response data format used in this manual.

<table>
<thead>
<tr>
<th>Response data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;BOOLEAN RESPONSE DATA&gt;</td>
<td>Defined in SCPI-99 &lt;br&gt;Indicates On/Off, Enable/Disable, or Yes/No. &lt;br&gt;When &quot;1&quot; is returned, it indicates an On/Enable state. &lt;br&gt;When &quot;0&quot; is returned, it indicates an Off/Disable state.</td>
</tr>
<tr>
<td>&lt;NR1 NUMERIC RESPONSE DATA&gt;</td>
<td>Defined in IEEE488.2 &lt;br&gt;Indicates a decimal integer value. &lt;br&gt;Examples: &lt;br&gt;123 &lt;br&gt;-500</td>
</tr>
<tr>
<td>&lt;NR2 NUMERIC RESPONSE DATA&gt;</td>
<td>Defined in IEEE488.2 &lt;br&gt;Indicates a numeric value in fixed point format. &lt;br&gt;Examples: &lt;br&gt;123.45 &lt;br&gt;-500.0</td>
</tr>
</tbody>
</table>

continued on next page...
<table>
<thead>
<tr>
<th>Response data type</th>
<th>Description</th>
</tr>
</thead>
</table>
| `<NR3 NUMERIC RESPONSE DATA>`                  | Defined in IEEE488.2  
Indicates a numeric value in floating point format.  
Examples:  
1.23E3  
-5.67E-4  |
| `<HEXADECIMAL NUMERIC RESPONSE DATA>`         | Conforms to the hexadecimal format defined in IEEE488.2 as follows:  
#H[A|B|C|D|E|F|<digit>]*  
<digit> is an ASCII character with a value in the range of 0x30 to 0x3F (48 to 57 in decimal), a numeric 0 to 9.  
Example:  
#H0011EEFF  |
| `<BINARY NUMERIC RESPONSE DATA>`              | Conforms to the binary format defined in IEEE488.2 as follows:  
#B{0|1}...  
Example:  
#B10101010  |
| `<STRING RESPONSE DATA>`                     | Defined in IEEE488.2  
A character string enclosed in a pair of double quotation marks ("}).  
Example:  
"Network Master - Testing the network."  |
| `<CHARACTER RESPONSE DATA>`                   | Defined in IEEE488.2  
Indicates two or more mnemonics for selections. Like program header mnemonics, `<CHARACTER RESPONSE DATA>` mnemonics can have a short and a long form. The Network Master always returns the short form. The syntax used in the Network Master additionally allows a digit as the first character of a mnemonic and also allows a dash (-) inside a mnemonic.  |
| `<EXPRESSION RESPONSE DATA>`                  | Defined in IEEE488.2  
A Network Master-defined set of `<RESPONSE DATA>` elements separated by a comma (,) and enclosed by a set of parenthesis.  
Example:  
(2,0.5),(3,0.25),(4,1.75)  
For further details refer to the detailed description of the Network Master specific commands.  |
| `<DEFINITE LENGTH ARBITRARY BLOCK RESPONSE DATA>` | Defined in IEEE488.2  
This response data type is used when the instrument streams binary data (typically PDF files) to the controller. It is defined as  
#<nonzero digit><digits><8 bit data bytes>,  
where:  
<nonzero digit> is a single ASCII character in the range of '1'-9'.  
It represents the length of <digits> in number of bytes.  
<digits> is a number of ASCII characters in the range of '0'-9', which together are a decimal representation of the number of succeeding data bytes.  
Example:  
#49137<9137 bytes of binary data>  |

Table 1.3: Network Master response data
1.3.2 Response Messages Terminator

A response message terminator indicates the end of the response message. Network Master appends the terminator to the end of a response message to indicate the end of the message. For Network Master the response message terminator is \{'NL\}'.

1.3.3 Prompt

For Ethernet based remote control a prompt can optionally be returned by the Network Master when all commands in a program message has completed. The prompt is inserted after the response message if any. It can be useful to enable the prompt when manually typing commands on the command line of the remote control interface. The prompt inserted is:

```
SCPI:>
```
1.4 Status

1.4.1 IEEE488.2 Standard Status and SCPI-defined Registers

![Diagram showing IEEE488.2 standard status and SCPI-defined registers/queues. ⊕ means logical OR.]

Figure 1.7: IEEE488.2 standard status and SCPI-defined registers/queues. ⊕ means logical OR.
## Status Byte

<table>
<thead>
<tr>
<th>Bit</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Port 1</td>
<td>Summary-message bit for the Port Event Summary register. Use the <code>STATus:PRESet</code> command described in section 2.5.16 and the <code>STATus:PORT:ENABle</code> command described in section 2.5.13 to enable generation of this summary-message.</td>
</tr>
<tr>
<td>1</td>
<td>Event Queue</td>
<td>Summary-message bit for the Event Queue for the currently selected application server. Use the <code>SYSTem:ERRor[:NEXT]?</code> command described in section 2.3.2 to retrieve the messages.</td>
</tr>
<tr>
<td>2</td>
<td>Error Queue</td>
<td>Summary-message bit for the Error queue for all connected application servers. Use the <code>INSTRument:ERRor[:NEXT]?</code> command described in section 2.4.18 to retrieve the messages.</td>
</tr>
<tr>
<td>3</td>
<td>QUESTIONable</td>
<td>Summary-message bit for the Questionable Status register. Use the <code>STATus:QUESTIONable:ENABle</code> command described in section 2.5.8 to enable generation of this summary-message.</td>
</tr>
<tr>
<td>4</td>
<td>Output Queue</td>
<td>Summary-message bit for the Output Queue.</td>
</tr>
<tr>
<td>5</td>
<td>Standard Event</td>
<td>Summary-message bit for the Standard Event Status register. Use the <code>*ESE</code> command described in section 2.2.2 to enable generation of this summary-message.</td>
</tr>
<tr>
<td>6</td>
<td>Master Summary</td>
<td>The Master Summary Status message. Use the <code>*SRE</code> command described in section 2.2.7 to enable generation of this summary-message.</td>
</tr>
<tr>
<td>7</td>
<td>OPERATION</td>
<td>Summary-message bit for the Operation Status register. Use the <code>STATus:OPERation:ENABle</code> command described in section 2.5.2 to enable generation of this summary-message.</td>
</tr>
</tbody>
</table>

Table 1.4: Bits in the Status Byte register (unused bits are not listed)

For more information about the Status Byte register, see section 2.2.8 on page 35.

### Standard Event Status

All condition bits are immediately changed back to 0 after they are set. This means that the only way to check the bits is to read the Event register. For more information on what triggers the Device Dependent, Execution and Command Errors see the Error/Event Queue section on page 20.

For more information on the Standard Event Status register see section 2.2.2 on page 32.


## Chapter 1 Overview

<table>
<thead>
<tr>
<th>Bit</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Operation Complete</td>
<td>The condition bit changes to 1 when *OPC command is received.</td>
</tr>
<tr>
<td>3</td>
<td>Device Dependent Error</td>
<td>The condition bit changes to 1 when a required SW or HW options is missing or the Error/Event queue is full.</td>
</tr>
<tr>
<td>4</td>
<td>Execution Error</td>
<td>The condition bit changes to 1 when a command fail to execute properly.</td>
</tr>
<tr>
<td>5</td>
<td>Command Error</td>
<td>The condition bit changes to 1 when a unknown or errored command is received.</td>
</tr>
<tr>
<td>7</td>
<td>Power On</td>
<td>The condition bit changes to 1 when the external power supply is connected.</td>
</tr>
</tbody>
</table>

Table 1.5: Bits in the Standard Event Status register (unused bits are not listed)

### Error/Event Queue

When an unexpected error or event occurs, an entry is added to the Error/Event queue. This queue can hold 4 errors or events. If the queue overflows, the most recent events are discarded. A summary-message in bit 2 of the Status Byte is 1 when the queue is not empty. Table 1.6 gives an overview of the different errors and events inserted in the queue.

For more information about the Event queue, see section 2.4.18 on page 50.

<table>
<thead>
<tr>
<th>Event Number</th>
<th>Error Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Command errors <em>(Command Error bit is simultaneously set)</em></td>
</tr>
<tr>
<td>-100</td>
<td>Command error</td>
</tr>
<tr>
<td>-102</td>
<td>Syntax error</td>
</tr>
<tr>
<td>-104</td>
<td>Data type error</td>
</tr>
<tr>
<td>-115</td>
<td>Unexpected number of parameters</td>
</tr>
<tr>
<td>-130</td>
<td>Suffix error</td>
</tr>
<tr>
<td>-131</td>
<td>Invalid suffix</td>
</tr>
<tr>
<td>-138</td>
<td>Suffix not allowed</td>
</tr>
<tr>
<td>-200</td>
<td>Execution errors <em>(Execution Error bit is simultaneously set)</em></td>
</tr>
<tr>
<td>-220</td>
<td>Parameter error</td>
</tr>
<tr>
<td>-221</td>
<td>Settings conflict</td>
</tr>
<tr>
<td>-222</td>
<td>Data out of range</td>
</tr>
<tr>
<td>-224</td>
<td>Illegal parameter value</td>
</tr>
<tr>
<td>-250</td>
<td>Mass storage error</td>
</tr>
<tr>
<td>1</td>
<td>Option Missing</td>
</tr>
<tr>
<td>-350</td>
<td>Queue overflow</td>
</tr>
</tbody>
</table>

Table 1.6: Errors and events that can occur in the Error/Event queue

For more information about the Error queue, see section 2.3.2 on page 36.
Questionable Status

<table>
<thead>
<tr>
<th>Bit</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td><em>No bits in this register are currently in use.</em>)</td>
</tr>
</tbody>
</table>

Table 1.7: Bit in the Questionable Status register (unused bits are not listed)

For more information about the Questionable Status register, see section 2.5.6 on page 54.

Operation Status

<table>
<thead>
<tr>
<th>Bit</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>Measuring</td>
<td>The measuring condition bit changes to 1 when the an Application Server is running a measurement or a test. It returns to 0 when the measurement or test is stopped.</td>
</tr>
</tbody>
</table>

Table 1.8: Bit in the Operation Status register (unused bits are not listed)

For more information about the Operation Status register, see section 2.5.1 on page 53.
1.4.2 Network Master Unique Status Registers

Figure 1.8 shows the structure of the Network Master Unique Status registers.

![Diagram of Network Master Unique Status registers]

**Figure 1.8:** The Network Master Unique Status registers for some of the supported interfaces.

Similar registers exist for T1, OTN, Physical, and T3 interfaces.
The Network Master Unique Status registers are used to report alarms and errors for all interfaces. Each interface has one or more registers to represent the current alarm and error status. Each of these Alarm and Error registers are summarized in a General Interface Summary register (AESummary), see Figure 1.9. The exact layout of each register is found under the Status section for each interface. There are two Port Status registers, one for each port on the Network Master. The Port Status registers summarize the AESummary registers from the active interfaces. The Port Status registers are again summarized in bit 0 and 1 of the Status Byte, see Figure 1.10.
All Network Master Unique Status registers follow the register model defined in section 11.4.2 of IEEE488.2. The register model is shown in Figure 1.11.

**Condition Registers**
The Condition registers reflect the real-time status of the instrument or summary-message bits of other status registers.

**Transition Filters**
The transition filters for the Network Master’s Unique Status registers are locked to the "Positive transition" criteria. This means that events will be generated when the condition bits goes from 0 to 1. It is not possible to change the transition criteria.

**Event Registers**
The Event registers store the transition filter output. These registers are cleared when read.

**Event Enable Registers**
The Event Enable registers cannot be read or written and on power-on they are all set to zero. To enable the Event Enable registers and hence generation of summary-messages, the STATus:PRESet command must be send. The STATus:PRESet command changes all the bits in the Event Enable registers to 1.

**Summary Register**
The summarized status of the following registers is stored in the Summary Register (Refer to Table 1.9).

- Alarm event registers
- Error event registers
- Event registers of lower layer summary registers

Like other status registers, the Summary register consists of the Condition register and the Event register, according to the register model defined in section 11.4.2 of IEEE488.2. The Summary register is locked to positive transition criteria because it consists of lower layer Event registers.

Figure 1.11: The register model for the Network Master Unique Status registers.
1.4 Status

1.4.3 Reading, Writing and Clearing Status Registers

The following two tables list the possibilities for reading and writing the various status registers and queues. They also show when and how registers are cleared or enabled.

<table>
<thead>
<tr>
<th>Registers</th>
<th>Reading</th>
<th>Writing</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>IEEE488.2 standard status registers</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Status Byte</td>
<td>*STB?</td>
<td>Not possible</td>
</tr>
<tr>
<td>Service Request Enable</td>
<td>*SRE?</td>
<td>*SRE</td>
</tr>
<tr>
<td>Standard Event Status Enable</td>
<td>*ESR?</td>
<td>Not possible</td>
</tr>
<tr>
<td></td>
<td>After reading, the register content is cleared.</td>
<td></td>
</tr>
<tr>
<td>Standard Event Status Enable</td>
<td>*ESE?</td>
<td>*ESE</td>
</tr>
<tr>
<td><strong>SCPI defined status registers</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Error/Event Queue</td>
<td>SYST:ERR?</td>
<td>Not possible</td>
</tr>
<tr>
<td></td>
<td>After reading, the error/event is removed from the queue.</td>
<td></td>
</tr>
<tr>
<td>Operation Event Enable</td>
<td>STAT:OPER?</td>
<td>Not possible</td>
</tr>
<tr>
<td></td>
<td>After reading, the register content is cleared.</td>
<td></td>
</tr>
<tr>
<td>Questionable Event Enable</td>
<td>STAT:QUES?</td>
<td>Not possible</td>
</tr>
<tr>
<td></td>
<td>After reading, the register content is cleared.</td>
<td></td>
</tr>
<tr>
<td><strong>Network Master unique status registers</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Condition</td>
<td>&lt;Interface&gt;:STAT:&lt;Port&gt;:&lt;Register&gt;:COND?</td>
<td>Not possible</td>
</tr>
<tr>
<td>Transition Filter</td>
<td>Not possible</td>
<td>Not possible</td>
</tr>
<tr>
<td>Event</td>
<td>&lt;Interface&gt;:STAT:&lt;Port&gt;:&lt;Register&gt;?</td>
<td>Not possible</td>
</tr>
<tr>
<td></td>
<td>After reading, the register content is cleared.</td>
<td></td>
</tr>
<tr>
<td>Enable</td>
<td>Not possible</td>
<td>Not possible</td>
</tr>
</tbody>
</table>

Table 1.9: Reading and writing of Status registers
### Registers

<table>
<thead>
<tr>
<th>Registers</th>
<th><em>RST</em></th>
<th><em>CLS</em></th>
<th>PowerOn</th>
<th>STAT:PRES</th>
</tr>
</thead>
<tbody>
<tr>
<td>IEEE488.2 standard status registers</td>
<td>Status Byte</td>
<td>No Changes</td>
<td>Cleared</td>
<td>Cleared</td>
</tr>
<tr>
<td>Service Request Enable</td>
<td>No Changes</td>
<td>No Changes</td>
<td>Cleared</td>
<td>No Changes</td>
</tr>
<tr>
<td>Standard Event Status</td>
<td>No Changes</td>
<td>Cleared</td>
<td>Cleared</td>
<td>No Changes</td>
</tr>
<tr>
<td>Standard Event Status Enable</td>
<td>No Changes</td>
<td>No Changes</td>
<td>Cleared</td>
<td>No Changes</td>
</tr>
<tr>
<td>SCPI defined status registers</td>
<td>Error/Event Queue</td>
<td>No Changes</td>
<td>Cleared</td>
<td>Cleared</td>
</tr>
<tr>
<td>Operation Event</td>
<td>No Changes</td>
<td>Cleared</td>
<td>Cleared</td>
<td>No Changes</td>
</tr>
<tr>
<td>Operation Enable</td>
<td>No Changes</td>
<td>No Changes</td>
<td>Cleared</td>
<td>No Changes</td>
</tr>
<tr>
<td>Questionable Event</td>
<td>No Changes</td>
<td>Cleared</td>
<td>Cleared</td>
<td>No Changes</td>
</tr>
<tr>
<td>Questionable Enable</td>
<td>No Changes</td>
<td>No Changes</td>
<td>Cleared</td>
<td>No Changes</td>
</tr>
<tr>
<td>Network Master Unique Status registers</td>
<td>Condition</td>
<td>No Changes</td>
<td>No Changes</td>
<td>Cleared</td>
</tr>
<tr>
<td>Transition Filter</td>
<td>No Changes</td>
<td>No Changes</td>
<td>No Changes</td>
<td>No Changes</td>
</tr>
<tr>
<td>Event</td>
<td>No Changes</td>
<td>Cleared</td>
<td>Cleared</td>
<td>No Changes</td>
</tr>
<tr>
<td>Enable</td>
<td>No Changes</td>
<td>No Changes</td>
<td>Cleared</td>
<td>Enabled (all 1’s)</td>
</tr>
</tbody>
</table>

Table 1.10: Status registers behaviour for different commands/events

**Notes**

The Condition resister of the Summary register is locked to positive transition criteria. Therefore, if clearing the register (*CLS*) while an alarm or error is occurring, the register bits stay in "0" (cleared) in spite of the alarm or error occurrence.
1.5 Controller Example

One example of how to connect a controller to the Network Master instrument is described in this section.

1.5.1 PuTTY

PuTTY is a free Telnet/SSH client which supports raw TCP connections. With PuTTY it is possible to get terminal emulation access to the instrument. It is recommended to enable the prompt when using PuTTY. PuTTY does not support file streaming like the MEAS:EXP command. PuTTY can be downloaded from http://www.chiark.greenend.org.uk/~sgtatham/putty/

Setup

1. Install PuTTY.
2. Start PuTTY.
3. In the PuTTY Configuration enable Implicit CR in every LF at Category: → Terminal.

![PuTTY Configuration](image-url)

Figure 1.12: Enable Implicit CR in every LF in PuTTY
Chapter 1 Overview

4. In the instrument GUI, find the instrument’s **IP Address** information, see Figure 1.13. Then type it in PuTTY, see Figure 1.14.

![Image of instrument GUI showing IP Address information]

Figure 1.13: **IP Address** on the instrument

5. In PuTTY, type 56001 in the Port field, select the Raw radio button in the Connection type field, and click the Open button. see Figure 1.14.

![Image of PuTTY configuration showing destination and Open button]

Figure 1.14: Specify the destination and Open the connection in PuTTY
6. A window appears, see Figure 1.15.

![Connection established with PuTTY](image)

Figure 1.15: Connection established with PuTTY
1.6 Definitions

1.6.1 NaN (Not a Number)
NaN is defined in SCPI-99. NaN is represented as 9.91E37 (NR3 NUMERIC RESPONSE DATA) as defined in IEEE 754. NaN is also used to represent missing data.

1.6.2 → Right Arrow
The right arrow → used in this document has two meanings:
- On the left side of the arrow is a query and returned value on the right hand side.
  Example: TMBP:RX1:PATT? → PRBS11

1.6.3 Data Bit (DB)
Data bit is represented as DBx where x represents the bit index in a register. DB1 is always LSB.

![Figure 1.16: Data bit](image)

1.6.4 Port Number (Logical Port)
Specify a logical port number assigned for each started application as a port number <Pt> in the SCPI command. The logical port numbers will be assigned in the order of Module1-Port1, Module1-Port2, Module2-Port1, and Module2-Port2. For OTDR module, specify the physical port number to “1”.

Example of the OTDR application start command
- INST:STAR OTDR-OTDR,1-PORT1 (Figure 1.17)
  Physical Port : Logical Port
  1-PORT1 : PORT1

![Figure 1.17: The physical and logical port number of the OTDR module](image)

Notes
- To switch between the Single Mode port (SM) and the Multi Mode port (MM) of the MU100021A, refer to section 3.1.1 on page 63.
- For MU100023A, specify PORT1 as the physical port. When the wavelength is set to 1310 nm or 1550 nm by the command in section 3.1.4 on page 64, PORT1 is selected as the measurement port. When set to 1650 nm, PORT2 is selected as the measurement port.
Chapter 2

SCPI Conformance Information

2.1 SCPI Version

The Network Master Remote Control application conforms to SCPI 1999.0
2.2 IEEE 488.2 Mandatory Commands

2.2.1 *CLS

<table>
<thead>
<tr>
<th>Syntax</th>
<th>*CLS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>This command clears all the event registers summarized in the Status Byte register. The error queue is emptied. Neither the Standard Event Status Enable register, nor the Service Request Enable register are affected by this command.</td>
</tr>
<tr>
<td>Parameter</td>
<td>None.</td>
</tr>
<tr>
<td>Response</td>
<td>None.</td>
</tr>
<tr>
<td>Example</td>
<td>*CLS</td>
</tr>
<tr>
<td>Note</td>
<td>All active (SCPI) sessions has their own set of standard registers.</td>
</tr>
</tbody>
</table>

2.2.2 *ESE

<table>
<thead>
<tr>
<th>Syntax</th>
<th>*ESE &lt;mask&gt;</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>This command 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. This register is cleared at power-on. The *RST and *CLS commands do not affect this register.</td>
</tr>
<tr>
<td>Parameter</td>
<td>&lt;mask&gt; = &lt;NUMERIC PROGRAM DATA&gt;</td>
</tr>
<tr>
<td></td>
<td>The bits and their values for the enable mask:</td>
</tr>
<tr>
<td></td>
<td>DB1 (1) = Operation Complete</td>
</tr>
<tr>
<td></td>
<td>DB2 = NOT USED</td>
</tr>
<tr>
<td></td>
<td>DB3 = NOT USED</td>
</tr>
<tr>
<td></td>
<td>DB4 (8) = Device Dependent Error</td>
</tr>
<tr>
<td></td>
<td>DB5 (16) = Execution Error</td>
</tr>
<tr>
<td></td>
<td>DB6 (32) = Command Error</td>
</tr>
<tr>
<td></td>
<td>DB7 = NOT USED</td>
</tr>
<tr>
<td></td>
<td>DB8 (128) = Power On</td>
</tr>
<tr>
<td></td>
<td>MINimum=0, MAXimum=255</td>
</tr>
<tr>
<td>Response</td>
<td>None.</td>
</tr>
<tr>
<td>Example</td>
<td>*ESE 16</td>
</tr>
<tr>
<td>Note</td>
<td>All active sessions has their own Standard Event Status Enable register.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Syntax</th>
<th>*ESE?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>This query returns the contents of the Standard Event Status Enable register.</td>
</tr>
<tr>
<td>Parameter</td>
<td>None.</td>
</tr>
<tr>
<td>Response</td>
<td>&lt;mask&gt; = &lt;NR1 NUMERIC RESPONSE DATA&gt;</td>
</tr>
<tr>
<td></td>
<td>See the *ESE command for bit values for the enable mask.</td>
</tr>
<tr>
<td>Example</td>
<td>*ESE? → 15</td>
</tr>
<tr>
<td>Note</td>
<td></td>
</tr>
</tbody>
</table>
2.2 IEEE 488.2 Mandatory Commands

### 2.2.3 *ESR?*

**Syntax**

*ESR?

