# BERTWave Series
# Remote Control
# Operation Manual

**Second Edition**

- **For safety and warning information, please read this manual before attempting to use the equipment.**
- **Additional safety and warning information is provided within the MP2100A/MP2101A/MP2102A BERTWave Operation Manual or MP2100B BERTWave Operation Manual. Please also refer to this document before using the equipment.**
- **Keep this manual with the equipment.**

**ANRITSU CORPORATION**

Document No.: M-W3773AE-2.0

# Safety Symbols

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

## Symbols used in manual

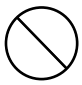
**⚠ DANGER** This indicates a very dangerous procedure that could result in serious injury or death if not performed properly.

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

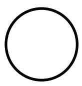
**⚠ CAUTION** This indicates a hazardous procedure or danger that could result in light-to-severe injury, or loss related to equipment malfunction, if proper precautions are not taken.

## Safety Symbols Used on Equipment and in Manual

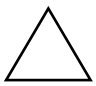
The following safety symbols are used inside or on the equipment near operation locations to provide information about safety items and operation precautions. Ensure that you clearly understand the meanings of the symbols and take the necessary precautions BEFORE using the equipment.



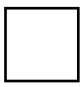
This indicates a prohibited operation. The prohibited operation is indicated symbolically in or near the barred circle.



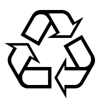
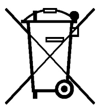
This indicates an obligatory safety precaution. The obligatory operation is indicated symbolically in or near the circle.



This indicates a warning or caution. The contents are indicated symbolically in or near the triangle.



This indicates a note. The contents are described in the box.

These indicate that the marked part should be recycled.

BERTWave Series
Remote Control Operation Manual

4 September 2015 (First Edition)
30 November 2015 (Second Edition)

## Notes On Export Management

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

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

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

# About This Manual

The manual set for the BERTWave Series consists of the following five operation manuals:

**MP2100B BERTWave Operation Manual Operation (M-W3772AE)**

This manual is intended for those who use the MP2100B BERTWave, and explains the installation procedure, handling precautions, connector connection procedures, panel operations, maintenance procedures, specifications and various functions.

**MP2100A/MP2101A/MP2102A BERTWave Operation Manual Operation (M-W3349AE)**

This manual is intended for those who use the MP2100A/MP2101A/MP2102A BERTWave, and explains the installation procedure, handling precautions, connector connection procedures, panel operations, maintenance procedures, specifications and various functions.

**BERTWave series Remote Control Operation Manual (M-W3773AE) (This Manual)**

This manual explains the commands to control the BERTWave, status register structure, and sample programs.

**MX210001A Jitter Analysis Software Operation Manual (M-W3569AE)**

This manual explains the operation method and remote control commands for the MX210001A Jitter Analysis Software.

**MX210002A Transmission Analysis Software Operation Manual (M-W3571AE)**

This manual explains the operation method and remote control commands for the MX210002A Transmission Analysis Software.

This manual explains the remote control commands.

This manual assumes the reader has the following information:

- The reader has read through the BERTWave Operation Manual Operation.
- The reader can create the C or Basic program.

For the connection of the power source and peripheral devices, panel operation, and maintenance, refer to the following manual:
**MP2100B BERTWave Operation Manual Operation**
**MP2100A/MP2101A/MP2102A BERTWave Operation Manual Operation**

# Table of Contents



1
2
3
Appendix
Index

# Table of Messages

# Chapter 1 Overview

This chapter explains the outline of the remote control, main uses, and glossary.

## 1.1 What is Remote Control?

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

The MP2100A/MP2101A/MP2102A/MP2100B (hereafter, BERTWave) supports the Ethernet interface. (When the Option 030 is installed, the GPIB interface can be used.)
When using either interface, set the number to distinguish the BERTWave from other equipment. This number must be an IP address and TCP port number when using the Ethernet interface, or the GPIB interface when using GPIB connection.

The character strings for controlling the BERTWave are called “command”. The command is composed of the ASCII character strings. For example, the following command sets when the signal of the pulse pattern generator (hereafter, PPG) is output to the connector.

```
:OUTput:DATA:OUTput ON
```

A command for reading data from the BERTWave is called “query message”. A query command has the question symbol (?) appended to the string. For example, sending the following command queries the PPG bit rate set at the instrument.

```
:OUTput:BITRate?
```

The controller PC receives the following response against the query message from the instrument.

```
1250000
```

The bit rate is 1250000 kbit/s.

When the BERTWave is measured via remote control, the Remote lamp on the screen is lit. Only the power switch and the key [Local/Panel Unlock] on the system menu are valid in this situation. This situation is called panel lock. To unlock the panel, touch [Local/Panel Unlock] on the system menu.

## 1.2 Main Uses for Remote Control

The main uses for remote control are:

Automation of measurement

To control measuring instruments by executing a program, instead of touch-panel operations. Measurement can be automated by describing the control procedures for controlling the measuring instruments, in the program.

Remote control of instruments

To collect measurement data by controlling measuring instruments installed at remote locations, over communications lines.

Control of multiple measuring instruments

To measure the characteristics of DUTs via the remote control of multiple measuring instruments.

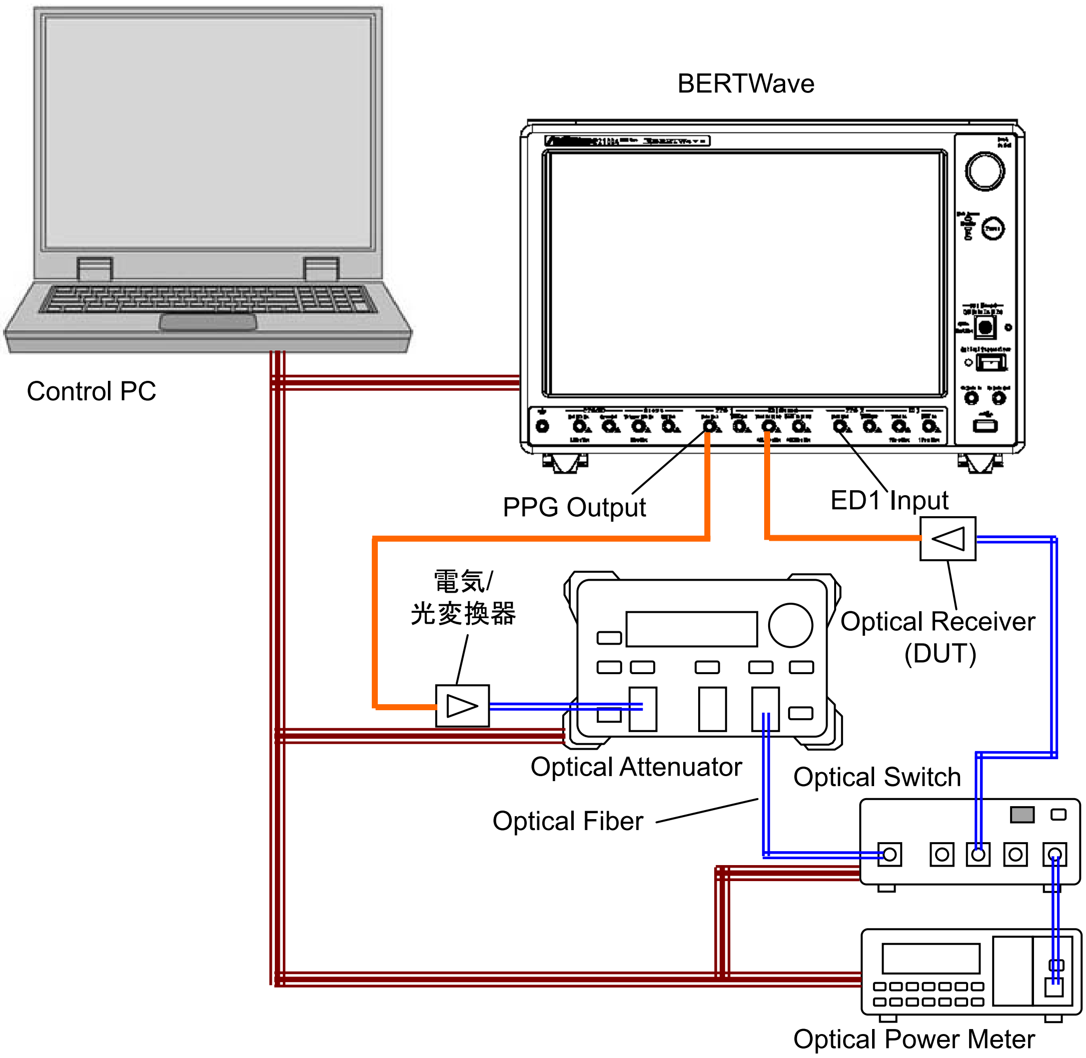