**Description**

This query returns the contents of the Standard Event Status register. This register is cleared after being read.

**Parameter**

None.

**Response**

<value> = <NR1 NUMERIC RESPONSE DATA>

The bits and their values for the register:
- DB1 (1) = Operation Complete
- DB2 = NOT USED
- DB3 = NOT USED
- DB4 (8) = Device Dependent Error
- DB5 (16) = Execution Error
- DB6 (32) = Command Error
- DB7 = NOT USED
- DB8 (128) = Power On

**Example**

*ESR? → 49

**Note**

All active sessions has their own Standard Event Status register.

### 2.2.4 *IDN?*

**Syntax**

*IDN?

**Description**

This query returns the instrument identification over the interface.

**Parameter**

None.

**Response**

<manufacturer>,<model>,<serial>,<version> =

<ARBITRARY ASCII RESPONSE DATA>

**Example**

*IDN? → Anritsu,MT1000A,6123456789,1.00

**Note**

### 2.2.5 *OPC*

**Syntax**

*OPC

**Description**

This command causes the instrument to generate the operation complete message in the Standard Event Status register when all pending selected instrument operations have been finished.

**Parameter**

None.

**Response**

None.

**Example**

*OPC

**Note**

All active sessions has their own Standard Event Status register.

**Syntax**

*OPC?

**Description**

This query places the ASCII character '1' into the instrument’s output queue when all pending operations have been finished.

**Parameter**

None.

**Response**

<operation complete> = <NR1 NUMERIC RESPONSE DATA>

**Example**

*OPC? → 1

**Note**

Only application servers connected to by the current session are synchronized.
2.2.6  *RST

<table>
<thead>
<tr>
<th>Syntax</th>
<th>*RST</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Description</strong></td>
<td>This command sets the instrument to reset setting (standard setting) stored in internal storage. The instrument is placed in the idle state awaiting a command. All running application/servers are closed when the *RST command is issued. The following are not changed: - Service Request Enable register (SRE) - Standard Event Status register (ESR) - Standard Event Status Enable register (ESE) - Any instrument specific Status Event or Status Event Enable registers</td>
</tr>
<tr>
<td>Parameter</td>
<td>None.</td>
</tr>
<tr>
<td>Response</td>
<td>None.</td>
</tr>
<tr>
<td>Example</td>
<td>*RST</td>
</tr>
<tr>
<td><strong>Note</strong></td>
<td>*RST is specially configured to be compatible with SCPI remote control.</td>
</tr>
</tbody>
</table>

2.2.7  *SRE

<table>
<thead>
<tr>
<th>Syntax</th>
<th>*SRE &lt;enable mask&gt;</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Description</strong></td>
<td>This command sets bits in the Service Request Enable register. A 1 in a bit in the enable register enables the corresponding bit in the Status Byte, also sets the Master Summary Status bit (DB7) in the Status Byte. The register is cleared at power-on. The *RST and *CLS commands do not affect the register.</td>
</tr>
<tr>
<td>Parameter</td>
<td>&lt;enable mask&gt; = &lt;NUMERIC PROGRAM DATA&gt; The bits and their values for the register: DB1 (1) = Port Event Summary DB2 (2) = Event Queue Summary for the currently selected application server. DB3 (4) = Error Queue Summary for all connected application servers. DB4 (8) = Questionable Status Summary DB5 (16) = Message Available (MAV) DB6 (32) = Standard Event Status Summary (ESB) DB7 = NOT USED DB8 (128) = Operation Status Summary MINimum=0, MAXimum=255</td>
</tr>
<tr>
<td>Response</td>
<td>None.</td>
</tr>
<tr>
<td>Example</td>
<td>*SRE 255</td>
</tr>
<tr>
<td><strong>Note</strong></td>
<td>All active sessions has their own Service Request Enable register.</td>
</tr>
</tbody>
</table>

**Syntax**  *SRE?**

| Description | This query returns the contents of the Service Request Enable register. |
| Parameter | None. |
| **Response** | <mask> = <NR1 NUMERIC RESPONSE DATA> See the *SRE command for bit values for the enable mask. |
| **Example** | *SRE? → 255 |
| **Note** | 34 |
2.2 IEEE 488.2 Mandatory Commands

2.2.8 *STB?

<table>
<thead>
<tr>
<th>Syntax</th>
<th>*STB?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>This query returns the contents of the Status Byte register. The Master Summary Status (MSS) bit is true when any bit of the STB register is set and a matching bit in the Service Request Enable Register is set, see *SRE. The Status Byte register including the MSS is not altered by this query.</td>
</tr>
<tr>
<td>Parameter</td>
<td>None.</td>
</tr>
<tr>
<td>Response</td>
<td>&lt;value&gt; = &lt;NR1 NUMERIC RESPONSE DATA&gt;</td>
</tr>
<tr>
<td>The bits and their values for the register:</td>
<td></td>
</tr>
<tr>
<td>DB1 (1) = Port Event Summary</td>
<td></td>
</tr>
<tr>
<td>DB2 (2) = Event Queue Summary for the currently selected application server.</td>
<td></td>
</tr>
<tr>
<td>DB3 (4) = Error Queue Summary for all connected application servers.</td>
<td></td>
</tr>
<tr>
<td>DB4 (8) = Questionable Status Summary</td>
<td></td>
</tr>
<tr>
<td>DB5 (16) = Message Available (MAV)</td>
<td></td>
</tr>
<tr>
<td>DB6 (32) = Standard Event Status Summary (ESB)</td>
<td></td>
</tr>
<tr>
<td>DB7 (64) = Master Summary Status (MSS)</td>
<td></td>
</tr>
<tr>
<td>DB8 (128) = Operation Status Summary</td>
<td></td>
</tr>
<tr>
<td>Example</td>
<td>*STB? → 7</td>
</tr>
</tbody>
</table>

2.2.9 *TST?

<table>
<thead>
<tr>
<th>Syntax</th>
<th>*TST?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>This query returns whether or not the instrument completed the self-test without any detected errors.</td>
</tr>
<tr>
<td>Parameter</td>
<td>None.</td>
</tr>
<tr>
<td>Response</td>
<td>&lt;result&gt; = &lt;NR1 NUMERIC RESPONSE DATA&gt;</td>
</tr>
<tr>
<td>0: No self-test errors detected</td>
<td></td>
</tr>
<tr>
<td>1: Self-test error detected</td>
<td></td>
</tr>
<tr>
<td>Example</td>
<td>*TST? → 0</td>
</tr>
</tbody>
</table>

Note | Self-test is performed automatically at the time of power up. |

2.2.10 *WAI

<table>
<thead>
<tr>
<th>Syntax</th>
<th>*WAI</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>This command prevents the instrument from executing any further commands until the current command has been finished. All pending operations are completed during the wait period.</td>
</tr>
<tr>
<td>Parameter</td>
<td>None.</td>
</tr>
<tr>
<td>Response</td>
<td>None.</td>
</tr>
<tr>
<td>Example</td>
<td>*WAI</td>
</tr>
</tbody>
</table>

Note | *WAI functions for commands called "overlap command". For now Network Master does not have any overlap commands. So *WAI does not work on the current Network Master. |
### 2.3 SCPI System Subsystem Commands

#### 2.3.1 SYSTem:VERSion?

<table>
<thead>
<tr>
<th><strong>Syntax</strong></th>
<th>SYSTem:VERSion?</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Description</strong></td>
<td>This query returns the SCPI revision to which the system complies.</td>
</tr>
<tr>
<td><strong>Parameter</strong></td>
<td>None.</td>
</tr>
<tr>
<td><strong>Response</strong></td>
<td>(&lt;version&gt; = &lt;NR2 NUMERIC RESPONSE DATA&gt;)</td>
</tr>
<tr>
<td><strong>Example</strong></td>
<td>SYST:VERS? → 1999.0</td>
</tr>
<tr>
<td><strong>Note</strong></td>
<td></td>
</tr>
</tbody>
</table>

#### 2.3.2 SYSTem:ERRor[:NEXT]?

<table>
<thead>
<tr>
<th><strong>Syntax</strong></th>
<th>SYSTem:ERRor[:NEXT]?</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Description</strong></td>
<td>This query returns the oldest entry of the error queue and removes the returned entry from the queue.</td>
</tr>
<tr>
<td><strong>Parameter</strong></td>
<td>None.</td>
</tr>
<tr>
<td><strong>Response</strong></td>
<td>(&lt;error number&gt; = &lt;NR1 NUMERIC RESPONSE DATA&gt; &lt;description&gt; = &lt;STRING RESPONSE DATA&gt;)</td>
</tr>
<tr>
<td><strong>Example</strong></td>
<td>SYST:ERR? → -222,&quot;Data out of range&quot;</td>
</tr>
<tr>
<td><strong>Note</strong></td>
<td>All active sessions has their own error queue. Application server ID is added to each error message when the additional message is selected TEST or BOTH. Application server ID is -1 for system errors. Application server ID is fixed to 0 if no error message is in error queue. Error command is added to each error message when the additional message is selected COMManD or BOTH.</td>
</tr>
</tbody>
</table>

#### 2.3.3 SYSTem:ERRor:ADDitonal[:MESSage]

<table>
<thead>
<tr>
<th><strong>Syntax</strong></th>
<th>SYSTem:ERRor:ADDitonal[:MESSage] &lt;message&gt;</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Description</strong></td>
<td>This command select additional message in the error message.</td>
</tr>
<tr>
<td><strong>Parameter</strong></td>
<td>(&lt;message&gt; = &lt;CHARACTER PROGRAM DATA&gt; NONE TEST COMManD BOTH)</td>
</tr>
<tr>
<td><strong>Response</strong></td>
<td>None.</td>
</tr>
<tr>
<td><strong>Example</strong></td>
<td>SYST:ERR:ADD BOTH SYST:ERR? → -115,&quot;Unexpected number of parameters:-1:INST:TERM&quot;</td>
</tr>
<tr>
<td><strong>Note</strong></td>
<td>This setting is applied only for the current session and defaulted to NONE when session closed. See also SYSTem:ERRor[:NEXT]?</td>
</tr>
</tbody>
</table>

| **Syntax**      | SYSTem:ERRor:ADDitonal[:MESSage]?
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Description</strong></td>
<td>This query returns additional message in the error message.</td>
</tr>
<tr>
<td><strong>Parameter</strong></td>
<td>None.</td>
</tr>
<tr>
<td><strong>Response</strong></td>
<td>(&lt;message&gt; = &lt;CHARACTER RESPONSE DATA&gt;)</td>
</tr>
<tr>
<td><strong>Example</strong></td>
<td>SYST:ERR:ADD? → NON</td>
</tr>
<tr>
<td><strong>Note</strong></td>
<td></td>
</tr>
</tbody>
</table>
### 2.3.4 SYSTem:DATE

<table>
<thead>
<tr>
<th><strong>Syntax</strong></th>
<th>SYSTem:DATE &lt;year&gt;,&lt;month&gt;,&lt;day&gt;</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Description</strong></td>
<td>This command sets the date of the internal calendar.</td>
</tr>
<tr>
<td><strong>Parameters</strong></td>
<td>&lt;year&gt; = &lt;NUMERIC PROGRAM DATA&gt; &lt;br&gt; Minimum = 1997, Maximum = 2036 &lt;br&gt; &lt;month&gt; = &lt;NUMERIC PROGRAM DATA&gt; &lt;br&gt; Minimum = 1, Maximum = 12 &lt;br&gt; &lt;day&gt; = &lt;NUMERIC PROGRAM DATA&gt; &lt;br&gt; Minimum = 1, Maximum = 31</td>
</tr>
<tr>
<td><strong>Response</strong></td>
<td>None.</td>
</tr>
<tr>
<td><strong>Example</strong></td>
<td>SYST:DATE 2009,12,31</td>
</tr>
<tr>
<td><strong>Note</strong></td>
<td></td>
</tr>
</tbody>
</table>
### 2.3.7 SYSTem:GPS:NSATellites?

<table>
<thead>
<tr>
<th>Syntax</th>
<th>SYSTem:GPS:NSATellites?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>This query returns the number of satellites found by GPS.</td>
</tr>
<tr>
<td>Parameter</td>
<td>None.</td>
</tr>
<tr>
<td>Response</td>
<td>&lt;count&gt; = &lt;NR1 NUMERIC RESPONSE DATA&gt;</td>
</tr>
<tr>
<td>Example</td>
<td>SYST:GPS:NSAT? → 5</td>
</tr>
<tr>
<td>Note</td>
<td>Return &quot;0&quot; if GPS is not available.</td>
</tr>
</tbody>
</table>

### 2.3.8 SYSTem:GPS:TIME?

<table>
<thead>
<tr>
<th>Syntax</th>
<th>SYSTem:GPS:TIME?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>This query returns the GPS time.</td>
</tr>
<tr>
<td>Parameter</td>
<td>None.</td>
</tr>
<tr>
<td>Response</td>
<td>&lt;time&gt; = &lt;CHARACTER RESPONSE DATA&gt;</td>
</tr>
<tr>
<td>Note</td>
<td>Return &quot;0&quot; if GPS is not available.</td>
</tr>
</tbody>
</table>

### 2.3.9 SYSTem:GPS:LOCation?

<table>
<thead>
<tr>
<th>Syntax</th>
<th>SYSTem:GPS:LOCation?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>This query returns the location.</td>
</tr>
<tr>
<td>Parameter</td>
<td>None.</td>
</tr>
<tr>
<td>Response</td>
<td>&lt;location&gt; = &lt;NR2 NUMERIC RESPONSE DATA&gt;</td>
</tr>
<tr>
<td>Note</td>
<td>Return &quot;0&quot; if GPS is not available.</td>
</tr>
</tbody>
</table>

### 2.3.10 SYSTem:GPS:POWEr

<table>
<thead>
<tr>
<th>Syntax</th>
<th>SYSTem:GPS:POWEr &lt;power&gt;</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>This command turns GPS module’s power ON/OFF (1 - ON, 0 - OFF).</td>
</tr>
<tr>
<td>Parameters</td>
<td>&lt;power&gt; = &lt;NUMERIC PROGRAM DATA&gt;</td>
</tr>
<tr>
<td>MINimum</td>
<td>0, MAXimum = 1</td>
</tr>
<tr>
<td>Response</td>
<td>None.</td>
</tr>
<tr>
<td>Example</td>
<td>SYST:GPS:POWE 0</td>
</tr>
<tr>
<td>Note</td>
<td></td>
</tr>
</tbody>
</table>

### 2.3.11 SYSTem:GPS:POWEr?

<table>
<thead>
<tr>
<th>Syntax</th>
<th>SYSTem:GPS:POWEr?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>This query returns GPS module’s power ON/OFF (1 - ON, 0 - OFF).</td>
</tr>
<tr>
<td>Parameter</td>
<td>None.</td>
</tr>
<tr>
<td>Response</td>
<td>&lt;power&gt; = &lt;NR1 NUMERIC RESPONSE DATA&gt;</td>
</tr>
<tr>
<td>Example</td>
<td>SYST:GPS:POWE? → 0</td>
</tr>
<tr>
<td>Note</td>
<td></td>
</tr>
</tbody>
</table>

### 2.3.12 SYSTem:GPS:SYNCmode

<table>
<thead>
<tr>
<th>Syntax</th>
<th>SYSTem:GPS:SYNCmode &lt;mode&gt;</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>This command turns GPS module’s sync mode to GPS/EXT (1 - GPS, 0 - EXT).</td>
</tr>
<tr>
<td>Parameters</td>
<td>&lt;mode&gt; = &lt;NUMERIC PROGRAM DATA&gt;</td>
</tr>
<tr>
<td>MINimum</td>
<td>0, MAXimum = 1</td>
</tr>
<tr>
<td>Response</td>
<td>None.</td>
</tr>
<tr>
<td>Example</td>
<td>SYST:GPS:SYNC 0</td>
</tr>
<tr>
<td>Note</td>
<td></td>
</tr>
</tbody>
</table>
### 2.3.13 SYSTem:GPS:SYNCmode?

<table>
<thead>
<tr>
<th>Syntax</th>
<th>SYSTem:GPS:SYNCmode?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>This query returns GPS module’s sync mode GPS/EXT (1 - GPS, 0 - EXT).</td>
</tr>
<tr>
<td>Parameter</td>
<td>None.</td>
</tr>
<tr>
<td>Response</td>
<td>&lt;mode&gt; = &lt;NR1 NUMERIC RESPONSE DATA&gt;</td>
</tr>
<tr>
<td>Example</td>
<td>SYST:GPS:SYNC? → 0</td>
</tr>
<tr>
<td>Note</td>
<td></td>
</tr>
</tbody>
</table>

### 2.3.14 SYSTem:GPS:ANTNpower

<table>
<thead>
<tr>
<th>Syntax</th>
<th>SYSTem:GPS:ANTNpower &lt;power&gt;</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>This command turns GPS module’s antenna power 5V/3.3V (1 - 5V, 0 - 3.3V).</td>
</tr>
<tr>
<td>Parameters</td>
<td>&lt;power&gt; = &lt;NUMERIC PROGRAM DATA&gt;</td>
</tr>
<tr>
<td></td>
<td>MINimum = 0, MAXimum = 1</td>
</tr>
<tr>
<td>Response</td>
<td>None.</td>
</tr>
<tr>
<td>Example</td>
<td>SYST:GPS:ANTN 0</td>
</tr>
<tr>
<td>Note</td>
<td></td>
</tr>
</tbody>
</table>

### 2.3.15 SYSTem:GPS:ANTNpower?

<table>
<thead>
<tr>
<th>Syntax</th>
<th>SYSTem:GPS:ANTNpower?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>This query returns GPS module’s antenna power 5V/3.3V (1 - 5V, 0 - 3.3V).</td>
</tr>
<tr>
<td>Parameter</td>
<td>None.</td>
</tr>
<tr>
<td>Response</td>
<td>&lt;power&gt; = &lt;NR1 NUMERIC RESPONSE DATA&gt;</td>
</tr>
<tr>
<td>Example</td>
<td>SYST:GPS:ANTN? → 0</td>
</tr>
<tr>
<td>Note</td>
<td></td>
</tr>
</tbody>
</table>

### 2.3.16 SYSTem:GPS:PWRElapsed?

<table>
<thead>
<tr>
<th>Syntax</th>
<th>SYSTem:GPS:PWRElapsed?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>This query returns GPS module’s power on elapsed seconds.</td>
</tr>
<tr>
<td>Parameter</td>
<td>None.</td>
</tr>
<tr>
<td>Response</td>
<td>&lt;time&gt; = &lt;NR1 NUMERIC RESPONSE DATA&gt;</td>
</tr>
<tr>
<td>Example</td>
<td>SYST:GPS:PWRE? → 0</td>
</tr>
<tr>
<td>Note</td>
<td></td>
</tr>
</tbody>
</table>

### 2.3.17 SYSTem:GPS:LOCElapsed?

<table>
<thead>
<tr>
<th>Syntax</th>
<th>SYSTem:GPS:LOCElapsed?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>This query returns GPS module’s oscillator lock elapsed seconds.</td>
</tr>
<tr>
<td>Parameter</td>
<td>None.</td>
</tr>
<tr>
<td>Response</td>
<td>&lt;time&gt; = &lt;NR1 NUMERIC RESPONSE DATA&gt;</td>
</tr>
<tr>
<td>Example</td>
<td>SYST:GPS:LOCE? → 0</td>
</tr>
<tr>
<td>Note</td>
<td></td>
</tr>
</tbody>
</table>

### 2.3.18 SYSTem:GPS:SYNElapsed?

<table>
<thead>
<tr>
<th>Syntax</th>
<th>SYSTem:GPS:SYNElapsed?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>This query returns GPS module’s sync mode elapsed seconds.</td>
</tr>
<tr>
<td>Parameter</td>
<td>None.</td>
</tr>
<tr>
<td>Response</td>
<td>&lt;time&gt; = &lt;NR1 NUMERIC RESPONSE DATA&gt;</td>
</tr>
<tr>
<td>Example</td>
<td>SYST:GPS:SYNE? → 0</td>
</tr>
<tr>
<td>Note</td>
<td></td>
</tr>
</tbody>
</table>
### 2.3.19 SYSTem:COMMunicate:TERMinator