**Figure 1.2-1 Example of Controlling Multiple Instruments**

Figure 1.2-1 shows an example of controlling multiple instruments. In this example, the bit error rates are measured with changes in the optical input level of the optical receiver. On the control PC, set the attenuation of the optical attenuator to read the optical power level measured by the optical power meter, and the bit error rate measured by the BERTWave. Table 1.2-1 shows the measurement result.

**Table 1.2-1 Bit Error Rate of Optical Receiver**

| Optical Power (dBm) | Bit Error Rate |
|---|---|
| –25.034 | 0.011442 |
| –24.523 | 0.0048758 |
| –24.031 | 0.001631 |
| –23.536 | 0.00044241 |
| –23.030 | 0.000078419 |
| –22.523 | 0.0000088616 |
| –22.031 | 0.000000616 |
| –21.524 | 0.000000016 |
| –21.037 | 0.00000000028235 |

## 1.3 Abbreviations

Table 1.3-1 shows the abbreviations used in this operation manual.

**Table 1.3-1 Abbreviations**

| Abbreviation | Formal name |
|---|---|
| ASCII | American Standard Code for Information Interchange |
| CR | Carriage Return |
| ED | Error Detector |
| EOI | End or Identify |
| ESER | Event Status Enable Register |
| ESR | Event Status Register |
| GPIB | General Purpose Interface Bus |
| IEC | International Electrotechnical Commission |
| IEEE | Institute of Electrical and Electronics Engineers |
| LAN | Local Area Network |
| LF | Line Feed |
| MAV | Message Available |
| MSS | Master Summary Status |
| OSER | Operation Status Enable Register |
| OSR | Operation Status Register |
| PPG | Pulse Pattern Generator |
| PC | Personal Computer |
| SCPI | Standard Commands for Programmable Interfaces |
| SRER | Service Request Enable Register |
| SRQ | Service Request |
| STB | Status Byte Register |
| TR | Transition Filter |
| VISA | Virtual Instrument Software Architecture |

# 1.4 Restrictions on Software Versions

Some of the commands described in this manual are only available in a specific version or later of the BERTWave software.

In the **Setup Utility** dialog box, touch [Information], and then check the version of MX210000A.