<table>
<thead>
<tr>
<th>Syntax</th>
<th>SYSTem:COMMunicate:TERMinator &lt;terminator&gt;</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>This command sets the terminator code which is appended to the query response.</td>
</tr>
<tr>
<td>Parameter</td>
<td>&lt;terminator&gt; = &lt;CHARACTER PROGRAM DATA&gt;</td>
</tr>
<tr>
<td></td>
<td>NONE(only GPIB)</td>
</tr>
<tr>
<td></td>
<td>LF</td>
</tr>
<tr>
<td></td>
<td>CRLF</td>
</tr>
<tr>
<td>Response</td>
<td>None.</td>
</tr>
<tr>
<td>Example</td>
<td>SYST:COMM:TERM LF</td>
</tr>
<tr>
<td>Note</td>
<td>This setting is applied only for the current session and defaulted to CRLF when session closed.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Syntax</th>
<th>SYSTem:COMMunicate:TERMinator?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>This query returns the terminator code which is appended to the query response.</td>
</tr>
<tr>
<td>Parameter</td>
<td>None.</td>
</tr>
<tr>
<td>Response</td>
<td>&lt;terminator&gt; = &lt;CHARACTER RESPONSE DATA&gt;</td>
</tr>
<tr>
<td>Example</td>
<td>SYST:COMM:TERM? → LF</td>
</tr>
<tr>
<td>Note</td>
<td></td>
</tr>
</tbody>
</table>

### 2.3.20 SYSTem:PROMpt

<table>
<thead>
<tr>
<th>Syntax</th>
<th>SYSTem:PROMpt &lt;enable&gt;</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>This command enables/disables appending of prompt to all replies from Remote Control interface.</td>
</tr>
<tr>
<td>Parameter</td>
<td>&lt;enable&gt; = &lt;BOOLEAN PROGRAM DATA&gt;</td>
</tr>
<tr>
<td>Response</td>
<td>None.</td>
</tr>
<tr>
<td>Example</td>
<td>SYST:PROM 1</td>
</tr>
<tr>
<td>Note</td>
<td>The prompt string is &quot;SCPI: &quot;</td>
</tr>
<tr>
<td></td>
<td>This setting is applied only for the current session and forgets when session closed.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Syntax</th>
<th>SYSTem:PROMpt?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>This query returns status of the prompt.</td>
</tr>
<tr>
<td>Parameter</td>
<td>None.</td>
</tr>
<tr>
<td>Response</td>
<td>&lt;enable&gt; = &lt;BOOLEAN PROGRAM DATA&gt;</td>
</tr>
<tr>
<td>Example</td>
<td>SYST:PROM? → SCPI: &gt; 1</td>
</tr>
<tr>
<td>Note</td>
<td></td>
</tr>
</tbody>
</table>

### 2.3.21 SYSTem:LOCal:CONTrol

<table>
<thead>
<tr>
<th>Syntax</th>
<th>SYSTem:LOCal:CONTrol &lt;enable&gt;</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>This command enables/disables local control.</td>
</tr>
<tr>
<td>Parameter</td>
<td>&lt;enable&gt; = &lt;BOOLEAN PROGRAM DATA&gt;</td>
</tr>
<tr>
<td>Response</td>
<td>None.</td>
</tr>
<tr>
<td>Example</td>
<td>SYST:LOC:CONT 1</td>
</tr>
<tr>
<td>Note</td>
<td>This setting is applied all connected sessions and forgets when turn off SCPI.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Syntax</th>
<th>SYSTem:LOCal:CONTrol?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>This query returns enables/disables local control.</td>
</tr>
<tr>
<td>Parameter</td>
<td>None.</td>
</tr>
<tr>
<td>Response</td>
<td>&lt;enable&gt; = &lt;BOOLEAN PROGRAM DATA&gt;</td>
</tr>
<tr>
<td>Example</td>
<td>SYST:LOC:CONT? → 1</td>
</tr>
<tr>
<td>Note</td>
<td></td>
</tr>
</tbody>
</table>
### 2.3.22 SYSTem:NCONnectivity:COMunication:AUTH:TYPE

<table>
<thead>
<tr>
<th>Syntax</th>
<th>SYSTem:NCONnectivity:COMunication:AUTH:TYPE &lt;type&gt;</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>This command sets the authentication type for HTTP requests.</td>
</tr>
<tr>
<td>Parameter</td>
<td>&lt;type&gt; = &lt;CHARACTER PROGRAM DATA&gt;</td>
</tr>
<tr>
<td></td>
<td>NONE: Does not use authentication.</td>
</tr>
<tr>
<td></td>
<td>BASIC: Uses basic authentication.</td>
</tr>
<tr>
<td></td>
<td>OAUTH20: Uses OAuth 2.0.</td>
</tr>
<tr>
<td>Response</td>
<td>None.</td>
</tr>
<tr>
<td>Example</td>
<td>SYST:NCON:COM:AUTH:TYPE BASIC</td>
</tr>
<tr>
<td>Note</td>
<td></td>
</tr>
</tbody>
</table>

### 2.3.23 SYSTem:NCONnectivity:COMunication:BASic

<table>
<thead>
<tr>
<th>Syntax</th>
<th>SYSTem:NCONnectivity:COMunication:BASic &lt;id&gt;,&lt;password&gt;</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>This command sets the ID and password for Basic authentication.</td>
</tr>
<tr>
<td>Parameter</td>
<td>&lt;id&gt; = &lt;STRING PROGRAM DATA&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;password&gt; = &lt;STRING PROGRAM DATA&gt;</td>
</tr>
<tr>
<td>Response</td>
<td>None.</td>
</tr>
<tr>
<td>Example</td>
<td>SYST:NCON:COM:BAS &quot;id&quot;,&quot;password&quot;</td>
</tr>
<tr>
<td>Note</td>
<td>The ID and password can be up to 128 characters.</td>
</tr>
</tbody>
</table>

### 2.3.24 SYSTem:NCONnectivity:COMunication:TURi

<table>
<thead>
<tr>
<th>Syntax</th>
<th>SYSTem:NCONnectivity:COMunication:TURi &lt;uri&gt;</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>This command sets the URI to upload scenario results.</td>
</tr>
<tr>
<td>Parameter</td>
<td>&lt;uri&gt; = &lt;STRING PROGRAM DATA&gt;</td>
</tr>
<tr>
<td>Response</td>
<td>None.</td>
</tr>
<tr>
<td>Example</td>
<td>SYST:NCON:COM:TUR &quot;<a href="https://sample.net/upload">https://sample.net/upload</a>&quot;</td>
</tr>
<tr>
<td>Note</td>
<td>A URI can be up to 2048 characters.</td>
</tr>
</tbody>
</table>

### 2.3.25 SYSTem:NCONnectivity:COMunication:TURi?

<table>
<thead>
<tr>
<th>Syntax</th>
<th>SYSTem:NCONnectivity:COMunication:TURi?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>This query returns the URI to upload scenario results.</td>
</tr>
<tr>
<td>Parameter</td>
<td>None.</td>
</tr>
<tr>
<td>Response</td>
<td>&lt;uri&gt; = &lt;CHARACTER RESPONSE DATA&gt;</td>
</tr>
<tr>
<td>Note</td>
<td>&lt;uri&gt; is the same number of asterisks as the number of characters in the password.</td>
</tr>
</tbody>
</table>
2.3.25 SYSTem:NCONnectivity:COMunication:SET

<table>
<thead>
<tr>
<th>Syntax</th>
<th>SYSTem:NCONnectivity:COMunication:SET &lt;uri&gt;</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>This command sends an HTTP GET request to the specified URI.</td>
</tr>
<tr>
<td>Parameter</td>
<td>&lt;uri&gt; = &lt;STRING PROGRAM DATA&gt;</td>
</tr>
<tr>
<td>Response</td>
<td>None.</td>
</tr>
</tbody>
</table>
| Example      | SYST:NCON:COM:SET "https://sample.net/config?argument=sample"
SYST:NCON:COM:LRET:RST? → 0
SYST:WAIT:DUR 5
SYST:NCON:COM:LRET:RST? → 1
SYST:NCON:COM:LRET:RRES? → {"key":"value"} |
| Note         | The URI can be up to 2048 characters. |

2.3.26 SYSTem:NCONnectivity:COMunication:LRETurn:RRESPonse?

<table>
<thead>
<tr>
<th>Syntax</th>
<th>SYSTem:NCONnectivity:COMunication:LRETurn:RRESPonse?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>This query returns a response to the sent HTTP GET request.</td>
</tr>
<tr>
<td>Parameter</td>
<td>None.</td>
</tr>
<tr>
<td>Response</td>
<td>&lt;response&gt; = &lt;CHARACTER RESPONSE DATA&gt;</td>
</tr>
<tr>
<td>Example</td>
<td>SYST:NCON:COM:LRET:RRES? → {&quot;key&quot;:&quot;value&quot;}</td>
</tr>
<tr>
<td>Note</td>
<td>The maximum number of characters in &lt;response&gt; is 65536.</td>
</tr>
</tbody>
</table>

2.3.27 SYSTem:NCONnectivity:COMunication:LRETurn:RSTatus?

<table>
<thead>
<tr>
<th>Syntax</th>
<th>SYSTem:NCONnectivity:COMunication:LRETurn:RSTatus?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>This query returns whether the Network Master successfully receives a response to the HTTP GET request.</td>
</tr>
<tr>
<td>Parameter</td>
<td>None.</td>
</tr>
</tbody>
</table>
| Response     | <status> = <NR1 NUMERIC RESPONSE DATA>
0: Incomplete
1: Done |
| Example      | SYST:NCON:COM:LRET:RST? → 1 |
| Note         | |

2.3.28 SYSTem:NCONnectivity:COMunication:LRETurn:ERRror?

<table>
<thead>
<tr>
<th>Syntax</th>
<th>SYSTem:NCONnectivity:COMunication:LRETurn:ERRror?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>This query returns an error status when it fails to receive an HTTP response.</td>
</tr>
<tr>
<td>Parameter</td>
<td>None.</td>
</tr>
</tbody>
</table>
| Response     | <error> = <NR1 NUMERIC RESPONSE DATA>
<message> = <STRING RESPONSE DATA>
0: No error.
1: Network Master is not connected to the cloud server.
2: The network was disconnected.
3: Log in to the cloud server, and implement OAuth 2.0 authentication.
4: An unexpected error occurred on the cloud server.
5: Failed to connect to the specified server.
6: An unexpected error was returned from the specified server.
7: The received response is in an invalid format.
8: Invalid ID or password for Basic authentication.
9: The request timed out.
10: Invalid license for MX109020A-003. |
| Note         | |
### 2.3.29 SYSTem:WAIT[:IDLE]

<table>
<thead>
<tr>
<th>Syntax</th>
<th>SYSTem:WAIT[:IDLE]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>This command waits for the instrument to go into IDLE state, i.e. no measurement or test is pending, running, loading or being stored. It also waits for load and save of settings to finish.</td>
</tr>
<tr>
<td>Parameter</td>
<td>None.</td>
</tr>
<tr>
<td>Response</td>
<td>None.</td>
</tr>
<tr>
<td>Example</td>
<td>SYST:WAIT</td>
</tr>
<tr>
<td>Note</td>
<td>There must be a connected application server for this command to be recognized as a legal command. Be careful when using this command as it may lead to undesired blocking of the remote interface. In some situations the instrument requires a remote command or other user intervention in order to return to IDLE state; e.g. when a measurement is running and measurement stop mode is set to MANual (MEAS:SET:STOP → MAN). In this situation, for the instrument to return to IDLE state press the START/STOP button on the GUI or apply the MEASurement:STOP command. The latter is NOT possible if SYST:WAIT:IDLE is currently being executed. If an undesired blocking occurs close and re-open the remote connection. And then send *RST command to reset undesired blocking remote connection.</td>
</tr>
</tbody>
</table>

### 2.3.30 SYSTem:WAIT:DURation

<table>
<thead>
<tr>
<th>Syntax</th>
<th>SYSTem:WAIT:DURation &lt;seconds&gt;</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>This command waits for the specified number of seconds.</td>
</tr>
<tr>
<td>Parameter</td>
<td>&lt;seconds&gt; = &lt;NUMERIC PROGRAM DATA&gt; MINimum = 1, MAXimum = 3600</td>
</tr>
<tr>
<td>Response</td>
<td>None.</td>
</tr>
<tr>
<td>Example</td>
<td>SYST:WAIT:DUR 5</td>
</tr>
<tr>
<td>Note</td>
<td>There must be a connected application server for this command to be recognized as a legal command.</td>
</tr>
</tbody>
</table>
Chapter 2 SCPI Conformance Information

2.4 SCPI Instrument Subsystem Commands

To use the application-specific SCPI commands, you need to connect the client session to the application server. By sending SCPI commands to the application server connected, you can control the application.

2.4.1 Connection to Application Server

To connect the client session to the application server, do one of the following:

- Use the `INST:CONN:ALL` command. This command connects a client session to the application server which is not occupied. If a different client session is already connected to the server (application server is occupied), the connection attempt fails.

- Use the `INST:CONN` command. This command connects a client session to all application servers which are not occupied.

- Use the `INST:STAR` command. This command starts a new application server and connects a client session to the application server.

2.4.2 Connection to multiple applications

A client session can connect to multiple application servers simultaneously. In this case, the SCPI command destination will be the selected application server. The destination can be set by using the `INST:SEL` command. To confirm the selected application server, use the `INST?` command. When a new application server has started by the `INST:STAR` command, the SCPI command destination will be changed to the newly started application server.

2.4.3 Connection from multiple users

When using a Network Master by multi-users, multiple client sessions connect to a Network Master. Under this condition, make sure that the application server is not occupied by any other user.

If a client session has already connected to an application server, Network Master cannot connect any other client session to the application server. To connect to the application server occupied by a client session, you need to release the application server by disconnecting the client session which is connecting currently. To release the application server, do one of the following:

- Use the `INST:DISC` command. You can disconnect the server by sending the `INST:DISC` command from the connecting client session.

- Terminate the Client Session. After the client session terminated, all application servers to which the client session was connected are released. You can terminate the client session by disconnecting connection to Network Master from the User PC.

You can connect a client session to the released application server again by using the `INST:CONN` command or the `INST:CONN:ALL` command.

2.4.4 Force Termination of Application Server

An application server to which a session is connecting doesn’t receive SCPI commands from any other session. An exception is the `INST:TERM:FORC` command. This command is always accepted even if sent from the other session and terminates the application server to which a session is connecting.

continued on next page...
### 2.4 SCPI Instrument Subsystem Commands

#### 2.4.5 INSTrument:STARt[:DEFault]

<table>
<thead>
<tr>
<th>Syntax</th>
<th>INSTrument:STARt[:DEFault] &lt;app name&gt;[, &lt;port name&gt;[, &lt;port name&gt;,...]]</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Description</strong></td>
<td>This command starts an application server with default settings.</td>
</tr>
</tbody>
</table>

**Parameters**

- `<app name>` = `<CHARACTER PROGRAM DATA>`
  - TP-APS-OTN: OTN Automatic Protection Switching application.
  - TP-APS-SDHPDH: SDH/PDH Automatic Protection Switching application.
  - TP-APS-SDHPDH-OTN: SDH/PDH over OTN Automatic Protection Switching application.
  - TP-BERT-CPRI: CPRI Bit Error Rate Test application.
  - TP-BERT-CPRI-OTN: CPRI over OTN Bit Error Rate Test application.
  - TP-BERT-ETH: Ethernet Bit Error Rate Test application.
  - TP-BERT-ETH-OTN: Ethernet over OTN Bit Error Rate Test application.
  - TP-BERT-FC: Fibre Channel Bit Error Rate Test application.
  - TP-BERT-FC-OTN: Fibre Channel over OTN Bit Error Rate Test application.
  - TP-BERT-ROE: eCPRI/RoE Bit Error Rate Test application.
  - TP-BERT-SDHPDH: PDH/SDH Bit Error Rate Test application.
  - TP-BERT-SDHPDH-OTN: PDH/SDH over OTN Bit Error Rate Test application.
  - TP-CABLE-ETH: Ethernet cable test application.
  - TP-CHSTAT-ETH: Ethernet channel statistics application.
  - TP-DISC-ETH: Ethernet discovery application.
  - TP-MONGEN-ETH: Ethernet monitor/generate application.
  - TP-MONGEN-ETH-OTN: Ethernet over OTN monitor/generate application.
  - TP-NOFRAME-DEVICE: No frame device test (Unframed Bit Error Rate Test) application.
  - TP-PASS-CPRI: CPRI pass-through application.
  - TP-PASS-ETH: Ethernet pass-through application.
  - TP-PERF-FC: Fibre Channel performance test application.
  - TP-PERF-FC-OTN: Fibre Channel over OTN performance test application.
  - TP-PING-ETH: Ethernet ICMP ping application.
  - TP-REFL-ETH: Ethernet reflector application.
  - TP-REFL-ETH-OTN: Ethernet over OTN reflector application.
  - TP-REFL-FC: Fibre Channel reflector application.
  - TP-REFL-FC-OTN: Fibre Channel over OTN reflector application.
  - TP-RFC-ETH: Ethernet RFC-2544 test application.
  - TP-RFC-ETH-OTN: Ethernet over OTN RFC-2544 test application.
  - TP-RFC6349-ETH: Ethernet RFC-6349 test application.
  - TP-RTD-OTN: OTN Round Trip Delay test application.
  - TP-RTD-SDHPDH: SDH/PDH Round Trip Delay test application.
  - TP-RTD-SDHPDH-OTN: SDH/PDH over OTN Round Trip Delay test application.
  - TP-SAT-ETH: Ethernet Service Activation Test application.
  - TP-SAT-ETH-OTN: Ethernet over OTN Service Activation Test application.
  - TP-SYNTETEST-ETH: Ethernet sync test application.
  - TP-TRACE-ETH: Ethernet trace-route application.
  - OTDR-OTDR: OTDR application.

- `<port name>` = `<CHARACTER PROGRAM DATA>`
  - 1-PORT1: Port 1 on module 1
  - 1-PORT2: Port 2 on module 1
  - 2-PORT1: Port 1 on module 2
  - 2-PORT2: Port 2 on module 2
  - 1-PORT-SING1: Single port 1 on module 1 (for MU110011A/MU110012A/MU110013A)
  - 1-PORT-MULT1: Port 1 on module 1 (for TP-BERT-ROE with "25G eCPRI/RoE Dual Ability")
  - 2-PORT-MULT2: Port 2 on module 1 (for TP-BERT-ROE with "25G eCPRI/RoE Dual Ability")

The physical port(s) given as parameters are assigned to logical port numbers in the application server. The logical port number range is from 1 to the actual number of port assigned to the application server.

...continued on next page...
The list of ports returned by the INStrument:STATe? <id> command reveals the virtual port number sequence. The ports will be shown sorted in (1st) module order and (2nd) port order.

**Response** None.

**Example**
```
INST:STAR TP-BERT-OTN,1-PORT1
INST?
```
```
→ 2
```

**Note** Operators can get started application server ID by using the INStrument:SELect? command. When you start an application server, the application server will be connected and selected automatically. When using this command, the application server will be started with DEFault setup.

### 2.4.6 INStrument:STARt:LAST

**Syntax**
```
INSTrument:STARt:LAST <app name>[, <port name>(), <port name>(), ...]
```

**Description** This command starts an application server and loads the applicable auto saved settings.

**Parameters** The parameters of this command are similar to the parameters of the INStrument:STARt[:DEFault] command above.

**Response** None.

**Example**
```
INST:STAR TP-BERT-OTN,1-PORT1
INST?
```
```
→ 2
```

**Note** When you start an application server, the application server will be connected and selected automatically. When using this command, the application server will be started with LAST setup.

### 2.4.7 INStrument:STARt:GUI

**Syntax**
```
INSTrument:STARt:GUI [<test index>]
```

**Description** This command starts the GUI display during remote control.

**Parameter**
```
<test index> = <NUMERIC PROGRAM DATA>
```

**Response** None.

**Example**
```
INST:STAR TP-BERT-OTN,1-PORT1
INST?
```
```
→ 2
```

**Note** Must connect to the application server first. Defaults to the current application server if a value is omitted.

### 2.4.8 INStrument:TERMinate

**Syntax**
```
INSTrument:TERMinate [<test index>]
```

**Description** This command terminates an application server.

**Parameter**
```
<test index> = <NUMERIC PROGRAM DATA>
```

**Response** None.

**Example**
```
INST:STAR TP-BERT-OTN,1-PORT1
INST?
```
```
→ 2
```

**Note** Must connect to the application server first. Defaults to the current application server if a value is omitted.
2.4 SCPI Instrument Subsystem Commands

Figure 2.1: INST:STAR automatically connects and selects started application server.

Figure 2.2: Original client session has to disconnect, before another client session can connect.

Figure 2.3: When connected to multiple application servers, client session selects to which application server the application specific commands are dispatched.

### 2.4.9 INSTRument:TERMinate:FORCE

<table>
<thead>
<tr>
<th>Syntax</th>
<th>INSTRument:TERMinate:FORCE [&lt;test index&gt;]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>This command force terminates an application server.</td>
</tr>
<tr>
<td>Parameter</td>
<td>&lt;$test index$&gt; = &lt;$NUMERIC PROGRAM DATA$&gt;</td>
</tr>
<tr>
<td>Response</td>
<td>None.</td>
</tr>
</tbody>
</table>
| Example         | INST:STAR TP-BERT-OTN,1-PORT1  
|                 | INST? → 2  
|                 | INST:TERM:FORC 2 |
| Note            | WARNING. This command can terminate the application to which the other session communicating.  
|                 | Defaults to the current application server if a value is omitted. |
## 2.4.10 INSTrument:COUNt?

<table>
<thead>
<tr>
<th>Syntax</th>
<th>INSTrument:COUNt?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>This query returns the number of active application servers.</td>
</tr>
<tr>
<td>Parameter</td>
<td>None.</td>
</tr>
<tr>
<td>Response</td>
<td>&lt;count&gt; = &lt;NR1 NUMERIC RESPONSE DATA&gt;</td>
</tr>
<tr>
<td>Example</td>
<td>INST:COUN? → 2</td>
</tr>
<tr>
<td>Note</td>
<td></td>
</tr>
</tbody>
</table>

### 2.4.11 INSTrument:CATalog?

<table>
<thead>
<tr>
<th>Syntax</th>
<th>INSTrument:CATalog?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>This query returns test indices, application name and port name of all active application servers.</td>
</tr>
<tr>
<td>Parameter</td>
<td>None.</td>
</tr>
<tr>
<td>Response</td>
<td>&lt;test index&gt; = &lt;EXPRESSION RESPONSE DATA&gt;</td>
</tr>
<tr>
<td>Example</td>
<td>INST:CAT? → (1,TP-SAT-ETH,1-PORT1),(2,TP-BERT-SDHPDH,1-PORT2)</td>
</tr>
<tr>
<td>Note</td>
<td>Return -1 if no application server is running.</td>
</tr>
</tbody>
</table>

### 2.4.12 INSTrument:STATe?

<table>
<thead>
<tr>
<th>Syntax</th>
<th>INSTrument:STATe? &lt;test index&gt;</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>This query returns status information about a given application server.</td>
</tr>
<tr>
<td>Parameter</td>
<td>&lt;test index&gt; = &lt;NUMERIC PROGRAM DATA&gt;</td>
</tr>
<tr>
<td>Response</td>
<td>&lt;app name&gt; = &lt;CHARACTER RESPONSE DATA&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;client connection&gt; = &lt;CHARACTER RESPONSE DATA&gt;</td>
</tr>
<tr>
<td></td>
<td>NON IP Address</td>
</tr>
<tr>
<td></td>
<td>&lt;select status&gt; = &lt;CHARACTER RESPONSE DATA&gt;</td>
</tr>
<tr>
<td></td>
<td>NON SELECTED</td>
</tr>
<tr>
<td></td>
<td>&lt;port name&gt; = &lt;CHARACTER RESPONSE DATA&gt;</td>
</tr>
<tr>
<td>Example</td>
<td>INST:STAT? 1 → TP-BERT-OTN,192.168.128.21,SELECTED,1-PORT1,2-PORT2...</td>
</tr>
<tr>
<td>Note</td>
<td></td>
</tr>
</tbody>
</table>

### 2.4.13 INSTrument:CONNect

<table>
<thead>
<tr>
<th>Syntax</th>
<th>INSTrument:CONNect &lt;test index&gt;</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>This command allows client session to connect to an existing application server.</td>
</tr>
<tr>
<td>Parameter</td>
<td>&lt;test index&gt; = &lt;NUMERIC PROGRAM DATA&gt;</td>
</tr>
<tr>
<td>Response</td>
<td>None.</td>
</tr>
<tr>
<td>Example</td>
<td>INST:CONN 1</td>
</tr>
<tr>
<td>Note</td>
<td>This command fails if the application server is already connect to by another client session. Use INSTrument:CATalog? query to acquire the list of all existing application servers. If command succeeds, the application server will be selected automatically.</td>
</tr>
</tbody>
</table>
### 2.4.14 INSTrument:CONNect:ALL

<table>
<thead>
<tr>
<th>Syntax</th>
<th>INSTrument:CONNect:ALL</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>This command allows client session to connect to all existing application servers.</td>
</tr>
<tr>
<td>Parameter</td>
<td>None.</td>
</tr>
<tr>
<td>Response</td>
<td>None.</td>
</tr>
<tr>
<td>Example</td>
<td>INST:CONN:ALL</td>
</tr>
<tr>
<td>Note</td>
<td>This command fails if no application server is selected when the command exits, e.g. because all application servers was already connected to by other client sessions, or because there are no application servers at all. If connected to multiple application servers, the application server with the lowest index will be selected, but selected index will not change if an application server was already selected prior to issuing this command.</td>
</tr>
</tbody>
</table>

### 2.4.15 INSTrument:CONNect[:CATalog]?

<table>
<thead>
<tr>
<th>Syntax</th>
<th>INSTrument:CONNect[:CATalog]?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>This query returns indices of all application servers for the current client session.</td>
</tr>
<tr>
<td>Parameter</td>
<td>None.</td>
</tr>
<tr>
<td>Response</td>
<td>&lt;test index&gt; = &lt;EXPRESSION RESPONSE DATA&gt;</td>
</tr>
<tr>
<td>Example</td>
<td>INST:CONN? (\rightarrow 0,1,...)</td>
</tr>
<tr>
<td>Note</td>
<td>Return -1 if current client session has no application servers.</td>
</tr>
</tbody>
</table>

### 2.4.16 INSTrument:DISConnect

<table>
<thead>
<tr>
<th>Syntax</th>
<th>INSTrument:DISConnect &lt;test index&gt;</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>This command disconnect the application server from the client session.</td>
</tr>
<tr>
<td>Parameter</td>
<td>&lt;test index&gt; = &lt;NUMERIC PROGRAM DATA&gt;</td>
</tr>
<tr>
<td>Response</td>
<td>None.</td>
</tr>
<tr>
<td>Example</td>
<td>INST:DISC 1</td>
</tr>
<tr>
<td>Note</td>
<td>When current application is disconnected, the application server lowest ID will be selected automatically. When a client session is disconnected all the currently connected application servers will be disconnected automatically.</td>
</tr>
</tbody>
</table>

### 2.4.17 INSTrument[:SELect]

<table>
<thead>
<tr>
<th>Syntax</th>
<th>INSTrument[:SELect] &lt;test index&gt;</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>This command select the current application server.</td>
</tr>
<tr>
<td>Parameter</td>
<td>&lt;test index&gt; = &lt;NUMERIC PROGRAM DATA&gt;</td>
</tr>
<tr>
<td>Response</td>
<td>None.</td>
</tr>
<tr>
<td>Example</td>
<td>INST:STAR TP-BERT-OTN,1-PORT1</td>
</tr>
<tr>
<td></td>
<td>INST? (\rightarrow 1)</td>
</tr>
<tr>
<td></td>
<td>INST:STAR TP-BERT-OTN,1-PORT2</td>
</tr>
<tr>
<td></td>
<td>INST? (\rightarrow 2)</td>
</tr>
<tr>
<td></td>
<td>INST 1</td>
</tr>
<tr>
<td>Note</td>
<td>All future commands are forwarded to the current application server until the current application server is changed.</td>
</tr>
</tbody>
</table>
Chapter 2 SCPI Conformance Information

### 2.4.18 INStrument:ERRor[:NEXT]?

<table>
<thead>
<tr>
<th>Syntax</th>
<th>INStrument:ERRor[:NEXT]?</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Description</strong></td>
<td>This query returns the oldest entry of the event queue for the currently selected application server and removes the returned entry from the queue.</td>
</tr>
<tr>
<td><strong>Parameter</strong></td>
<td>None.</td>
</tr>
<tr>
<td><strong>Response</strong></td>
<td>&lt;description&gt; = &lt;STRING RESPONSE DATA&gt;</td>
</tr>
<tr>
<td><strong>Example</strong></td>
<td>INST:ERR? → &quot;Signal abnormal&quot;</td>
</tr>
<tr>
<td><strong>Note</strong></td>
<td>Application server has its own event queue. This event queue is not destroyed when the session is closed. Before checking the server status by using this command, controller may check DB2 (Event Queue Summary) of the Status Byte Register to see if it is set. DB2 of the Status Byte Register will aggregate input from potentially many application servers, and clear if event queues of all servers are empty.</td>
</tr>
</tbody>
</table>

### 2.4.19 INStrument:PORT?

<table>
<thead>
<tr>
<th>Syntax</th>
<th>INStrument:PORT?</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Description</strong></td>
<td>This query returns ports assigned for the currently selected application server.</td>
</tr>
<tr>
<td><strong>Parameter</strong></td>
<td>None.</td>
</tr>
</tbody>
</table>
| **Response**         | <port name> = <CHARACTER RESPONSE DATA> 
Expression format: Character list |
| **Example**          | INST:PORT? → 1-PORT1,1-PORT2 |
| **Note**             | Operates on the application server currently selected by the INStrument:SELect command. Returns NON if the client session does not have a currently selected application server. Returns NON if no ports are assigned for the currently selected application server. |

### 2.4.20 INStrument:PORT:FREE? <app name>

<table>
<thead>
<tr>
<th>Syntax</th>
<th>INStrument:PORT:FREE? &lt;app name&gt;</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Description</strong></td>
<td>This query returns all unused ports for the target application name.</td>
</tr>
<tr>
<td><strong>Parameter</strong></td>
<td>&lt;app name&gt; = &lt;CHARACTER PROGRAM DATA&gt;</td>
</tr>
</tbody>
</table>
| **Response**         | <port name> = <CHARACTER RESPONSE DATA> 
Expression format: Character list |
| **Example**          | INST:PORT:FREE? TP-BERT-OTN → 1-PORT1,1-PORT2 |
| **Note**             | |
### 2.4.21 INSTrument:PORT:CATalog?

<table>
<thead>
<tr>
<th>Syntax</th>
<th>INSTrument:PORT:CATalog?</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Description</strong></td>
<td>This query returns all ports of device.</td>
</tr>
<tr>
<td><strong>Parameter</strong></td>
<td>None.</td>
</tr>
</tbody>
</table>
| **Response**      | <port name> = <CHARACTER RESPONSE DATA>  
Expression format: Character list |
| **Example**       | INST:PORT:CAT? → 1-PORT1,1-PORT2 |
| **Note**          | When SCPI client uses any of the STATus:PORT: commands, <bit> index is the same as returned by the INSTrument:PORT:CATalog? |

### 2.4.22 INSTrument:MODule:CATalog?

<table>
<thead>
<tr>
<th>Syntax</th>
<th>INSTrument:MODule:CATalog?</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Description</strong></td>
<td>This query returns module names of device.</td>
</tr>
<tr>
<td><strong>Parameter</strong></td>
<td>None.</td>
</tr>
<tr>
<td><strong>Response</strong></td>
<td>{&lt;module n&gt;,}* = &lt;CHARACTER RESPONSE DATA&gt;</td>
</tr>
<tr>
<td><strong>Example</strong></td>
<td>INST:MOD:CAT? → MU100010A,MU100011A</td>
</tr>
</tbody>
</table>

### 2.4.23 INSTrument:CTRL:NAME?

<table>
<thead>
<tr>
<th>Syntax</th>
<th>INSTrument:CTRL:NAME?</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Description</strong></td>
<td>This query returns model name.</td>
</tr>
<tr>
<td><strong>Parameter</strong></td>
<td>None.</td>
</tr>
<tr>
<td><strong>Response</strong></td>
<td>&lt;model name&gt; = &lt;CHARACTER RESPONSE DATA&gt;</td>
</tr>
<tr>
<td><strong>Example</strong></td>
<td>INST:CTRL:NAME? → MT1000A</td>
</tr>
</tbody>
</table>

### 2.4.24 INSTrument:CTRL:SN?

<table>
<thead>
<tr>
<th>Syntax</th>
<th>INSTrument:CTRL:SN?</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Description</strong></td>
<td>This query returns controller serial number.</td>
</tr>
<tr>
<td><strong>Parameter</strong></td>
<td>None.</td>
</tr>
<tr>
<td><strong>Response</strong></td>
<td>&lt;serial number&gt; = &lt;CHARACTER RESPONSE DATA&gt;</td>
</tr>
<tr>
<td><strong>Example</strong></td>
<td>INST:CTRL:SN? → 1234567890</td>
</tr>
</tbody>
</table>

### 2.4.25 INSTrument:CTRL:TRT?

<table>
<thead>
<tr>
<th>Syntax</th>
<th>INSTrument:CTRL:TRT?</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Description</strong></td>
<td>This query returns controller total run time(sec).</td>
</tr>
<tr>
<td><strong>Parameter</strong></td>
<td>None.</td>
</tr>
<tr>
<td><strong>Response</strong></td>
<td>&lt;time&gt; = &lt;NR1 NUMERIC RESPONSE DATA&gt;</td>
</tr>
<tr>
<td><strong>Example</strong></td>
<td>INST:CTRL:TRT? → 5000000</td>
</tr>
</tbody>
</table>

### 2.4.26 INSTrument:CTRL:OPTion:CATalog?

<table>
<thead>
<tr>
<th>Syntax</th>
<th>INSTrument:CTRL:OPTion:CATalog?</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Description</strong></td>
<td>This query returns controller enabled options.</td>
</tr>
<tr>
<td><strong>Parameter</strong></td>
<td>None.</td>
</tr>
<tr>
<td><strong>Response</strong></td>
<td>{&lt;option n&gt;,}* = &lt;CHARACTER RESPONSE DATA&gt;</td>
</tr>
<tr>
<td><strong>Example</strong></td>
<td>INST:CTRL:OPT:CAT? → MT1000A-303,MT1000A-005</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Note</strong></th>
<th></th>
</tr>
</thead>
</table>
### 2.4.27 INSTrument:MODule<Md>:NAME?

<table>
<thead>
<tr>
<th>Syntax</th>
<th>INSTrument:MODule&lt;Md&gt;:NAME?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>This query returns module model number.</td>
</tr>
<tr>
<td>Parameter</td>
<td>None.</td>
</tr>
<tr>
<td>Response</td>
<td>&lt;model name&gt; = &lt;CHARACTER RESPONSE DATA&gt;</td>
</tr>
<tr>
<td>Example</td>
<td>INST:MOD1:NAME? → MU100010A</td>
</tr>
</tbody>
</table>

### 2.4.28 INSTrument:MODule<Md>:SN?

<table>
<thead>
<tr>
<th>Syntax</th>
<th>INSTrument:MODule&lt;Md&gt;:SN?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>This query returns module serial number.</td>
</tr>
<tr>
<td>Parameter</td>
<td>None.</td>
</tr>
<tr>
<td>Response</td>
<td>&lt;serial number&gt; = &lt;CHARACTER RESPONSE DATA&gt;</td>
</tr>
<tr>
<td>Example</td>
<td>INST:MOD1:SN? → 1234567890</td>
</tr>
</tbody>
</table>

### 2.4.29 INSTrument:MODule<Md>:TRT?

<table>
<thead>
<tr>
<th>Syntax</th>
<th>INSTrument:MODule&lt;Md&gt;:TRT?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>This query returns module total run time(sec).</td>
</tr>
<tr>
<td>Parameter</td>
<td>None.</td>
</tr>
<tr>
<td>Response</td>
<td>&lt;time&gt; = &lt;NR1 NUMERIC RESPONSE DATA&gt;</td>
</tr>
<tr>
<td>Example</td>
<td>INST:MOD1:TRT? → 5000000</td>
</tr>
</tbody>
</table>

### 2.4.30 INSTrument:MODule<Md>:OPTion:CATalog?

<table>
<thead>
<tr>
<th>Syntax</th>
<th>INSTrument:MODule&lt;Md&gt;:OPTion:CATalog?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>This query returns module enabled options.</td>
</tr>
<tr>
<td>Parameter</td>
<td>None.</td>
</tr>
<tr>
<td>Response</td>
<td>{&lt;option n&gt;,}^* = &lt;CHARACTER RESPONSE DATA&gt;</td>
</tr>
</tbody>
</table>

Note
2.5 SCPI Status Subsystem Commands

2.5.1 STATus:OPERation[:EVENt]?

<table>
<thead>
<tr>
<th>Syntax</th>
<th>STATus:OPERation[:EVENt]?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>This query returns and clears the operation event register.</td>
</tr>
<tr>
<td>Parameter</td>
<td>None.</td>
</tr>
<tr>
<td>Response</td>
<td>&lt;value&gt; = &lt;NR1 NUMERIC RESPONSE DATA&gt;</td>
</tr>
<tr>
<td></td>
<td>The bits and their values for the register:</td>
</tr>
<tr>
<td></td>
<td>DB1-DB4 = NOT USED</td>
</tr>
<tr>
<td></td>
<td>DB5 (16) = Measuring</td>
</tr>
<tr>
<td></td>
<td>DB6-DB16 = NOT USED</td>
</tr>
<tr>
<td>Example</td>
<td>STAT:OPER? → 16</td>
</tr>
<tr>
<td>Note</td>
<td>All active sessions have their own register and it is cleared when the session starts.</td>
</tr>
</tbody>
</table>

2.5.2 STATus:OPERation:CONDition?

<table>
<thead>
<tr>
<th>Syntax</th>
<th>STATus:OPERation:CONDition?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>This query returns the operation condition register.</td>
</tr>
<tr>
<td>Parameter</td>
<td>None.</td>
</tr>
<tr>
<td>Response</td>
<td>&lt;value&gt; = &lt;NR1 NUMERIC RESPONSE DATA&gt;</td>
</tr>
<tr>
<td></td>
<td>The bits and their values for the register:</td>
</tr>
<tr>
<td></td>
<td>DB1-DB4 = NOT USED</td>
</tr>
<tr>
<td></td>
<td>DB5 (16) = Measuring</td>
</tr>
<tr>
<td></td>
<td>DB6-DB16 = NOT USED</td>
</tr>
<tr>
<td>Example</td>
<td>STAT:OPER:COND? → 16</td>
</tr>
<tr>
<td>Note</td>
<td></td>
</tr>
</tbody>
</table>

2.5.3 STATus:OPERation:ENABle

<table>
<thead>
<tr>
<th>Syntax</th>
<th>STATus:OPERation:ENABle &lt;mask&gt;</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>This command sets the enable mask for the operation event register.</td>
</tr>
<tr>
<td>Parameter</td>
<td>&lt;mask&gt; = &lt;NUMERIC PROGRAM DATA&gt;</td>
</tr>
<tr>
<td></td>
<td>The bits and their values for the register:</td>
</tr>
<tr>
<td></td>
<td>DB1-DB4 = NOT USED</td>
</tr>
<tr>
<td></td>
<td>DB5 (16) = Measuring</td>
</tr>
<tr>
<td></td>
<td>DB6-DB16 = NOT USED</td>
</tr>
<tr>
<td></td>
<td>MINimum = 0, MAXimum = 65535</td>
</tr>
<tr>
<td>Response</td>
<td>None.</td>
</tr>
<tr>
<td>Example</td>
<td>STAT:OPER:ENAB 65535</td>
</tr>
<tr>
<td>Note</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Syntax</th>
<th>STATus:OPERation:ENABle?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>This query returns the enable mask for the operation event register.</td>
</tr>
<tr>
<td>Parameter</td>
<td>None.</td>
</tr>
<tr>
<td>Response</td>
<td>&lt;mask&gt; = &lt;NR1 NUMERIC RESPONSE DATA&gt;</td>
</tr>
<tr>
<td>Example</td>
<td>STAT:OPER:ENAB? → 16</td>
</tr>
<tr>
<td>Note</td>
<td></td>
</tr>
</tbody>
</table>
### 2.5.4 STATus:OPERation:PTRansition

<table>
<thead>
<tr>
<th>Syntax</th>
<th>STATus:OPERation:PTRansition &lt;mask&gt;</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>This command sets the positive transition filter for the operation event register.</td>
</tr>
<tr>
<td>Parameter</td>
<td>&lt;mask&gt; = &lt;NUMERIC PROGRAM DATA&gt;</td>
</tr>
<tr>
<td></td>
<td>The bits and their values for the register:</td>
</tr>
<tr>
<td></td>
<td>DB1-DB4 = NOT USED</td>
</tr>
<tr>
<td></td>
<td>DB5 (16) = Measuring</td>
</tr>
<tr>
<td></td>
<td>DB6-DB16 = NOT USED</td>
</tr>
<tr>
<td></td>
<td>MINimum = 0, DEFault = 65535, MAXimum = 65535</td>
</tr>
<tr>
<td>Response</td>
<td>None.</td>
</tr>
<tr>
<td>Example</td>
<td>STAT:OPER:PTR 16384</td>
</tr>
<tr>
<td>Note</td>
<td></td>
</tr>
</tbody>
</table>

### 2.5.5 STATus:OPERation:NTRansition

<table>
<thead>
<tr>
<th>Syntax</th>
<th>STATus:OPERation:NTRansition &lt;mask&gt;</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>This command sets the negative transition filter for the operation event register.</td>
</tr>
<tr>
<td>Parameter</td>
<td>&lt;mask&gt; = &lt;NUMERIC PROGRAM DATA&gt;</td>
</tr>
<tr>
<td></td>
<td>The bits and their values for the register:</td>
</tr>
<tr>
<td></td>
<td>DB1-DB4 = NOT USED</td>
</tr>
<tr>
<td></td>
<td>DB5 (16) = Measuring</td>
</tr>
<tr>
<td></td>
<td>DB6-DB16 = NOT USED</td>
</tr>
<tr>
<td></td>
<td>MINimum = 0, MAXimum = 65535</td>
</tr>
<tr>
<td>Response</td>
<td>None.</td>
</tr>
<tr>
<td>Example</td>
<td>STAT:OPER:NTR 16384</td>
</tr>
<tr>
<td>Note</td>
<td></td>
</tr>
</tbody>
</table>

### 2.5.6 STATus:QUEStionable[:EVENt]?

<table>
<thead>
<tr>
<th>Syntax</th>
<th>STATus:QUEStionable[:EVENt]?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>This query returns and clears the questionable event register.</td>
</tr>
<tr>
<td>Parameter</td>
<td>None.</td>
</tr>
<tr>
<td>Response</td>
<td>&lt;value&gt; = &lt;NR1 NUMERIC RESPONSE DATA&gt;</td>
</tr>
<tr>
<td></td>
<td>The bits and their values for the register:</td>
</tr>
<tr>
<td></td>
<td>DB1-DB14 = NOT USED</td>
</tr>
<tr>
<td></td>
<td>DB15 (16384) = Command Warning</td>
</tr>
<tr>
<td></td>
<td>DB16 = NOT USED</td>
</tr>
<tr>
<td>Example</td>
<td>STAT:QUES?  → 16384</td>
</tr>
<tr>
<td>Note</td>
<td>All active sessions has their own register and it is cleared when the session starts.</td>
</tr>
</tbody>
</table>
### 2.5.7 STATus:QUEStionable:CONDition?