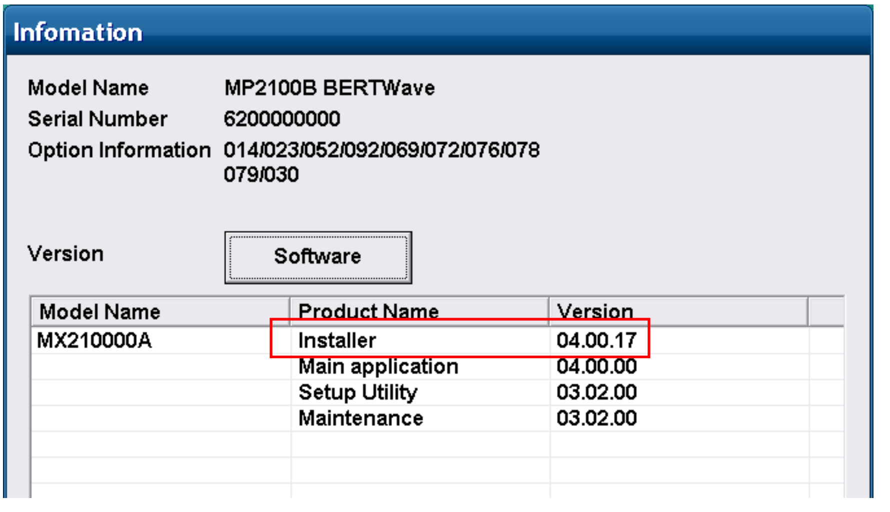



**Figure 1.4-1 Display of Software Version Number**

# Chapter 2 Before Use

This chapter explains the preparations for using remote control.

# 2.1 Required Equipment

The equipment required for remote control is as follows

- Control PC
- Ethernet interface
- Ethernet cable
- GPIB interface (when the Option 030 is installed)
- GPIB cable (when the Option 030 is installed)
- Program development tool

### Ethernet Interface

Prepare Ethernet-compliant interface and cable.

### GPIB Interface

Prepare IEEE 488.2-compliant GPIB interface and cable.

When using the sample program described in Appendix C, VISA is required.

### Program Development Tool

Prepare a tool for developing and running programs for performing remote control. For the requirements specification of the program development tool, refer to the manuals that come with the prepared tool.

### Control PC

Prepare the PC that meets the operating environment for the GPIB interface (when the Option 030 is installed) and program development tools.

# 2.2 Connecting Equipment

## 2.2.1 Connecting Ethernet

Connect the Ethernet connector on the side-panel of the BERTWave and external devices using LAN cables.

Use a LAN crossover cable to connect the BERTWave and a control PC directly. Use a LAN straight cable via a network hub when connecting to multiple external devices.

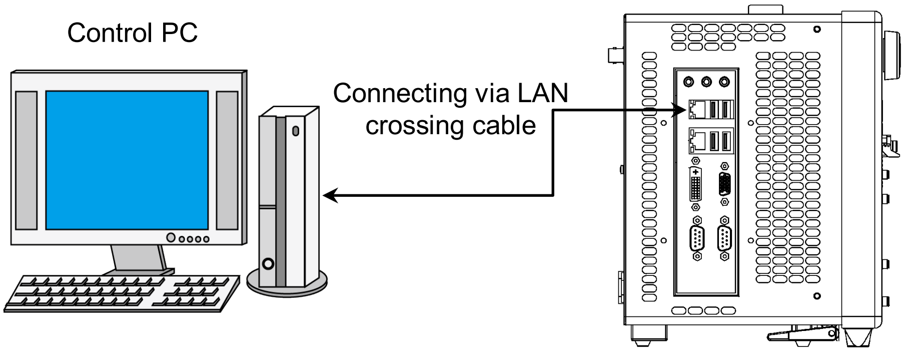



**Figure 2.2.1-1 Direct Connection between BERTWave and Control PC**

***Note:***

When connecting the BERTWave via LAN, confirm the network settings before measurement.

- The IP address of the BERTWave and that of the other devices are not overlapped.
- The IP address of the control PC is included in the address range set at the subnet mask.

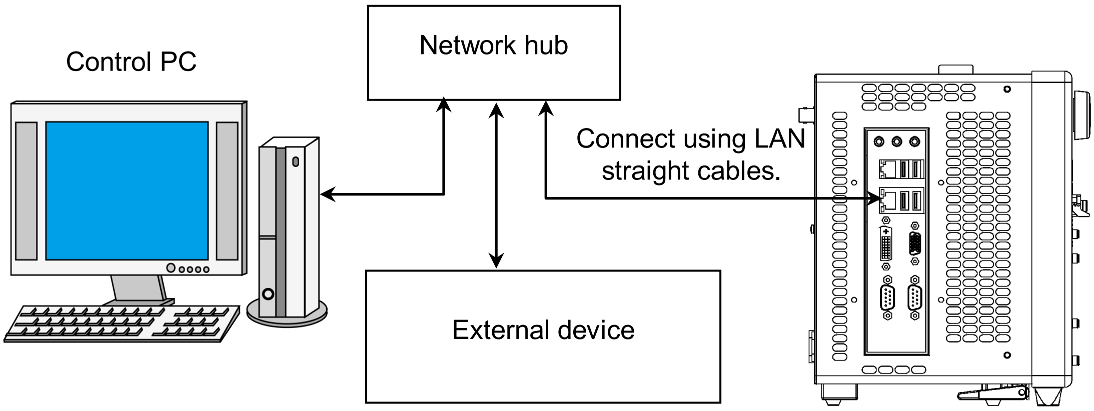



**Figure 2.2.1-2 Sample Connection with Multiple External Devices**

***Note:***

The control PC may have difficulty in communicating with the BERTWave, depending on the status of communications between them. The direct connection is recommended to ensure communication stability.

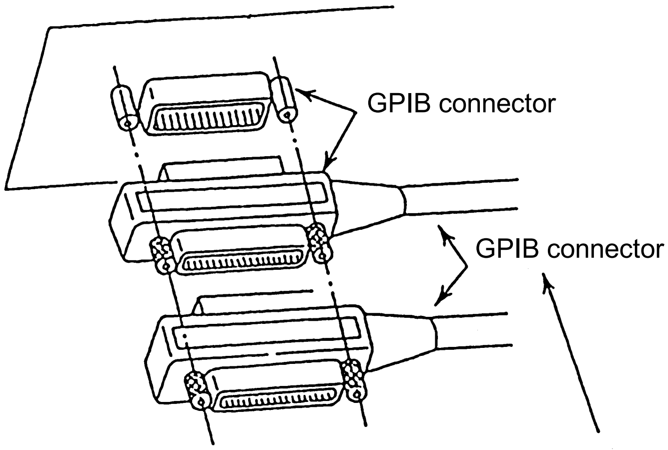
## 2.2.2 Connecting GPIB

Connect the GPIB connector on the rear panel of the BERTWave and an external device using a GPIB cable.

### ⚠ CAUTION

**Always connect the GPIB cable BEFORE turning on the power to the BERTWave. Connecting it while the power is on may damage internal circuits.**

Up to 15 devices, including the external PC controller can be connected to one BERTWave unit. Always follow the conditions shown below when connecting devices.




| | |
|---|---|
| Total cable length: | Up to 20 m |
| Cable length between devices: | Up to 4 m |
| Number of devices that can be connected: | Up to 15 |

**Figure 2.2.2-1 GPIB Cable Connection 1**

Connect cables without forming loops.