<table>
<thead>
<tr>
<th>Syntax</th>
<th>STATus:QUEStionable:CONDition?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>This query returns the questionable condition register.</td>
</tr>
<tr>
<td>Parameter</td>
<td>None.</td>
</tr>
<tr>
<td>Response</td>
<td><code>&lt;value&gt; = &lt;NR1 NUMERIC RESPONSE DATA&gt;</code></td>
</tr>
<tr>
<td></td>
<td>The bits and their values for the register:</td>
</tr>
<tr>
<td></td>
<td>DB1-DB14 = NOT USED</td>
</tr>
<tr>
<td></td>
<td>DB15 (16384) = Command Warning</td>
</tr>
<tr>
<td></td>
<td>DB16 = NOT USED</td>
</tr>
<tr>
<td>Example</td>
<td>STAT:QUES:COND? → 16384</td>
</tr>
<tr>
<td>Note</td>
<td></td>
</tr>
</tbody>
</table>

### 2.5.8 STATus:QUEStionable:ENABle

<table>
<thead>
<tr>
<th>Syntax</th>
<th>STATus:QUEStionable:ENABle <code>&lt;mask&gt;</code></th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>This command sets the enable mask for the questionable event register.</td>
</tr>
<tr>
<td>Parameter</td>
<td><code>&lt;mask&gt; = &lt;NUMERIC PROGRAM DATA&gt;</code></td>
</tr>
<tr>
<td></td>
<td>The bits and their values for the register:</td>
</tr>
<tr>
<td></td>
<td>DB1-DB14 = NOT USED</td>
</tr>
<tr>
<td></td>
<td>DB15 (16384) = Command Warning</td>
</tr>
<tr>
<td></td>
<td>DB16 = NOT USED</td>
</tr>
<tr>
<td></td>
<td>MINimum = 0, MAXimum = 65535</td>
</tr>
<tr>
<td>Response</td>
<td>None.</td>
</tr>
<tr>
<td>Example</td>
<td>STAT:QUES:ENAB 16384</td>
</tr>
<tr>
<td>Note</td>
<td></td>
</tr>
</tbody>
</table>

### 2.5.9 STATus:QUEStionable:PTRansition

<table>
<thead>
<tr>
<th>Syntax</th>
<th>STATus:QUEStionable:PTRansition <code>&lt;mask&gt;</code></th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>This command sets the positive transition filter for the questionable event register.</td>
</tr>
<tr>
<td>Parameter</td>
<td><code>&lt;mask&gt; = &lt;NUMERIC PROGRAM DATA&gt;</code></td>
</tr>
<tr>
<td></td>
<td>The bits and their values for the register:</td>
</tr>
<tr>
<td></td>
<td>DB1-DB14 = NOT USED</td>
</tr>
<tr>
<td></td>
<td>DB15 (16384) = Command Warning</td>
</tr>
<tr>
<td></td>
<td>DB16 = NOT USED</td>
</tr>
<tr>
<td></td>
<td>MINimum = 0, DEFault = 65535, MAXimum = 65535</td>
</tr>
<tr>
<td>Response</td>
<td>None.</td>
</tr>
<tr>
<td>Example</td>
<td>STAT:QUES:PTR 16384</td>
</tr>
<tr>
<td>Note</td>
<td></td>
</tr>
</tbody>
</table>

### 2.5.8 STATus:QUEStionable:ENABle?

<table>
<thead>
<tr>
<th>Syntax</th>
<th>STATus:QUEStionable:ENABle?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>This query returns the enable mask for the questionable event register.</td>
</tr>
<tr>
<td>Parameter</td>
<td>None.</td>
</tr>
<tr>
<td>Response</td>
<td><code>&lt;mask&gt; = &lt;NR1 NUMERIC RESPONSE DATA&gt;</code></td>
</tr>
<tr>
<td>Example</td>
<td>STAT:QUES:ENAB? → 16384</td>
</tr>
<tr>
<td>Note</td>
<td></td>
</tr>
</tbody>
</table>

### 2.5.9 STATus:QUEStionable:PTRansition?

<table>
<thead>
<tr>
<th>Syntax</th>
<th>STATus:QUEStionable:PTRansition?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>This query returns the positive transition filter for the questionable event register.</td>
</tr>
<tr>
<td>Parameter</td>
<td>None.</td>
</tr>
<tr>
<td>Response</td>
<td><code>&lt;mask&gt; = &lt;NR1 NUMERIC RESPONSE DATA&gt;</code></td>
</tr>
<tr>
<td>Example</td>
<td>STAT:QUES:PTR? → 16384</td>
</tr>
<tr>
<td>Note</td>
<td></td>
</tr>
</tbody>
</table>
### 2.5.10 STAT:QUEStionable:NTRansition

**Syntax**

```
STAT:QUEStionable:NTRansition <mask>
```

**Description**

This command sets the negative transition filter for the questionable event register.

**Parameter**

```
<mask> = <NUMERIC PROGRAM DATA>
```

The bits and their values for the register:

- DB1-DB14 = NOT USED
- DB15 (16384) = Command Warning
- DB16 = NOT USED

```
MINimum = 0, MAXimum = 65535
```

**Response**

None.

**Example**

```
STAT:QUE:NTR 16384
```

**Note**

### 2.5.11 STAT:PORT[:EVENt]?

**Syntax**

```
STAT:PORT[:EVENt]? <port name>
```

**Description**

This query returns and clears the port event register.

**Parameter**

```
<port name> = <CHARACTER PROGRAM DATA>
```

Expression format: Character list

**Response**

```
-enable> = <BOOLEAN RESPONSE DATA>
```

**Example**

```
STAT:PORT? 1-PORT1
```

**Note**

All active sessions have their own register and it is cleared when the session starts.

```
<port name> is the same as the one returned by INSTRument:PORT:CATalog?.
```

### 2.5.12 STAT:PORT:CONDition?

**Syntax**

```
STAT:PORT:CONDition? <port name>
```

**Description**

This query returns the port condition register.

**Parameter**

```
<port name> = <CHARACTER PROGRAM DATA>
```

Expression format: Character list

**Response**

```
-enable> = <BOOLEAN RESPONSE DATA>
```

**Example**

```
STAT:PORT:COND? 1-PORT1
```

**Note**

```
<port name> is the same as one of the returned by the INSTRument:PORT:CATalog?.
```

### 2.5.13 STAT:PORT:ENABle

**Syntax**

```
STAT:PORT:ENABle <port name>,<enable>
```

**Description**

This command sets the enable mask for the port event register.

**Parameters**

```
<port name> = <CHARACTER PROGRAM DATA>
```

Expression format: Character list

```
<enable> = <BOOLEAN PROGRAM DATA>
```

**Response**

None.

**Example**

```
STAT:PORT:ENAB 1-PORT1,ON
```

**Note**

```
<port name> is the same as one of the returned by the INSTRument:PORT:CATalog?.
```
### 2.5 SCPI Status Subsystem Commands

<table>
<thead>
<tr>
<th>Syntax</th>
<th>Description</th>
<th>Parameter</th>
<th>Response</th>
<th>Example</th>
<th>Note</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>STATus:PORT:ENABle?</strong> &lt;port name&gt;</td>
<td>This query returns the enable mask for the port event register.</td>
<td>&lt;port name&gt; = &lt;CHARACTER PROGRAM DATA&gt; &lt;enable&gt; = &lt;BOOLEAN RESPONSE DATA&gt; Expression format: Character list</td>
<td></td>
<td></td>
<td>&lt;port name&gt; is the same as one of the returned by the INSTRument:PORT:CATalog?</td>
</tr>
</tbody>
</table>

#### 2.5.14 STATus:PORT:PTRansition

<table>
<thead>
<tr>
<th>Syntax</th>
<th>Description</th>
<th>Parameter</th>
<th>Response</th>
<th>Example</th>
<th>Note</th>
</tr>
</thead>
<tbody>
<tr>
<td>STATus:PORT:PTRansition &lt;port name&gt;,&lt;enable&gt;</td>
<td>This command sets the positive transition filter for the port event register.</td>
<td>&lt;port name&gt; = &lt;CHARACTER PROGRAM DATA&gt; &lt;enable&gt; = &lt;BOOLEAN PROGRAM DATA&gt; Expression format: Character list</td>
<td>None.</td>
<td>STAT:PORT:PTR 1-PORT1,ON</td>
<td>&lt;port name&gt; is the same as one of the returned by the INSTRument:PORT:CATalog?</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Syntax</th>
<th>Description</th>
<th>Parameter</th>
<th>Response</th>
<th>Example</th>
<th>Note</th>
</tr>
</thead>
<tbody>
<tr>
<td>STATus:PORT:PTRansition? &lt;port name&gt;</td>
<td>This query returns the positive transition filter for the port event register.</td>
<td>&lt;port name&gt; = &lt;CHARACTER PROGRAM DATA&gt; &lt;enable&gt; = &lt;BOOLEAN RESPONSE DATA&gt; Expression format: Character list</td>
<td></td>
<td></td>
<td>&lt;port name&gt; is the same as one of the returned by the INSTRument:PORT:CATalog?</td>
</tr>
</tbody>
</table>

#### 2.5.15 STATus:PORT:NTRansition

<table>
<thead>
<tr>
<th>Syntax</th>
<th>Description</th>
<th>Parameter</th>
<th>Response</th>
<th>Example</th>
<th>Note</th>
</tr>
</thead>
<tbody>
<tr>
<td>STATus:PORT:NTRansition &lt;port name&gt;,&lt;enable&gt;</td>
<td>This command sets the negative transition filter for the port event register.</td>
<td>&lt;port name&gt; = &lt;CHARACTER PROGRAM DATA&gt; &lt;enable&gt; = &lt;BOOLEAN PROGRAM DATA&gt; Expression format: Character list</td>
<td>None.</td>
<td>STAT:PORT:NTR 1-PORT1,OFF</td>
<td>&lt;port name&gt; is the same as one of the returned by the INSTRument:PORT:CATalog?</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Syntax</th>
<th>Description</th>
<th>Parameter</th>
<th>Response</th>
<th>Example</th>
<th>Note</th>
</tr>
</thead>
<tbody>
<tr>
<td>STATus:PORT:NTRansition? &lt;port name&gt;</td>
<td>This query returns the negative transition filter for the port event register.</td>
<td>&lt;port name&gt; = &lt;CHARACTER PROGRAM DATA&gt; &lt;enable&gt; = &lt;BOOLEAN RESPONSE DATA&gt; Expression format: Character list</td>
<td></td>
<td></td>
<td>&lt;port name&gt; is the same as one of the returned by the INSTRument:PORT:CATalog?</td>
</tr>
</tbody>
</table>
### 2.5.16 STATUs:PRESet

<table>
<thead>
<tr>
<th>Syntax</th>
<th>STATus:PRESet</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>For the instrument-dependent status data structures, the PRESet command sets the enable register to all 1’s and the transition filter register to recognize only positive transitions. For the SCPI-mandated status structures (operation, questionable and port status) the PRESet command sets the transition filter registers to recognize only positive transitions and set the enable registers to 0’s. This command does not affect either the Status Byte or the Standard Event Status register. PRESet does not clear any of the event registers or any item from the error or event queues.</td>
</tr>
<tr>
<td>Parameter</td>
<td>None.</td>
</tr>
<tr>
<td>Response</td>
<td>None.</td>
</tr>
<tr>
<td>Example</td>
<td>STAT:PRES</td>
</tr>
<tr>
<td>Note</td>
<td>All active sessions has their own set of registers. This command affects registers in all connected application servers.</td>
</tr>
</tbody>
</table>
2.6 Mass Memory Subsystem Commands

The commands in this section operates on files and directories placed in the following storage areas:

<table>
<thead>
<tr>
<th>Location</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Internal/</td>
<td>The internal storage of the Network Master.</td>
</tr>
<tr>
<td>Usb/</td>
<td>An USB connected storage device. This location is only accessible when a USB storage device is mounted.</td>
</tr>
<tr>
<td>Internal/remote/</td>
<td>A remote network drive. Refer to the User Manual for information on how to connect to an external storage location. This location is only accessible when external storage is configured and the Network Master is able to connect to it.</td>
</tr>
</tbody>
</table>

Files must be located in one of the locations described in the table above - or a in a sub-directory of one of these.

### 2.6.1 MMEMory:LOAD

**Syntax**

```
MMEMory:LOAD <file>
```

**Description**

This command loads a file into the currently selected application server. The file may contain settings only or both settings and results data.

**Parameter**

```
<file> = <STRING PROGRAM DATA>
```

The path and name of the file to be loaded.

**Response**

None

**Example**

```
MMEM:LOAD "Internal/otdr-settings.cfg"
```

**Note**

There must be a connected application server for this command to be recognized as a legal command.

The application server must be in the idle state and the content of the loaded file must match the application server type.

The files stored in `Internal/` directory or in its subdirectory can be loaded.

When USB storage device is attached to Network Masterset Usb/ directory to the file path to access the files.

### 2.6.2 MMEMory:STORe:STATe

**Syntax**

```
MMEMory:STORe:STATe <file>
```

**Description**

This command stores the current settings to a file on the instrument.

**Parameter**

```
<file> = <STRING PROGRAM DATA>
```

The path and name of the file to store the data.

**Response**

None

**Example**

```
MMEM:STOR:STAT "Internal/my-otdr-settings.cfg"
```

**Note**

There must be a connected application server for this command to be recognized as a legal command.

The application server must be in the idle state.

When USB storage device is attached to Network Masterset Usb/ directory to the file path to access the files.
### 2.6.3 MMEMory:STORe:DATA

**Syntax**: `MMEMory:STORe:DATA <file>`

**Description**: This command stores the current settings and result data to a file on the instrument.

**Parameter**: `<file> = <STRING PROGRAM DATA>`
The path and name of the file to store the data.

**Response**: None

**Example**: `MMEM:STOR:DATA "Usb/my-otdr-trace.sor"`

**Note**: There must be a connected application server for this command to be recognized as a legal command.
The application server must be in the idle state.
When USB storage device is attached to Network Masterset Usb/ directory to the file path to access the files.

### 2.6.4 MMEMory:DELete

**Syntax**: `MMEMory:DELete <file>`

**Description**: This command deletes a file.

**Parameter**: `<file> = <STRING PROGRAM DATA>`
The path to the file to be deleted.

**Response**: None.

**Example**: `MMEM:DEL "Internal/report.pdf"`

### 2.6.5 MMEMory:DATA?

**Syntax**: `MMEMory:DATA? <file>`

**Description**: This command retrieves a file.

**Parameter**: `<file> = <STRING PROGRAM DATA>`
The path to the file to be retrieved.

**Response**: `<DEFINITE LENGTH ARBITRARY BLOCK RESPONSE DATA> = \#<nonzero digit><digits><8 bit data bytes>, where:
<nonzero digit> is a single ASCII character in the range of '1'-'9'. It represents the length of
<digits> in number of bytes.
<digits> is a number of ASCII characters in the range of '0'-'9', which together are a decimal
representation of the number of succeeding data bytes.`

**Example**: `MMEM:DATA? "Internal/report.pdf" → \#49137<9137 bytes of binary data>`

**Note**: This command cannot be used together with other commands in a compound command.

### 2.6.6 MMEMory:COPY

**Syntax**: `MMEMory:COPY <source-file>,<destination-file>`

**Description**: This command copies a file.

**Parameter**: `<source-file> = <STRING PROGRAM DATA>`
The path to the file to be copied.

`<destination-file> = <STRING PROGRAM DATA>`
The path to the new file.

**Response**: None.

**Example**: `MMEM:COPY "Internal/report.pdf","Usb/report.pdf"`

**Note**
### 2.6.7 MMEMory:MOVE

<table>
<thead>
<tr>
<th>Syntax</th>
<th>MMEMory:MOVE &lt;old-file&gt;,&lt;new-file&gt;</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>This command moves or renames a file.</td>
</tr>
<tr>
<td>Parameter</td>
<td>&lt;old-file&gt; = &lt;STRING PROGRAM DATA&gt;</td>
</tr>
<tr>
<td></td>
<td>The path to the file to be moved or renamed.</td>
</tr>
<tr>
<td></td>
<td>&lt;new-file&gt; = &lt;STRING PROGRAM DATA&gt;</td>
</tr>
<tr>
<td></td>
<td>The new path to the file.</td>
</tr>
<tr>
<td>Response</td>
<td>None.</td>
</tr>
<tr>
<td>Example</td>
<td>MMEM:MOVE &quot;Internal/report.pdf&quot;,&quot;Usb/report.pdf&quot;</td>
</tr>
</tbody>
</table>

### 2.6.8 MMEMory:INFO?

<table>
<thead>
<tr>
<th>Syntax</th>
<th>MMEMory:INFO? &lt;file&gt;</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>This command retrieves information about a file.</td>
</tr>
<tr>
<td>Parameter</td>
<td>&lt;file&gt; = &lt;STRING PROGRAM DATA&gt;</td>
</tr>
<tr>
<td></td>
<td>The path to the file to retrieve file information about.</td>
</tr>
<tr>
<td>Response</td>
<td>&lt;file-date-time&gt; = &lt;STRING RESPONSE DATA&gt;</td>
</tr>
<tr>
<td></td>
<td>Last file modification date.</td>
</tr>
<tr>
<td></td>
<td>&lt;file-size&gt; = &lt;NR1 NUMERIC RESPONSE DATA&gt;</td>
</tr>
<tr>
<td></td>
<td>The file size in bytes.</td>
</tr>
</tbody>
</table>

### 2.6.9 MMEMory:CATalog?

<table>
<thead>
<tr>
<th>Syntax</th>
<th>MMEMory:CATalog? &lt;directory&gt;,&lt;pattern&gt;</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>This command lists the files present in a directory.</td>
</tr>
<tr>
<td>Parameters</td>
<td>&lt;directory&gt; = &lt;STRING PROGRAM DATA&gt;</td>
</tr>
<tr>
<td></td>
<td>The path to the directory to be listed.</td>
</tr>
<tr>
<td></td>
<td>&lt;pattern&gt; = &lt;STRING PROGRAM DATA&gt;</td>
</tr>
<tr>
<td></td>
<td>An optional case sensitive file name pattern.</td>
</tr>
<tr>
<td></td>
<td>Wildcard characters are * and ?.</td>
</tr>
<tr>
<td>Response</td>
<td>{{&lt;item&gt;} + {,}+} = &lt;EXPRESSION RESPONSE DATA&gt;</td>
</tr>
<tr>
<td></td>
<td>A list of quoted file and directory names.</td>
</tr>
<tr>
<td>Example</td>
<td>MMEM:CAT? &quot;Internal/reports&quot; → (&quot;report.pdf&quot;,&quot;setup.cfg&quot;)</td>
</tr>
</tbody>
</table>

### 2.6.10 MMEMory:DCATalog?

<table>
<thead>
<tr>
<th>Syntax</th>
<th>MMEMory:DCATalog? &lt;directory&gt;</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>This command lists the sub-directories present in a directory.</td>
</tr>
<tr>
<td>Parameter</td>
<td>&lt;directory&gt; = &lt;STRING PROGRAM DATA&gt;</td>
</tr>
<tr>
<td></td>
<td>The path to the directory to be listed.</td>
</tr>
<tr>
<td>Response</td>
<td>{{&lt;directory&gt;} + {,}+} = &lt;EXPRESSION RESPONSE DATA&gt;</td>
</tr>
<tr>
<td></td>
<td>A list for quoted directory names.</td>
</tr>
<tr>
<td>Example</td>
<td>MMEM:DCAT? &quot;Internal/&quot; → (&quot;diagnostics&quot;,&quot;favorites&quot;,&quot;logs&quot;,&quot;remote&quot;,&quot;screens&quot;,&quot;windowsinstaller&quot;)</td>
</tr>
</tbody>
</table>

Note
2.6.11 MMEMory:MDIRectory

<table>
<thead>
<tr>
<th>Syntax</th>
<th>MMEMory:MDIRectory &lt;directory&gt;</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>This command makes a new sub-directory.</td>
</tr>
<tr>
<td>Parameter</td>
<td>&lt;directory&gt; = &lt;STRING PROGRAM DATA&gt;</td>
</tr>
<tr>
<td></td>
<td>The path to the directory to be created.</td>
</tr>
<tr>
<td>Response</td>
<td>None.</td>
</tr>
<tr>
<td>Example</td>
<td>MMEM:MDIR &quot;Internal/reports&quot;</td>
</tr>
<tr>
<td>Note</td>
<td></td>
</tr>
</tbody>
</table>

2.6.12 MMEMory:RDIRectory

<table>
<thead>
<tr>
<th>Syntax</th>
<th>MMEMory:RDIRectory &lt;directory&gt;[,&lt;force&gt;]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>This command removes an existing directory.</td>
</tr>
<tr>
<td>Parameter</td>
<td>&lt;directory&gt; = &lt;STRING PROGRAM DATA&gt;</td>
</tr>
<tr>
<td></td>
<td>The path to the directory to be created.</td>
</tr>
<tr>
<td></td>
<td>&lt;force&gt; = &lt;BOOLEAN PROGRAM DATA&gt;</td>
</tr>
<tr>
<td></td>
<td>If set to ON then ALL CONTENTS i.e. files and sub-directories will be deleted.</td>
</tr>
<tr>
<td>Response</td>
<td>None.</td>
</tr>
<tr>
<td>Example</td>
<td>MMEM:RDIR &quot;Internal/reports&quot;</td>
</tr>
<tr>
<td>Note</td>
<td>None.</td>
</tr>
</tbody>
</table>

2.6.13 MMEMory:SAVE

<table>
<thead>
<tr>
<th>Syntax</th>
<th>MMEMory:SAVE &lt;enable&gt;</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>This command enables or disables auto saving configuration when the application is terminated</td>
</tr>
<tr>
<td>Parameter</td>
<td>&lt;enable&gt; = &lt;BOOLEAN PROGRAM DATA&gt;</td>
</tr>
<tr>
<td></td>
<td>DEFault = OFF</td>
</tr>
<tr>
<td>Response</td>
<td>None.</td>
</tr>
<tr>
<td>Example</td>
<td>INST:STAR:LAST OTDR-OTDR,1-PORT1</td>
</tr>
<tr>
<td></td>
<td>MMEMory:SAVE ON</td>
</tr>
<tr>
<td></td>
<td>INST? → 2</td>
</tr>
<tr>
<td></td>
<td>INST::TERM 2</td>
</tr>
<tr>
<td>Note</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Syntax</th>
<th>MMEMory:SAVE?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>This query returns whether or not auto saving configuration enabled when the application is terminated</td>
</tr>
<tr>
<td>Parameter</td>
<td>None.</td>
</tr>
<tr>
<td>Response</td>
<td>&lt;enable&gt; = &lt;NR1 NUMERIC RESPONSE DATA&gt;</td>
</tr>
<tr>
<td>Example</td>
<td>MMEMory:SAVE? → 1</td>
</tr>
<tr>
<td>Note</td>
<td></td>
</tr>
</tbody>
</table>
Chapter 3

Standard OTDR Application Commands

This chapter describes the commands available for the Standard OTDR application. The remote control commands are unavailable for the FTTA and Construction applications.