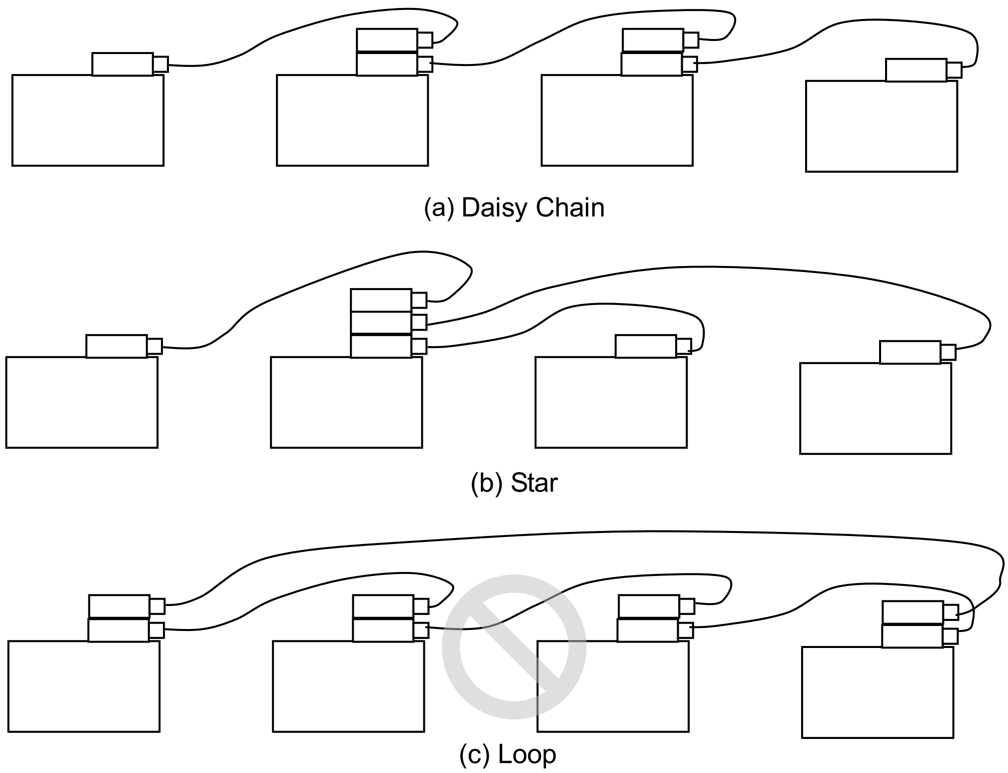



**Figure 2.2.2-2 GPIB Cable Connection 2**

# 2.3 Setting Interface

For details on how to set the interface, refer to Section 2.14 “Setting Interface for Remote Control” in the *MP2100B BERTWave Operation Manual*.

## 2.3.1 Setting Interface

Set the remote control interface to the Ethernet using the following method, and enter the IP address.

1. Switch on the power to the BERTWave.
2. Touch [Setup Utility] at the Selector screen.
3. Touch [Remote Control].
4. Touch the Active Interface button to set the button display to [Ethernet].
   When the Option 030 is not installed, the Active Interface button is disabled.
5. Set the IP address, subnet mask, gateway and port number.
   The gateway address can be omitted.
   The port number can be set from 1024 to 5001.
6. Touch [Apply], and then the settings are completed.
   Touch [Exit], and then the set value is deleted.

***Note:***

Do not set the following IP address.
192.168.1.0 to 192.168.1.255

## 2.3.2 Setting GPIB

Set the remote control interface to the GPIB using the following method, and enter the GPIB address.

1. Switch on the power to the BERTWave.
2. Touch [Setup Utility] at the Selector screen.
3. Touch [Remote Control].
4. Touch the Active Interface button to set the button display to [GPIB].

   The Active Interface button is available only when the Option 030 is installed.
5. Touch the GPIB address on the test box.
6. Set the GPIB address using the numeric value input panel.
   You cannot enter the numeric value in the text box directly using the attached key board.
7. Touch [Apply] to complete the setting.
   Touch [Exit], and then the values set at step 4 to 6 are deleted.

# 2.4 Checking Connection

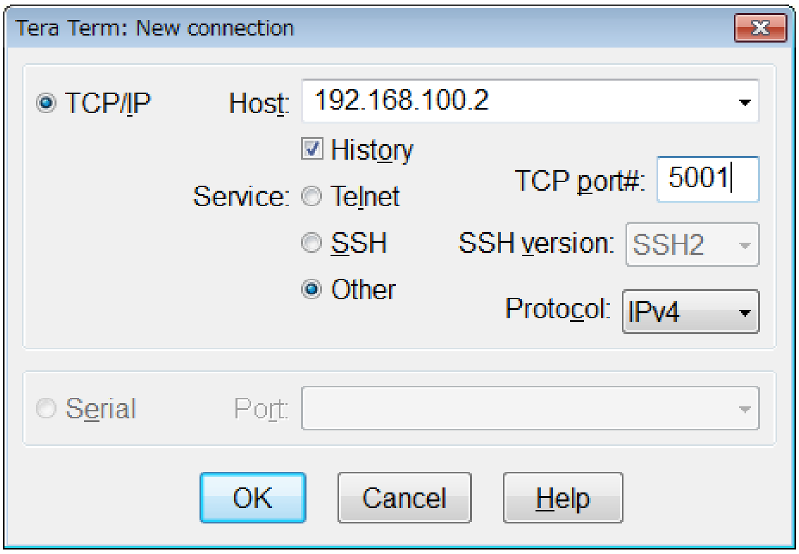
This section describes how to check if the Control PC can recognize the BERTWave

## 2.4.1 When using Ethernet (Windows 7)

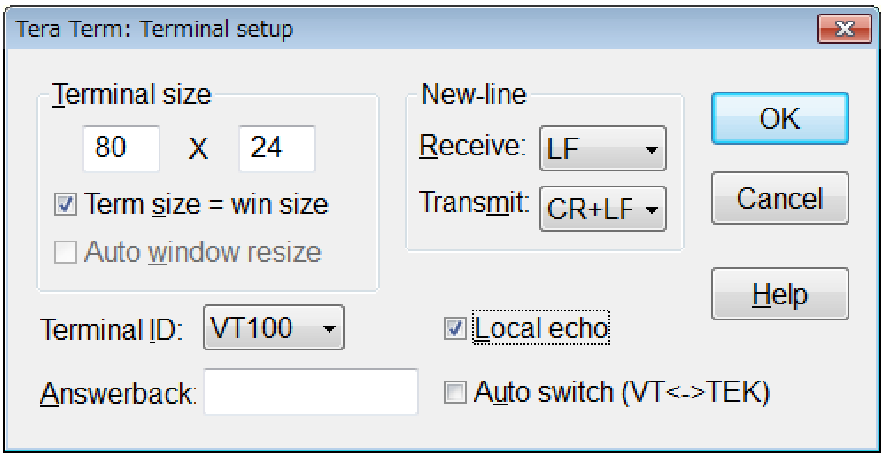
This section explains how to use the free software, Tera Term Version 4.69.

1. When starting Tera Term, the [New connection] window is opened.
   Enter the IP address and TCP port number in the [Host].
   Set the service to [Others] and protocol to [IPv4].
   Click [OK].

   If the BERTWave IP address is set to 192.168.100.2, and the port number is set to 5001, set as follows.




2. When the BERTWave is recognized, the communication window is displayed.
3. Click [Settings (S)] - [Terminal (T)..] on the menu.
4. Set the return cord reception to [LF] and those of counterpart to [CR+LF]. Check the local echo and click [OK].




5. Send *IDN?.
   Confirm that the response is displayed from the BERTWave.

***Note:***

When the panel lock is released by the panel operation, the communication with the BERTWave is terminated and Tera Term will be closed.

## 2.4.2 When using GPIB

1. Install the software drivers for the GPIB interface.
2. Run the software.
   For the operation method, refer to the GPIB interface operation manual.
3. Check the displayed instrument address.

# 2.5 Message Format

## 2.5.1 Message Types

Messages are composed of the character strings indicating message and message end. The character string indicating the message end is LF (Line Feed) or CR (Carriage Return) +LF.

***Note:***

If LF or CR+LF is not attached to the message end, a timeout error occurs because the communication does not end.

Messages are composed of the following types depending on the transmission direction:

#### Program Messages

Messages sent from PC to instrument

There are two types of the program messages:

- Command
  This can be used for measurement condition settings and measurement start.
- Query
  This queries the status and settings of the measuring instrument. When transmitting the query, the instrument creates a response message to the query.