3.1 Measurement Conditions

3.1.1 OTDR:SOURce:PORT

<table>
<thead>
<tr>
<th>Syntax</th>
<th>OTDR:SOURce:PORT &lt;port&gt;</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>Sets the port for OTDR measurement.</td>
</tr>
<tr>
<td>Parameters</td>
<td>&lt;port&gt; = CHARACTER PROGRAM DATA &gt;</td>
</tr>
<tr>
<td></td>
<td>SM: Single Mode</td>
</tr>
<tr>
<td></td>
<td>MM: Multi Mode</td>
</tr>
<tr>
<td>Response</td>
<td>None.</td>
</tr>
<tr>
<td>Example</td>
<td>otdr:sour:port SM</td>
</tr>
<tr>
<td>Note</td>
<td>The setting of “MM” is valid only for MU100021A. For the MU100023A, specify “SM”. When the wavelength is set to 1310 nm or 1550 nm by the command in section 3.1.4 on page 64, PORT1 is selected as the measurement port. When set to 1650 nm, PORT2 is selected as the measurement port.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Syntax</th>
<th>OTDR:SOURce:PORT?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>Queries the current output port setting for OTDR.</td>
</tr>
<tr>
<td>Parameters</td>
<td>None.</td>
</tr>
<tr>
<td>Response</td>
<td>&lt;port&gt; = CHARACTER RESPONSE DATA &gt;</td>
</tr>
<tr>
<td>Example</td>
<td>otdr:sour:port? → SM</td>
</tr>
<tr>
<td>Note</td>
<td></td>
</tr>
</tbody>
</table>

3.1.2 OTDR:SOURce:TESt

<table>
<thead>
<tr>
<th>Syntax</th>
<th>OTDR:SOURce:TESt &lt;mode&gt;</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>Sets the measurement mode.</td>
</tr>
<tr>
<td>Parameters</td>
<td>&lt;mode&gt; = CHARACTER PROGRAM DATA &gt;</td>
</tr>
<tr>
<td></td>
<td>AUTO: Auto Mode</td>
</tr>
<tr>
<td></td>
<td>MANUAL: Manual Mode</td>
</tr>
<tr>
<td>Response</td>
<td>None</td>
</tr>
<tr>
<td>Example</td>
<td>otdr:sour:test AUTO</td>
</tr>
<tr>
<td>Note</td>
<td></td>
</tr>
</tbody>
</table>
### 3.1.3 OTDR:SOURce:WAVelength:AVAilable?

**Syntax**

```
OTDR:SOURce:WAVelength:AVAilable?
```

**Description**
Queries the list of available wavelengths of the OTDR module.

**Parameters**
None.

**Response**

```
<wavelength> = <NUMERIC RESPONSE DATA>
```

**Example**
```
otdr:sour:wav:ava?
```
→ 1310, 1550

**Note**

Wavelength ALL cannot be selected.

### 3.1.4 OTDR:SOURce:WAVelength

**Syntax**

```
OTDR:SOURce:WAVelength <wavelength>
```

**Description**
Sets the wavelength used for the measurement in nanometer (nm) unit. Settable wavelength depends on the module.

**Parameters**

```
<wavelength> = <NUMERIC PROGRAM DATA>
```

**Response**
None.

**Example**
```
otdr:sour:wav 1310
```

**Note**

Wavelength ALL cannot be selected.

### 3.1.5 OTDR:SOURce:RANge:AVAilable?

**Syntax**

```
OTDR:SOURce:RANge:AVAilable?
```

**Description**
Queries the list of available distance ranges (km) for the current wavelength settings.

**Parameters**
None.

**Response**

```
<range> = <NUMERIC RESPONSE DATA>
```

**Example**
```
otdr:sour:ran:ava?
```
→ 5.0, 10.0, 20.0, 50.0, 100.0, 200.0, 300.0

**Note**

The first wavelength will be returned when Wavelength ALL is set.

### 3.1.6 OTDR:SOURce:RANge

**Syntax**

```
OTDR:SOURce:RANge <range>
```

**Description**
Sets the distance range (km) of the measurement. One of the available distance ranges on the current wavelength settings can be set.

**Parameters**

```
<range> = <NUMERIC PROGRAM DATA>
```

**Response**
None.

**Example**
```
otdr:sour:ran 100
```

**Note**


### 3.1 Measurement Conditions

#### 3.1.7 OTDR:SOURce:RESo:AVAilable?

<table>
<thead>
<tr>
<th>Syntax</th>
<th>OTDR:SOURce:RESo:AVAilable?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>Queries the list of available sampling resolution.</td>
</tr>
<tr>
<td>Parameters</td>
<td>None.</td>
</tr>
<tr>
<td>Response</td>
<td>(&lt;\text{res}&gt; = &lt;\text{CHARACTER RESPONSE DATA}&gt;)</td>
</tr>
<tr>
<td>Example</td>
<td><code>otdr:sour:res:ava? \rightarrow \text{COARSE, MEDIUM, FINE}</code></td>
</tr>
<tr>
<td>Note</td>
<td></td>
</tr>
</tbody>
</table>

#### 3.1.8 OTDR:SOURce:RESo

<table>
<thead>
<tr>
<th>Syntax</th>
<th>OTDR:SOURce:RESo (&lt;\text{res}&gt;)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>Sets the resolution of the measurement.</td>
</tr>
<tr>
<td>Parameters</td>
<td>(&lt;\text{res}&gt; = &lt;\text{CHARACTER PROGRAM DATA}&gt;)</td>
</tr>
<tr>
<td>Response</td>
<td>None.</td>
</tr>
<tr>
<td>Example</td>
<td><code>otdr:sour:res MEDIUM</code></td>
</tr>
<tr>
<td>Note</td>
<td>Settable pulse width depends on the distance range and resolution.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Syntax</th>
<th>OTDR:SOURce:RESo?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>Queries the current resolution setting.</td>
</tr>
<tr>
<td>Parameters</td>
<td>None.</td>
</tr>
<tr>
<td>Response</td>
<td>(&lt;\text{res}&gt; = &lt;\text{CHARACTER RESPONSE DATA}&gt;)</td>
</tr>
<tr>
<td>Example</td>
<td><code>otdr:sour:res? \rightarrow \text{MEDIUM}</code></td>
</tr>
<tr>
<td>Note</td>
<td></td>
</tr>
</tbody>
</table>

#### 3.1.9 OTDR:SOURce:PULSe:AVAilable?

<table>
<thead>
<tr>
<th>Syntax</th>
<th>OTDR:SOURce:PULSe:AVAilable?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>Queries the list of available pulse width (ns).</td>
</tr>
<tr>
<td>Parameters</td>
<td>None.</td>
</tr>
<tr>
<td>Response</td>
<td>(&lt;\text{width}&gt; = &lt;\text{NUMERIC RESPONSE DATA}&gt;)</td>
</tr>
<tr>
<td>Example</td>
<td><code>otdr:sour:puls:ava? \rightarrow 10, 20, 50, 100</code></td>
</tr>
<tr>
<td>Note</td>
<td></td>
</tr>
</tbody>
</table>

#### 3.1.10 OTDR:SOURce:PULSe

<table>
<thead>
<tr>
<th>Syntax</th>
<th>OTDR:SOURce:PULSe (&lt;\text{width}&gt;)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>Sets current pulse width.</td>
</tr>
<tr>
<td>Parameter</td>
<td>(&lt;\text{width}&gt; = &lt;\text{NUMERIC PROGRAM DATA}&gt;)</td>
</tr>
<tr>
<td>Response</td>
<td>None.</td>
</tr>
<tr>
<td>Example</td>
<td><code>otdr:sour:puls 100</code></td>
</tr>
<tr>
<td>Note</td>
<td>Settable pulse width depends on the distance range and resolution.</td>
</tr>
</tbody>
</table>
### Chapter 3 Standard OTDR Application Commands

<table>
<thead>
<tr>
<th><strong>Syntax</strong></th>
<th>OTDR:SOURce:PULSe?</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Description</strong></td>
<td>Queries the current pulse width (ns) setting.</td>
</tr>
<tr>
<td><strong>Parameter</strong></td>
<td>None.</td>
</tr>
<tr>
<td><strong>Response</strong></td>
<td>&lt;width&gt; = &lt;NUMERIC RESPONSE DATA&gt;</td>
</tr>
<tr>
<td><strong>Example</strong></td>
<td>otdr:sour:puls? → 100</td>
</tr>
<tr>
<td><strong>Note</strong></td>
<td></td>
</tr>
</tbody>
</table>

#### 3.1.11 OTDR:SOURce:AVERages:TIMe

<table>
<thead>
<tr>
<th><strong>Syntax</strong></th>
<th>OTDR:SOURce:AVERages:TIMe &lt;time&gt;</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Description</strong></td>
<td>Sets the averaging time (s) used in the manual mode.</td>
</tr>
<tr>
<td><strong>Parameter</strong></td>
<td>&lt;time&gt; = &lt;NUMERIC PROGRAM DATA&gt;</td>
</tr>
<tr>
<td><strong>Response</strong></td>
<td>None.</td>
</tr>
<tr>
<td><strong>Example</strong></td>
<td>otdr:sour:aver:tim 120</td>
</tr>
<tr>
<td><strong>Note</strong></td>
<td>During the measurement, this command will be ignored.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Syntax</strong></th>
<th>OTDR:SOURce:AVERages:TIMe?</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Description</strong></td>
<td>Queries the current averaging time setting.</td>
</tr>
<tr>
<td><strong>Parameter</strong></td>
<td>None.</td>
</tr>
<tr>
<td><strong>Response</strong></td>
<td>&lt;time&gt; = &lt;NUMERIC RESPONSE DATA&gt;</td>
</tr>
<tr>
<td><strong>Example</strong></td>
<td>otdr:sour:aver:tim? → 120</td>
</tr>
<tr>
<td><strong>Note</strong></td>
<td></td>
</tr>
</tbody>
</table>
### 3.2 IOR/BSC

#### 3.2.1 OTDR:SENSe:FIBer:IOR

<table>
<thead>
<tr>
<th>Syntax</th>
<th>OTDR:SENSe:FIBer:IOR &lt;ior&gt;</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>Sets index of refraction.</td>
</tr>
<tr>
<td>Parameters</td>
<td>&lt;ior&gt; = &lt;NUMERIC PROGRAM DATA&gt;</td>
</tr>
<tr>
<td>Range:</td>
<td>1.300000 to 1.700000</td>
</tr>
<tr>
<td>Response</td>
<td>None.</td>
</tr>
<tr>
<td>Example</td>
<td>otdr:sens:fib:ior 1.45</td>
</tr>
<tr>
<td>Note</td>
<td>The set value will be applied to the measurement result next time.</td>
</tr>
</tbody>
</table>

#### 3.2.2 OTDR:SENSe:FIBer:BSC

<table>
<thead>
<tr>
<th>Syntax</th>
<th>OTDR:SENSe:FIBer:BSC &lt;bsc&gt;</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>Sets backscatter coefficient of the fiber.</td>
</tr>
<tr>
<td>Parameter</td>
<td>&lt;bsc&gt; = &lt;NUMERIC PROGRAM DATA&gt;</td>
</tr>
<tr>
<td>Range:</td>
<td>-90.00 to -40.00</td>
</tr>
<tr>
<td>Response</td>
<td>None.</td>
</tr>
<tr>
<td>Example</td>
<td>otdr:sens:fib:bsc -83.0</td>
</tr>
<tr>
<td>Note</td>
<td>The set value will be applied to the measurement result next time.</td>
</tr>
</tbody>
</table>

#### Syntax

OTDR:SENSe:FIBer:BSC?

Description: Queries the current backscatter coefficient setting.

Parameter: None.

Example:

```
 otdr:sens:fib:bsc?
```

Note:

```
→ -83.0
```

### Syntax

OTDR:SENSe:FIBer:IOR?

Description: Queries the current index of refraction setting.

Parameter: None.

Example:

```
 otdr:sens:fib:ior?
```

Note:

```
1.450000
```
## 3.3 Splitter

### 3.3.1 OTDR:SOURce:SPLitter

<table>
<thead>
<tr>
<th>Syntax</th>
<th>OTDR:SOURce:SPLitter &lt;number&gt;[,&lt;splitter1&gt;[,&lt;splitter2&gt;[,&lt;splitter3&gt;]]]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>Sets the number of splitters on the fiber and the number of split of each splitter.</td>
</tr>
<tr>
<td>Parameter</td>
<td>&lt;number&gt; = &lt;CHARACTER PROGRAM DATA&gt;</td>
</tr>
<tr>
<td></td>
<td>NONE: No splitter.</td>
</tr>
<tr>
<td></td>
<td>1 to 3: Sets splitters of the specified number.</td>
</tr>
<tr>
<td></td>
<td>DETECT: Detects the number of splitters automatically.</td>
</tr>
<tr>
<td>&lt;splitter1&gt;,</td>
<td>&lt;splitter2&gt;, &lt;splitter3&gt; = &lt;CHARACTER PROGRAM DATA&gt;</td>
</tr>
<tr>
<td></td>
<td>X2: 1×2</td>
</tr>
<tr>
<td></td>
<td>X4: 1×4</td>
</tr>
<tr>
<td></td>
<td>X8: 1×8</td>
</tr>
<tr>
<td></td>
<td>X16: 1×16</td>
</tr>
<tr>
<td></td>
<td>X32: 1×32</td>
</tr>
<tr>
<td></td>
<td>X64: 1×64</td>
</tr>
<tr>
<td></td>
<td>X128: 1×128</td>
</tr>
<tr>
<td></td>
<td>AUTO: 1×?? (Detects the number of splits automatically)</td>
</tr>
<tr>
<td>Response</td>
<td>None.</td>
</tr>
<tr>
<td>Example</td>
<td>otdr:sour:spl NONE</td>
</tr>
<tr>
<td></td>
<td>otdr:sour:spl 1,X2</td>
</tr>
<tr>
<td></td>
<td>otdr:sour:spl 2,X4,X8</td>
</tr>
<tr>
<td></td>
<td>otdr:sour:spl 3,X4,X8,X16</td>
</tr>
<tr>
<td></td>
<td>otdr:sour:spl DETECT</td>
</tr>
</tbody>
</table>

### Note

- **Syntax**
  - OTDR:SOURce:SPLitter?

- **Description**
  - Queries the number of splitters and the number of splits of each splitter.

- **Parameter**
  - None.

- **Response**
  - <number>[,<splitter1>[,<splitter2>[,<splitter3>]]] = <CHARACTER RESPONSE DATA>
  - <number> = <CHARACTER RESPONSE DATA>
  - <splitter1>, <splitter2>, <splitter3> = <CHARACTER RESPONSE DATA>

- **Example**
  - otdr:sour:spl? → NONE
  - otdr:sour:spl? → 1,X2
  - otdr:sour:spl? → 2,X4,X8
  - otdr:sour:spl? → 3,X4,X8,X16
  - otdr:sour:spl? → DETECT

### Note
3.4 Status

3.4.1 OTDR:SENSe:AVERages:TIMe?

<table>
<thead>
<tr>
<th>Syntax</th>
<th>OTDR:SENSe:AVERages:TIMe?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>Queries the elapsed time since the measurement start.</td>
</tr>
<tr>
<td>Parameter</td>
<td>None.</td>
</tr>
<tr>
<td>Response</td>
<td>&lt;time&gt; = &lt;NUMERIC RESPONSE DATA&gt;</td>
</tr>
<tr>
<td>Example</td>
<td>otdr:sens:aver:tim? → 28</td>
</tr>
</tbody>
</table>

Note

3.4.2 OTDR:SENSe:TRACe:READY?

<table>
<thead>
<tr>
<th>Syntax</th>
<th>OTDR:SENSe:TRACe:READY?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>Queries if trace data is ready.</td>
</tr>
<tr>
<td>Parameter</td>
<td>None.</td>
</tr>
<tr>
<td>Response</td>
<td>&lt;status&gt; = &lt;BOOLEAN RESPONSE DATA&gt;</td>
</tr>
<tr>
<td></td>
<td>1 = trace data is ready.</td>
</tr>
<tr>
<td></td>
<td>0 = no trace data available in the memory.</td>
</tr>
<tr>
<td>Example</td>
<td>meas:star</td>
</tr>
<tr>
<td></td>
<td>syst:wait:idle</td>
</tr>
<tr>
<td></td>
<td>otdr:sens:trac:ready? → 1</td>
</tr>
</tbody>
</table>

Note

Use the SYSTem:WAIT:IDLE command to wait for completing the test.
### 3.5 Measurement Functions

#### 3.5.1 OTDR:SENSe:CONCheck

**Syntax**

```
OTDR:SENSe:CONCheck <value>
```

**Description**

Sets whether to perform Connection Check.

**Parameter**

```
<value> = <BOOLEAN PROGRAM DATA>
1 = Performs Connection check.
0 = Does not perform Connection check.
```

**Response**

None.

**Example**

```
otdr:sens:conc 1
```

**Note**

**Syntax**

```
OTDR:SENSe:CONCheck?
```

**Description**

Queries whether to perform Connection Check.

**Parameter**

None.

**Response**

```
<value> = <BOOLEAN RESPONSE DATA>
1 = Performs Connection check.
0 = Does not perform Connection check.
```

**Example**

```
otdr:sens:conc?
```

**Note**

```
→ 1
```

---

#### 3.5.2 OTDR:SENSe:CONState?

**Syntax**

```
OTDR:SENSe:CONState?
```

**Description**

Queries connection check result.

**Parameter**

None.

**Response**

```
<result> = <CHARACTER RESPONSE DATA>
NONE: Connection check has not been performed.
Poor: Connection loss is high.
Fair: Connection loss is low.
Good: Connection loss is very low.
```

**Example**

```
otdr:sens:cons?
```

**Note**

```
→ Good
```

---

#### 3.5.3 OTDR:CONTinue

**Syntax**

```
OTDR:CONTinue
```

**Description**

The measurement resumes after performing the Connection Check.

The measurement can resume when the Connection Check result is Fair or Good.

**Parameters**

None.

**Response**

None.

**Example**

```
otdr:cont
```

**Note**

---

#### 3.5.4 OTDR:SENSe:LIVCheck

**Syntax**

```
OTDR:SENSe:LIVCheck <value>
```

**Description**

Sets whether to perform the live fiber check.

**Parameter**

```
<value> = <BOOLEAN PROGRAM DATA>
1 = Performs the live fiber check.
0 = Does not perform the live fiber check.
```

**Response**

None.

**Example**

```
otdr:sens:livc 1
```

**Note**
### 3.5 Measurement Functions

<table>
<thead>
<tr>
<th>Syntax</th>
<th>OTDR:SENSe:LIVCheck?</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Description</strong></td>
<td>Queries whether to perform the live fiber check.</td>
</tr>
<tr>
<td><strong>Parameter</strong></td>
<td>None.</td>
</tr>
<tr>
<td><strong>Response</strong></td>
<td>(&lt;\text{value}&gt; = \langle\text{BOOLEAN RESPONSE DATA}\rangle)</td>
</tr>
<tr>
<td></td>
<td>1 = Performs the live fiber check.</td>
</tr>
<tr>
<td></td>
<td>0 = Does not perform the live fiber check.</td>
</tr>
<tr>
<td><strong>Example</strong></td>
<td><code>otdr:sens:livc? \rightarrow 1</code></td>
</tr>
<tr>
<td><strong>Note</strong></td>
<td></td>
</tr>
</tbody>
</table>

#### 3.5.5 OTDR:SENSe:FIBCheck

<table>
<thead>
<tr>
<th>Syntax</th>
<th>OTDR:SENSe:FIBCheck (&lt;\text{value}&gt;)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Description</strong></td>
<td>Sets whether to perform fiber length Check.</td>
</tr>
<tr>
<td><strong>Parameters</strong></td>
<td>(&lt;\text{value}&gt; = \langle\text{BOOLEAN PROGRAM DATA}\rangle)</td>
</tr>
<tr>
<td></td>
<td>1 = Performs fiber length check.</td>
</tr>
<tr>
<td></td>
<td>0 = Does not perform fiber length check.</td>
</tr>
<tr>
<td><strong>Response</strong></td>
<td>None.</td>
</tr>
<tr>
<td><strong>Example</strong></td>
<td><code>otdr:sens:fibc 1</code></td>
</tr>
<tr>
<td><strong>Note</strong></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Syntax</th>
<th>OTDR:SENSe:FIBCheck?</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Description</strong></td>
<td>Queries whether to perform fiber length Check.</td>
</tr>
<tr>
<td><strong>Parameters</strong></td>
<td>None.</td>
</tr>
<tr>
<td><strong>Response</strong></td>
<td>(&lt;\text{value}&gt; = \langle\text{BOOLEAN RESPONSE DATA}\rangle)</td>
</tr>
<tr>
<td></td>
<td>1 = Performs fiber length check.</td>
</tr>
<tr>
<td></td>
<td>0 = Does not perform fiber length check.</td>
</tr>
<tr>
<td><strong>Example</strong></td>
<td><code>otdr:sens:fibc? \rightarrow 1</code></td>
</tr>
<tr>
<td><strong>Note</strong></td>
<td></td>
</tr>
</tbody>
</table>
3.6 Analysis

3.6.1 OTDR:SENSe:PATCh:LAUnch

<table>
<thead>
<tr>
<th>Syntax</th>
<th>OTDR:SENSe:PATCh:LAUnch &lt;value&gt;</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>Sets the patch code for the launch fiber.</td>
</tr>
</tbody>
</table>
| Parameter       | <value> = <CHARACTER PROGRAM DATA>
|                 | NONE = Does not set the launch fiber point. |
|                 | EVENT1 = sets the distance of event1 to starting points of the launch fiber. |
|                 | EVENT2 = sets the distance of event2 to starting points of the launch fiber. |
|                 | EVENT3 = sets the distance of event3 to starting points of the launch fiber. |
|                 | <value> = <NR2 NUMERIC PROGRAM DATA>
|                 | sets the start point of the launch fiber by distance. |
|                 | Range: Within the current distance range. |
| Response        | None. |
| Example         | OTDR:SENSe:PATCh:LAUnch 10.0 |
| Note            | |

3.6.2 OTDR:SENSe:PATCh:LAUnch:CORrelation

<table>
<thead>
<tr>
<th>Syntax</th>
<th>OTDR:SENSe:PATCh:LAUnch:CORrelation &lt;value&gt;</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>Set the correlation coefficient for the launch fiber.</td>
</tr>
<tr>
<td>Parameter</td>
<td>&lt;value&gt; = &lt;NR2 NUMERIC PROGRAM DATA&gt;</td>
</tr>
<tr>
<td>Response</td>
<td>None.</td>
</tr>
<tr>
<td>Example</td>
<td>OTDR:SENSe:PATCh:LAUnch:CORrelation 10.0</td>
</tr>
<tr>
<td>Note</td>
<td>This is valid only when the distance is set to the launch fiber of the patch-cord setup.</td>
</tr>
</tbody>
</table>

3.6.3 OTDR:SENSe:PATCh:RECeive

<table>
<thead>
<tr>
<th>Syntax</th>
<th>OTDR:SENSe:PATCh:RECeive &lt;value&gt;</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>Sets the patch code for the receive fiber.</td>
</tr>
</tbody>
</table>
| Parameter       | <value> = <CHARACTER PROGRAM DATA>
|                 | NONE = Does not set the receive fiber point. |
|                 | EVENT1 = sets the distance of event1 to starting points of the receive fiber. |
|                 | EVENT2 = sets the distance of event2 to starting points of the receive fiber. |
|                 | EVENT3 = sets the distance of event3 to starting points of the receive fiber. |
|                 | <value> = <NR2 NUMERIC PROGRAM DATA>
|                 | sets the start point of the receive fiber by distance. |
|                 | Range: Within the current distance range. |
| Response        | None. |
| Example         | OTDR:SENSe:PATCh:RECeive 10.0 |
| Note            | |
### 3.6 Analysis

<table>
<thead>
<tr>
<th>Syntax</th>
<th>OTDR:SENSe:PATCH:RECeive?</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Description</strong></td>
<td>Queries the patch code setup for the receive fiber.</td>
</tr>
<tr>
<td><strong>Parameter</strong></td>
<td>None.</td>
</tr>
</tbody>
</table>
| **Response**     | \(<\text{value}\> = \langle\text{CHARACTER RESPONSE DATA}\rangle\)  
|                  | \(<\text{value}\> = \langle\text{NR2 NUMERIC PROGRAM DATA}\rangle\) |
| **Example**      | OTDR:SENSe:PATCH:RECeive? → EVENT2 |
| **Note**         |                           |

#### 3.6.4 OTDR:SENSe:PATCH:RECeive:CORRelation

<table>
<thead>
<tr>
<th>Syntax</th>
<th>OTDR:SENSe:PATCH:RECeive:CORRelation (&lt;\text{value}&gt;)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Description</strong></td>
<td>Set the correlation coefficient for the receive fiber.</td>
</tr>
<tr>
<td><strong>Parameter</strong></td>
<td>(&lt;\text{value}&gt; = \langle\text{NUMERIC PROGRAM DATA}\rangle) sets the correlation coefficient for the receive fiber.</td>
</tr>
<tr>
<td><strong>Response</strong></td>
<td>None.</td>
</tr>
<tr>
<td><strong>Example</strong></td>
<td>OTDR:SENSe:PATCH:RECeive:CORRelation 10.0</td>
</tr>
<tr>
<td><strong>Note</strong></td>
<td>This is valid only when the distance is set to the receive fiber of the patch-cord setup.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Syntax</th>
<th>OTDR:SENSe:PATCH:RECeive:CORRelation?</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Description</strong></td>
<td>Queries the correlation coefficient for the receive fiber.</td>
</tr>
<tr>
<td><strong>Parameter</strong></td>
<td>None.</td>
</tr>
<tr>
<td><strong>Response</strong></td>
<td>(&lt;\text{value}&gt; = \langle\text{NUMERIC RESPONSE DATA}\rangle)</td>
</tr>
<tr>
<td><strong>Example</strong></td>
<td>OTDR:SENSe:PATCH:RECeive:CORRelation? → 2.0</td>
</tr>
<tr>
<td><strong>Note</strong></td>
<td>This is valid only when the distance is set to the receive fiber of the patch-cord setup.</td>
</tr>
</tbody>
</table>

#### 3.6.5 OTDR:SENSe:ACURsor

<table>
<thead>
<tr>
<th>Syntax</th>
<th>OTDR:SENSe:ACURsor (&lt;\text{value}&gt;)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Description</strong></td>
<td>Sets the cursor A position (km).</td>
</tr>
<tr>
<td><strong>Parameter</strong></td>
<td>(&lt;\text{value}&gt; = \langle\text{NUMERIC PROGRAM DATA}\rangle) Range: Within the range of the setting distance range.</td>
</tr>
<tr>
<td><strong>Response</strong></td>
<td>None.</td>
</tr>
<tr>
<td><strong>Example</strong></td>
<td>otdr:sens:acur 20.5</td>
</tr>
<tr>
<td><strong>Note</strong></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Syntax</th>
<th>OTDR:SENSe:ACURsor?</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Description</strong></td>
<td>Queries the cursor A position (km).</td>
</tr>
<tr>
<td><strong>Parameter</strong></td>
<td>None.</td>
</tr>
<tr>
<td><strong>Response</strong></td>
<td>(&lt;\text{value}&gt; = \langle\text{NUMERIC RESPONSE DATA}\rangle)</td>
</tr>
<tr>
<td><strong>Example</strong></td>
<td>otdr:sens:acur? → 20.5</td>
</tr>
<tr>
<td><strong>Note</strong></td>
<td></td>
</tr>
</tbody>
</table>

#### 3.6.6 OTDR:SENSe:BCURsor

<table>
<thead>
<tr>
<th>Syntax</th>
<th>OTDR:SENSe:BCURsor (&lt;\text{value}&gt;)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Description</strong></td>
<td>Sets the cursor B position (km).</td>
</tr>
<tr>
<td><strong>Parameter</strong></td>
<td>(&lt;\text{value}&gt; = \langle\text{NUMERIC PROGRAM DATA}\rangle) Range: Within the range of the setting distance range.</td>
</tr>
<tr>
<td><strong>Response</strong></td>
<td>None.</td>
</tr>
<tr>
<td><strong>Example</strong></td>
<td>otdr:sens:bcur 20.5</td>
</tr>
<tr>
<td><strong>Note</strong></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Syntax</th>
<th>OTDR:SENSe:BCURsor?</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Description</strong></td>
<td>Queries the cursor B position (km).</td>
</tr>
<tr>
<td><strong>Parameter</strong></td>
<td>None.</td>
</tr>
<tr>
<td><strong>Response</strong></td>
<td>(&lt;\text{value}&gt; = \langle\text{NUMERIC RESPONSE DATA}\rangle)</td>
</tr>
<tr>
<td><strong>Example</strong></td>
<td>otdr:sens:bcur? → 20.5</td>
</tr>
<tr>
<td><strong>Note</strong></td>
<td></td>
</tr>
</tbody>
</table>
## 3.6.7 OTDR:SENSe:LSALeft

<table>
<thead>
<tr>
<th>Syntax</th>
<th>OTDR:SENSe:LSALeft &lt;start&gt;,&lt;stop&gt;</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>Sets start and stop positions (km) for left LSA marker.</td>
</tr>
<tr>
<td>Parameter</td>
<td>&lt;start&gt; = &lt;NUMERIC PROGRAM DATA&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;stop&gt; = &lt;NUMERIC PROGRAM DATA&gt;</td>
</tr>
<tr>
<td></td>
<td>Range: -100.0 to 400.0</td>
</tr>
<tr>
<td>Response</td>
<td>None.</td>
</tr>
<tr>
<td>Example</td>
<td>otdr:sens:lsal 0.0, 0.5</td>
</tr>
<tr>
<td>Note</td>
<td>Start value must be less than stop value. If the set value is out of the distance range, the maximum in the distance range will be set to the marker position.</td>
</tr>
</tbody>
</table>

## 3.6.8 OTDR:SENSe:LSARight

<table>
<thead>
<tr>
<th>Syntax</th>
<th>OTDR:SENSe:LSARight &lt;start&gt;,&lt;stop&gt;</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>Sets start and stop positions (km) for right LSA marker.</td>
</tr>
<tr>
<td>Parameter</td>
<td>&lt;start&gt; = &lt;NUMERIC PROGRAM DATA&gt;</td>
</tr>
<tr>
<td></td>
<td>Range: -100.0 to 400.0</td>
</tr>
<tr>
<td></td>
<td>&lt;stop&gt; = &lt;NUMERIC PROGRAM DATA&gt;</td>
</tr>
<tr>
<td></td>
<td>Range: -100.0 to 400.0</td>
</tr>
<tr>
<td>Response</td>
<td>None.</td>
</tr>
<tr>
<td>Example</td>
<td>otdr:sens:lsar 0.0, 0.5</td>
</tr>
<tr>
<td>Note</td>
<td>Start value must be less than stop value. If the set value is out of the distance range, the maximum in the distance range will be set to the marker position.</td>
</tr>
</tbody>
</table>

## 3.6.9 OTDR:SENSe:LOSS:MODE

<table>
<thead>
<tr>
<th>Syntax</th>
<th>OTDR:SENSe:LOSS:MODE &lt;mode&gt;</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>Sets the Loss Mode.</td>
</tr>
<tr>
<td>Parameters</td>
<td>&lt;mode&gt; = &lt;CHARACTER PROGRAM DATA&gt;</td>
</tr>
<tr>
<td></td>
<td>SPLICE: Splice Loss</td>
</tr>
<tr>
<td></td>
<td>TP: 2-Pt Loss</td>
</tr>
<tr>
<td></td>
<td>TPLSA: 2-Pt LSA</td>
</tr>
<tr>
<td></td>
<td>DBKM: dB/km Loss</td>
</tr>
<tr>
<td></td>
<td>DBKMLSA: dB/km LSA</td>
</tr>
<tr>
<td></td>
<td>TPDBKM: 2-Pt Loss, dB/km</td>
</tr>
<tr>
<td></td>
<td>ORL: ORL</td>
</tr>
<tr>
<td>Response</td>
<td>None.</td>
</tr>
<tr>
<td>Example</td>
<td>otdr:sens:loss:mode SPLICE</td>
</tr>
<tr>
<td>Note</td>
<td></td>
</tr>
</tbody>
</table>

---

74
### 3.6 Analysis

<table>
<thead>
<tr>
<th>Syntax</th>
<th>OTDR:SENSe:LOSS:MODE?</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Description</strong></td>
<td>Queries the current Loss Mode.</td>
</tr>
<tr>
<td><strong>Parameter</strong></td>
<td>None.</td>
</tr>
<tr>
<td><strong>Response</strong></td>
<td><code>&lt;mode&gt;</code> = <code>&lt;CHARACTER RESPONSE DATA&gt;</code></td>
</tr>
<tr>
<td><strong>Example</strong></td>
<td><code>otdr:sens:loss:mode?</code> → SPLICE</td>
</tr>
<tr>
<td><strong>Note</strong></td>
<td></td>
</tr>
</tbody>
</table>

#### 3.6.10 OTDR:SENSe:ORL:MODE

<table>
<thead>
<tr>
<th>Syntax</th>
<th>OTDR:SENSe:ORL:MODE <code>&lt;mode&gt;</code></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Description</strong></td>
<td>Sets ORL Mode.</td>
</tr>
<tr>
<td><strong>Parameters</strong></td>
<td><code>&lt;mode&gt;</code> = <code>&lt;CHARACTER PROGRAM DATA&gt;</code></td>
</tr>
<tr>
<td></td>
<td>ACURSOR: Cursor A.</td>
</tr>
<tr>
<td></td>
<td>ORIGIN: Origin.</td>
</tr>
<tr>
<td></td>
<td>FULL: Full Trace.</td>
</tr>
<tr>
<td><strong>Response</strong></td>
<td>None.</td>
</tr>
<tr>
<td><strong>Example</strong></td>
<td><code>otdr:sens:orl:mode ACURSOR</code></td>
</tr>
<tr>
<td><strong>Note</strong></td>
<td></td>
</tr>
</tbody>
</table>

#### 3.6.11 OTDR:SENSe:ANALyze:PARameters

<table>
<thead>
<tr>
<th>Syntax</th>
<th>OTDR:SENSe:ANALyze:PARameters <code>&lt;spliceloss&gt;,&lt;reflectance&gt;,&lt;endloss&gt;,&lt;macrobend&gt;,&lt;splitter1x2&gt;,&lt;splitter1x4&gt;,&lt;splitter1x8&gt;,&lt;splitter1x16&gt;,&lt;splitter1x32&gt;,&lt;splitter1x64&gt;,&lt;splitter1x128&gt;</code></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Description</strong></td>
<td>Sets thresholds for automatic detection.</td>
</tr>
<tr>
<td><strong>Parameter</strong></td>
<td><code>&lt;spliceloss&gt;</code> = <code>&lt;NUMERIC PROGRAM DATA&gt;</code></td>
</tr>
<tr>
<td></td>
<td>Range: 0.01 to 9.99</td>
</tr>
<tr>
<td></td>
<td><code>&lt;reflectance&gt;</code> = <code>&lt;NUMERIC PROGRAM DATA&gt;</code></td>
</tr>
<tr>
<td></td>
<td>Range: -70.0 to -20.0</td>
</tr>
<tr>
<td></td>
<td><code>&lt;endloss&gt;</code> = <code>&lt;NUMERIC PROGRAM DATA&gt;</code></td>
</tr>
<tr>
<td></td>
<td>Range: 1 to 99</td>
</tr>
<tr>
<td></td>
<td><code>&lt;MacroBend&gt;</code> = <code>&lt;NUMERIC PROGRAM DATA&gt;</code></td>
</tr>
<tr>
<td></td>
<td>Range: 0.3 to 2.0</td>
</tr>
<tr>
<td></td>
<td><code>&lt;SplitterLoss(1x2)&gt;</code> = <code>&lt;NUMERIC PROGRAM DATA&gt;</code></td>
</tr>
<tr>
<td></td>
<td>Range: 1.0 to 30.0</td>
</tr>
<tr>
<td></td>
<td><code>&lt;SplitterLoss(1x4)&gt;</code> = <code>&lt;NUMERIC PROGRAM DATA&gt;</code></td>
</tr>
<tr>
<td></td>
<td>Range: 1.0 to 30.0</td>
</tr>
<tr>
<td></td>
<td><code>&lt;SplitterLoss(1x8)&gt;</code> = <code>&lt;NUMERIC PROGRAM DATA&gt;</code></td>
</tr>
<tr>
<td></td>
<td>Range: 1.0 to 30.0</td>
</tr>
<tr>
<td></td>
<td><code>&lt;SplitterLoss(1x16)&gt;</code> = <code>&lt;NUMERIC PROGRAM DATA&gt;</code></td>
</tr>
<tr>
<td></td>
<td>Range: 1.0 to 30.0</td>
</tr>
<tr>
<td></td>
<td><code>&lt;SplitterLoss(1x32)&gt;</code> = <code>&lt;NUMERIC PROGRAM DATA&gt;</code></td>
</tr>
<tr>
<td></td>
<td>Range: 1.0 to 30.0</td>
</tr>
<tr>
<td></td>
<td><code>&lt;SplitterLoss(1x64)&gt;</code> = <code>&lt;NUMERIC PROGRAM DATA&gt;</code></td>
</tr>
<tr>
<td></td>
<td>Range: 1.0 to 30.0</td>
</tr>
<tr>
<td></td>
<td><code>&lt;SplitterLoss(1x128)&gt;</code> = <code>&lt;NUMERIC PROGRAM DATA&gt;</code></td>
</tr>
<tr>
<td></td>
<td>Range: 1.0 to 30.0</td>
</tr>
<tr>
<td><strong>Response</strong></td>
<td>None.</td>
</tr>
<tr>
<td><strong>Example</strong></td>
<td><code>otdr:sens:anal:par 0.05,-60.0,3,0.3,4.1,7.0,10.0,13.0,16.0,19.0,22.0</code></td>
</tr>
<tr>
<td><strong>Note</strong></td>
<td>The analysis will restart automatically if the thresholds have changed.</td>
</tr>
<tr>
<td>Syntax</td>
<td>OTDR:SENSE:ANALyze:PARameters?</td>
</tr>
<tr>
<td>-----------------</td>
<td>--------------------------------</td>
</tr>
<tr>
<td>Description</td>
<td>Queries the current thresholds setting for the automatic detection.</td>
</tr>
<tr>
<td>Response</td>
<td></td>
</tr>
<tr>
<td>&lt;splice loss&gt;</td>
<td>= &lt;NUMERIC RESPONSE DATA&gt;</td>
</tr>
<tr>
<td>Range: 0.01 to 9.99</td>
<td></td>
</tr>
<tr>
<td>&lt;reflectance&gt;</td>
<td>= &lt;NUMERIC RESPONSE DATA&gt;</td>
</tr>
<tr>
<td>Range: -70.0 to -20.0</td>
<td></td>
</tr>
<tr>
<td>&lt;end loss&gt;</td>
<td>= &lt;NUMERIC RESPONSE DATA&gt;</td>
</tr>
<tr>
<td>Range: 1 to 99</td>
<td></td>
</tr>
<tr>
<td>&lt;Macro Bend&gt;</td>
<td>= &lt;NUMERIC RESPONSE DATA&gt;</td>
</tr>
<tr>
<td>Range: 0.3 to 2.0</td>
<td></td>
</tr>
<tr>
<td>&lt;Splitter Loss(1x2)&gt;</td>
<td>= &lt;NUMERIC RESPONSE DATA&gt;</td>
</tr>
<tr>
<td>Range: 1.0 to 30.0</td>
<td></td>
</tr>
<tr>
<td>&lt;Splitter Loss(1x4)&gt;</td>
<td>= &lt;NUMERIC RESPONSE DATA&gt;</td>
</tr>
<tr>
<td>Range: 1.0 to 30.0</td>
<td></td>
</tr>
<tr>
<td>&lt;Splitter Loss(1x8)&gt;</td>
<td>= &lt;NUMERIC RESPONSE DATA&gt;</td>
</tr>
<tr>
<td>Range: 1.0 to 30.0</td>
<td></td>
</tr>
<tr>
<td>&lt;Splitter Loss(1x16)&gt;</td>
<td>= &lt;NUMERIC RESPONSE DATA&gt;</td>
</tr>
<tr>
<td>Range: 1.0 to 30.0</td>
<td></td>
</tr>
<tr>
<td>&lt;Splitter Loss(1x32)&gt;</td>
<td>= &lt;NUMERIC RESPONSE DATA&gt;</td>
</tr>
<tr>
<td>Range: 1.0 to 30.0</td>
<td></td>
</tr>
<tr>
<td>&lt;Splitter Loss(1x64)&gt;</td>
<td>= &lt;NUMERIC RESPONSE DATA&gt;</td>
</tr>
<tr>
<td>Range: 1.0 to 30.0</td>
<td></td>
</tr>
<tr>
<td>&lt;Splitter Loss(1x128)&gt;</td>
<td>= &lt;NUMERIC RESPONSE DATA&gt;</td>
</tr>
<tr>
<td>Range: 1.0 to 30.0</td>
<td></td>
</tr>
<tr>
<td>Example</td>
<td>otdr:sens:anal:par? \rightarrow 0.05,-60.0,3.0,3.1,7.0,10.0,13.0,16.0,19.0,22.0</td>
</tr>
<tr>
<td>Note</td>
<td></td>
</tr>
</tbody>
</table>
3.7 TRACE

3.7.1 OTDR:TRACe:PARameters?

<table>
<thead>
<tr>
<th>Syntax</th>
<th>OTDR:TRACe:PARameters?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>Queries the main OTDR parameters used to collect the trace data.</td>
</tr>
<tr>
<td>Parameter</td>
<td>None.</td>
</tr>
<tr>
<td>Response</td>
<td></td>
</tr>
<tr>
<td>Example</td>
<td>otdr:trac:par? \rightarrow 1310, 16.415554, 50, 6144, 0.656621, 1.467700, -78.500000</td>
</tr>
<tr>
<td>Note</td>
<td></td>
</tr>
</tbody>
</table>

3.7.2 OTDR:TRACe:ANALyze

<table>
<thead>
<tr>
<th>Syntax</th>
<th>OTDR:TRACe:ANALyze</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>Performs analysis on the trace.</td>
</tr>
<tr>
<td>Parameter</td>
<td>None.</td>
</tr>
<tr>
<td>Response</td>
<td>None.</td>
</tr>
<tr>
<td>Example</td>
<td>otdr:trac:anal</td>
</tr>
<tr>
<td>Note</td>
<td>The analysis cannot be performed during the measurement.</td>
</tr>
</tbody>
</table>

3.7.3 OTDR:TRACe:ANALyze:ORL

<table>
<thead>
<tr>
<th>Syntax</th>
<th>OTDR:TRACe:ANALyze:ORL</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>Performs ORL calculations on the trace.</td>
</tr>
<tr>
<td>Parameter</td>
<td>None.</td>
</tr>
<tr>
<td>Response</td>
<td>None.</td>
</tr>
<tr>
<td>Example</td>
<td>otdr:trac:anal:orl</td>
</tr>
<tr>
<td>Note</td>
<td>The ORL calculations cannot be performed during the measurement.</td>
</tr>
</tbody>
</table>

3.7.4 OTDR:TRACe:MDLOss?

<table>
<thead>
<tr>
<th>Syntax</th>
<th>OTDR:TRACe:MDLOss?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>Queries the trace loss values according to the current loss mode.</td>
</tr>
<tr>
<td>Parameter</td>
<td>None.</td>
</tr>
<tr>
<td>Response</td>
<td></td>
</tr>
<tr>
<td>Note</td>
<td>This query command will be ignored during the measurement.</td>
</tr>
</tbody>
</table>

3.7.5 OTDR:TRACe:EELOss?

<table>
<thead>
<tr>
<th>Syntax</th>
<th>OTDR:TRACe:EELOss?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>Queries trace end-to-end loss.</td>
</tr>
<tr>
<td>Parameter</td>
<td>None.</td>
</tr>
<tr>
<td>Response</td>
<td></td>
</tr>
<tr>
<td>Example</td>
<td>otdr:trac:eelo?    \rightarrow -4.610</td>
</tr>
<tr>
<td>Note</td>
<td>This query command will be ignored during the measurement.</td>
</tr>
</tbody>
</table>
3.7.6 OTDR:TRACe:LOAD:TEXT?

<table>
<thead>
<tr>
<th>Syntax</th>
<th>OTDR:TRACe:LOAD:TEXT? [&lt;start&gt; [,&lt;end&gt;]]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>Queries SOR trace information in the text format.</td>
</tr>
<tr>
<td>Parameter</td>
<td><code>&lt;start&gt;</code> = <code>&lt;NR2 NUMERIC PROGRAM DATA&gt;</code> Range: Within the range of the setting distance range (km). <code>&lt;end&gt;</code> = <code>&lt;NR2 NUMERIC PROGRAM DATA&gt;</code> Range: Within the range of the setting distance range (km).</td>
</tr>
<tr>
<td>Response</td>
<td><code>(&lt;parameters&gt;+,*)) = &lt;EXPRESSION RESPONSE DATA&gt;</code></td>
</tr>
</tbody>
</table>

Example

```
otdr:trac:load:text?  
#6140479 //SCPI data size message for binary data transfer
WL = 1310 nm //Wavelength
FBR = SM //Fiber Type
DR = 5 km //Distance Range
PW = 50 ns //Pulse Width
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

<... Trace Data ...>

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 1.801 dB //dB/km Loss value
Cumulative Loss 5.94 dB //Cumulative loss value
```

Note

This command will be ignored if the measurement is being executed or the measurement results are not ready.

3.7.7 OTDR:TRACe:HOFFset?

<table>
<thead>
<tr>
<th>Syntax</th>
<th>OTDR:TRACe:HOFFset?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>Queries the horizontal offset (km) for the displayed trace(s).</td>
</tr>
<tr>
<td>Parameter</td>
<td>None.</td>
</tr>
<tr>
<td>Response</td>
<td><code>&lt;value&gt; = &lt; NR2 NUMERIC RESPONSE DATA&gt;</code></td>
</tr>
</tbody>
</table>

Example

```
otdr:trac:hoff?  
→ -0.234
```
Chapter 4

OLTS Application Commands

This chapter describes the commands available for the OLTS (Optical Loss Test Set) application. The remote control commands are unavailable for the FTTA and Construction applications.

4.1 Loss Test Set

4.1.1 OLTS:SOURce:POWer:WAVelength:AVAilable?

<table>
<thead>
<tr>
<th>Syntax</th>
<th>OLTS:SOURce:POWer:WAVelength:AVAilable?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>Queries the list of available wavelengths for the light source of the OLTS application.</td>
</tr>
<tr>
<td>Parameters</td>
<td>None.</td>
</tr>
<tr>
<td>Response</td>
<td>&lt;wavelength&gt; = &lt;NUMERIC RESPONSE DATA&gt;</td>
</tr>
<tr>
<td>Example</td>
<td>olts:sour:pow:wav:ava? → 1310, 1550</td>
</tr>
<tr>
<td>Note</td>
<td></td>
</tr>
</tbody>
</table>

4.1.2 OLTS:SOURce:POWer:WAVelength

<table>
<thead>
<tr>
<th>Syntax</th>
<th>OLTS:SOURce:POWer:WAVelength &lt;wavelength&gt;</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>Sets the wavelength used for the measurement in nanometer (nm) unit. Settable wavelength depends on the module.</td>
</tr>
<tr>
<td>Parameters</td>
<td>&lt;wavelength&gt; = &lt;NUMERIC PROGRAM DATA&gt;</td>
</tr>
<tr>
<td>Response</td>
<td>None.</td>
</tr>
<tr>
<td>Example</td>
<td>olts:sour:pow:wav 1310</td>
</tr>
<tr>
<td>Note</td>
<td>Wavelength ALL cannot be selected.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Syntax</th>
<th>OLTS:SOURce:POWer:WAVelength?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>Queries the current wavelength setting.</td>
</tr>
<tr>
<td>Parameter</td>
<td>None.</td>
</tr>
<tr>
<td>Response</td>
<td>&lt;wavelength&gt; = &lt;NUMERIC RESPONSE DATA&gt;</td>
</tr>
<tr>
<td>Example</td>
<td>olts:sour:pow:wav? → 1310</td>
</tr>
<tr>
<td>Note</td>
<td>The first wavelength will be returned when Wavelength ALL is set.</td>
</tr>
</tbody>
</table>

4.1.3 OLTS:SOURce:AM:FREQuency:AVAilable?

<table>
<thead>
<tr>
<th>Syntax</th>
<th>OLTS:SOURce:AM:FREQuency:AVAilable?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>Queries the list of frequencies available for output modes of the light source of the OLTS application.</td>
</tr>
<tr>
<td>Parameters</td>
<td>None.</td>
</tr>
<tr>
<td>Response</td>
<td>&lt;value&gt; = &lt;NUMERIC RESPONSE DATA&gt;</td>
</tr>
<tr>
<td>Example</td>
<td>olts:sour:am:freq:ava? → CW, 270hz, 1khz, 2khz</td>
</tr>
<tr>
<td>Note</td>
<td></td>
</tr>
</tbody>
</table>
4.1.4  OLTS:SOURce:AM:FREQuency

Syntax
OLTS:SOURce:AM:FREQuency <value>

Description
Sets the output mode of the light source.

Parameters
<value> = <NUMERIC PROGRAM DATA>

Response
None.

Example
olt:sour:am:freq CW

Note

Syntax
OLTS:SOURce:AM:FREQuency?

Description
Queries the output mode of the light source.

Parameter
None.

Response
<freq> = <NUMERIC RESPONSE DATA>

Example
olt:sour:am:freq? → CW

Note

4.1.5  OLTS:SOURce:POWer:STATe

Syntax
OLTS:SOURce:POWer:STATe <status>

Description
Turns on or off the light source.

Parameters
<status> = <NUMERIC PROGRAM DATA>
1 = On
0 = Off

Response
None.

Example
olt:sour:pow:stat 1

Note

Syntax
OLTS:SOURce:POWer:STATe?

Description
Queries whether the light source is turned on.

Parameter
None.

Response
<status> = <NUMERIC RESPONSE DATA>
1 = On
0 = Off

Example
olt:sour:pow:stat? → 1

Note

4.1.6  OLTS:SENSe:POWer:WAVelength:AVAilable?

Syntax
OLTS:SENSe:POWer:WAVelength:AVAilable?

Description
Queries the list of available wavelengths for the power meter of the OLTS application.

Parameters
None.

Response
<wavelength> = <NUMERIC RESPONSE DATA>
850, 1300, 1310, 1490, 1550, 1625, 1650

Example
olt:sens:pow:wav:ava?

Note

4.1.7  OLTS:SENSe:POWer:WAVelength

Syntax
OLTS:SENSe:POWer:WAVelength <wavelength>

Description
Sets the wavelength used for the power meter in nanometer (nm) unit.

Parameters
<wavelength> = <NUMERIC PROGRAM DATA>

Response
None.

Example
olt:sens:pow:wav 1310

Note
4.1 Loss Test Set

<table>
<thead>
<tr>
<th>Syntax</th>
<th>OLTS:SENSe:POWer:WAVelength?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>Queries the current wavelength setting of the power meter.</td>
</tr>
<tr>
<td>Parameter</td>
<td>None.</td>
</tr>
<tr>
<td>Response</td>
<td>$&lt;\text{wavelength}&gt; = &lt;\text{NUMERIC RESPONSE DATA}&gt;$</td>
</tr>
<tr>
<td>Example</td>
<td>olts:sens:pow:wav? $\rightarrow$ 1310</td>
</tr>
</tbody>
</table>

**Note**

4.1.8 OLTS:SENSe:CORRection:COLLect:ZERO

<table>
<thead>
<tr>
<th>Syntax</th>
<th>OLTS:SENSe:CORRection:COLLect:ZERO</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>Performs Zero Offset of the power meter.</td>
</tr>
<tr>
<td>Parameters</td>
<td>None.</td>
</tr>
<tr>
<td>Response</td>
<td>None.</td>
</tr>
<tr>
<td>Example</td>
<td>olts:sens:corr:coll:zero</td>
</tr>
</tbody>
</table>

**Note**

<table>
<thead>
<tr>
<th>Syntax</th>
<th>OLTS:SENSe:CORRection:COLLect:ZERO?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>Queries the Zero Offset result of the power meter.</td>
</tr>
<tr>
<td>Parameters</td>
<td>None.</td>
</tr>
<tr>
<td>Response</td>
<td>$&lt;\text{status}&gt; = &lt;\text{CHARACTER RESPONSE DATA}&gt;$ Pass = Passed. Fail = Failed.</td>
</tr>
<tr>
<td>Example</td>
<td>olts:sens:corr:coll:zero? $\rightarrow$ Pass</td>
</tr>
</tbody>
</table>

**Note**

4.1.9 OLTS:SENSe:AVERage:COUNt

<table>
<thead>
<tr>
<th>Syntax</th>
<th>OLTS:SENSe:AVERage:COUNt $&lt;\text{value}&gt;$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>Sets the average count of the power meter.</td>
</tr>
<tr>
<td>Parameters</td>
<td>$&lt;\text{value}&gt; = &lt;\text{NUMERIC PROGRAM DATA}&gt;$</td>
</tr>
<tr>
<td>Response</td>
<td>None.</td>
</tr>
<tr>
<td>Example</td>
<td>olts:sens:aver:coun 1</td>
</tr>
</tbody>
</table>

**Note**

<table>
<thead>
<tr>
<th>Syntax</th>
<th>OLTS:SENSe:AVERage:COUNt?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>Queries the average count of the power meter.</td>
</tr>
<tr>
<td>Parameter</td>
<td>None.</td>
</tr>
<tr>
<td>Response</td>
<td>$&lt;\text{value}&gt; = &lt;\text{NUMERIC RESPONSE DATA}&gt;$</td>
</tr>
<tr>
<td>Example</td>
<td>olts:sens:aver:coun? $\rightarrow$ 1</td>
</tr>
</tbody>
</table>

**Note**

4.1.10 OLTS:SENSe:POWer:REFerence

<table>
<thead>
<tr>
<th>Syntax</th>
<th>OLTS:SENSe:POWer:REFerence $&lt;\text{reference}&gt;$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>Sets the reference value of the power meter in dBm unit.</td>
</tr>
<tr>
<td>Parameters</td>
<td>$&lt;\text{reference}&gt; = &lt;\text{NUMERIC PROGRAM DATA}&gt;$</td>
</tr>
<tr>
<td>Response</td>
<td>None.</td>
</tr>
<tr>
<td>Example</td>
<td>olts:sens:pow:ref -10.5</td>
</tr>
</tbody>
</table>

**Note**

<table>
<thead>
<tr>
<th>Syntax</th>
<th>OLTS:SENSe:POWer:REFerence?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>Queries the reference value of the power meter in dBm unit.</td>
</tr>
<tr>
<td>Parameter</td>
<td>None.</td>
</tr>
<tr>
<td>Response</td>
<td>$&lt;\text{reference}&gt; = &lt;\text{NUMERIC RESPONSE DATA}&gt;$</td>
</tr>
<tr>
<td>Example</td>
<td>olts:sens:pow:ref? $\rightarrow$ -10.50</td>
</tr>
</tbody>
</table>

**Note**
4.1.11 OLTS:SENSe:POWer:THReshold

<table>
<thead>
<tr>
<th>Syntax</th>
<th>OLTS:SENSe:POWer:THReshold &lt;threshold&gt;</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>Sets the threshold for PASS/FAIL judgement of the power meter in dB unit.</td>
</tr>
<tr>
<td>Parameters</td>
<td>&lt;threshold&gt; = &lt;NUMERIC PROGRAM DATA&gt;</td>
</tr>
<tr>
<td>Response</td>
<td>None.</td>
</tr>
<tr>
<td>Example</td>
<td>olt:sens:pow:thr -1.5</td>
</tr>
<tr>
<td>Note</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Syntax</th>
<th>OLTS:SENSe:POWer:THReshold?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>Queries the threshold for PASS/FAIL judgement of the power meter in dB unit.</td>
</tr>
<tr>
<td>Parameter</td>
<td>None.</td>
</tr>
<tr>
<td>Response</td>
<td>&lt;threshold&gt; = &lt;NUMERIC RESPONSE DATA&gt;</td>
</tr>
<tr>
<td>Example</td>
<td>olt:sens:pow:thr? → -1.50</td>
</tr>
<tr>
<td>Note</td>
<td></td>
</tr>
</tbody>
</table>

4.1.12 OLTS:FETCh:POWer

<table>
<thead>
<tr>
<th>Syntax</th>
<th>OLTS:FETCh:POWer?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>Queries the power measured by the power meter in dBm unit.</td>
</tr>
<tr>
<td>Parameter</td>
<td>None.</td>
</tr>
<tr>
<td>Response</td>
<td>&lt;power&gt; = &lt;NUMERIC RESPONSE DATA&gt;</td>
</tr>
<tr>
<td>Example</td>
<td>olt:fetc:pow? → -20.00</td>
</tr>
<tr>
<td>Note</td>
<td></td>
</tr>
</tbody>
</table>
Chapter 5
Measurement

5.1 Measurement Start and Stop

5.1.1 MEASurement:APPLication?

<table>
<thead>
<tr>
<th>Syntax</th>
<th>MEASurement:APPLication?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>This query returns the application server type.</td>
</tr>
<tr>
<td>Parameter</td>
<td>None.</td>
</tr>
<tr>
<td>Response</td>
<td>&lt;application&gt; = &lt;CHARACTER RESPONSE DATA&gt;</td>
</tr>
<tr>
<td></td>
<td>OTDR-OTDR: Standard OTDR application.</td>
</tr>
<tr>
<td></td>
<td>OTDR-OLTS: OLTS application.</td>
</tr>
<tr>
<td>Example</td>
<td>MEAS:APPL? → OTDR-OTDR</td>
</tr>
<tr>
<td>Note</td>
<td></td>
</tr>
</tbody>
</table>

5.1.2 MEASurement:STARt

<table>
<thead>
<tr>
<th>Syntax</th>
<th>MEASurement:STARt</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>This command starts a measurement. It operates the same as pressing the START button on the GUI.</td>
</tr>
<tr>
<td>Parameter</td>
<td>None.</td>
</tr>
<tr>
<td>Response</td>
<td>None.</td>
</tr>
<tr>
<td>Example</td>
<td>MEAS:STAR</td>
</tr>
<tr>
<td>Note</td>
<td></td>
</tr>
</tbody>
</table>

5.1.3 MEASurement:STOP

<table>
<thead>
<tr>
<th>Syntax</th>
<th>MEASurement:STOP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>This command stops an ongoing measurement. It operates the same as pressing the STOP button on the GUI.</td>
</tr>
<tr>
<td>Parameter</td>
<td>None.</td>
</tr>
<tr>
<td>Response</td>
<td>None.</td>
</tr>
<tr>
<td>Example</td>
<td>MEAS:STOP</td>
</tr>
<tr>
<td>Note</td>
<td></td>
</tr>
</tbody>
</table>

5.1.4 MEASurement:RESult:SUMMary?

<table>
<thead>
<tr>
<th>Syntax</th>
<th>MEASurement:RESult:SUMMary?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>Queries the result of the measurement.</td>
</tr>
<tr>
<td>Parameter</td>
<td>None.</td>
</tr>
<tr>
<td>Response</td>
<td>&lt;summary&gt; = &lt;CHARACTER RESPONSE DATA&gt;</td>
</tr>
<tr>
<td></td>
<td>PASS = Passed.</td>
</tr>
<tr>
<td></td>
<td>FAIL = Failed.</td>
</tr>
<tr>
<td>Example</td>
<td>MEAS:RES:SUMM? → PASS</td>
</tr>
<tr>
<td>Errors</td>
<td></td>
</tr>
</tbody>
</table>
Appendix A

Example Scripts

This chapter shows various example scripts for all interfaces which are remote controllable.

A.1 Hints

To ensure that the instrument always start from a well defined state, it is in general a good idea to begin all scripts with the following command. It will terminate all application servers (virtual instruments).

```plaintext
*RST
```
A.2 OTDR Measurement

This example runs an OTDR measurement. It requires to connect a single mode optical fiber to SM port. This example performs an auto mode test with wavelength of 1310 nm and stores the trace data. When the connection check is enabled and its result is Poor (Failed), the file is not saved and an error occurs.

```
*RST
INST:STAR OTDR-OTDR,1-PORT1
SYST:WAIT:IDLE

OTDR:SOUR:PORT SM
OTDR:SOUR:TES AUTO
OTDR:SOUR:WAV 1310

MEAS:STAR
SYST:WAIT:IDLE

OTDR:SENS:TRAC:READY?
MMEM:STOR:DATA "Usb/my-otdr-trace.sor"

SYST:ERR?
INST:TERM
```
A.3 Sending an HTTP GET request

This example sends an HTTP GET request and receives a response.

<table>
<thead>
<tr>
<th>Input script</th>
</tr>
</thead>
<tbody>
<tr>
<td>SYST:NCON:COM:AUTH:TYPE BASIC</td>
</tr>
<tr>
<td>SYST:NCON:COM:BAS &quot;id&quot;,&quot;password&quot;</td>
</tr>
<tr>
<td>SYST:NCON:COM:SET &quot;<a href="https://sample.net/config?argument=sample">https://sample.net/config?argument=sample</a>&quot;</td>
</tr>
<tr>
<td>SYST:NCON:COM:LRET:RST?</td>
</tr>
<tr>
<td>SYST:WAIT:DUR 5</td>
</tr>
<tr>
<td>SYST:NCON:COM:LRET:RST?</td>
</tr>
<tr>
<td>SYST:NCON:COM:LRET:ERR?</td>
</tr>
<tr>
<td>SYST:NCON:COM:LRET:RRES?</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Output</th>
<th>→</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td></td>
</tr>
<tr>
<td>0,&quot;No error.&quot;</td>
<td></td>
</tr>
<tr>
<td>{&quot;key&quot;,&quot;Value&quot;}</td>
<td></td>
</tr>
</tbody>
</table>
A.4 Setting parameters for uploading scenario results

This example sets parameters for uploading scenario results.

```
SYST:NCON:COM:AUTH:TYPE BASIC
SYST:NCON:COM:BAS "id","password"
SYST:NCON:COM:TUR "https://sample.net/upload"
```