#### Response Messages

Messages sent from instrument to PC controller

## 2.5.2 Message Configuration

The messages are composed of header and data parts separated by more than a half width space. Program messages always have a header but sometimes have no data. Response messages always have data but sometimes have no header.



### Header

The command header has the following types:

- Simple header
  The header is composed of alphanumeric characters and underbars, and the initial character is an alphabetic character.
  Example: `STA`
- Common command header
  The header is composed of alphanumeric characters and underbars, and the initial character is an asterisk (*).
  Example: `*CLS`
- Multiple headers
  Single headers are linked by colons. Colons can be used at the header. Multiple headers can be used to configure layered processing.
  Example: `:SENSE:MEASURE:START`

Queries have a question mark (?) appended to the header.

Example:
```
*ESE?
:CONFIGURE?
```

### Data

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

String data is ASCII code enclosed in quotation marks.
An example of the program message when inputting `Model ANR-005` at the title is shown below.

Example:
```
:SYSYEM:MEMORY:STORE 'Model ANR-005',0,ALL
:SYESEM:MEMORY:STORE "Model ANR-005",0,ALL
```

When quotation marks are included in the character string, paired marks are used.

Example:
```
He said "Good product". → "He said ""Good Product""."
He said 'Good product'. → 'He said ''Good Product''.
'
```

In addition, paired quotation marks can be used inside other paired quotation marks.

Example:

```
He said "Good product". → 'He said "Good Product".'
He said 'Good product'. → "He said 'Good Product'."
```

The numeric values can be described by using numeric data, input numeric values either as decimal, binary, octal, or hexadecimal numbers. When using the binary, octal, or hexadecimal numbers, put #B,#O, or #H before the data.

Example:

```
10   #B1010             #O12        #HA
1550 #B11000001110      #O3016      #H60E
```

When using decimal numbers, use integer number, fixed point, and floating point. The following examples indicate the same values.

Example:

```
-10        -10.00      -1E1
1250       1250.000    1.25E3
0.0023     2.3E-4
```

For the binary data, the head string starts with a sign (#) and continues with data after a numeric value indicating the data length.

Example: `#42002an%*qe4445+\…`

4 digits

2002 bytes binary data

When there are multiple data in a message, separate each of them with commas (,).

Example: `:INPUT:DATA:ATTFACTOR 1,6`
`:SENSE:MEASURE:EALARM:PERIOD 0,0,1,0`

When concatenating multiple program messages, separate the message with semicolons (;).

Example: `:MOD:ID 5;:DISP:MODE EYE;:SAMP:STAT RUN`

***Notes:***

When sending multiple messages separated by semicolons, the maximum length of the concatenated string is 1024 bytes.

When sending a concatenated string of query messages, response messages are separated by semicolons.

Example: `:MOD:ID 1;:OUTP:BITR:STAN?;:OUTP:BITR?`
`>"10G_LAN";10312500`

## 2.5.3 Common Commands

The GPIB specifications (IEEE 488.2) define equipment commands. In this manual, these defined commands are called common commands.

The common commands are divided into mandatory and option commands. The BERTWave supports the common commands listed in Table 2.5.3-1.

**Table 2.5.3-1 Common Commands**

| Command | Explanation |
|---|---|
| *CLS | Clears stand event register and output queue |
| *ESE | Sets and queries standard event enable register |
| *ESR | Queries standard event register |
| *IDN | Queries product information |
| *OPC | Sets/queries bit setting and bit 0 for status byte indicating message processing completion |
| *OPT | Queries option information |
| *RST | Initializes BERTWave setting conditions |
| *SRE | Sets and queries SRER |
| *STB | Queries status byte register |
| *TRG | Starts measurement |
| *WAI | Waits previous sent message completion |

## 2.5.4 Device Dependent Commands

In this manual, commands that differ according to the functions of the measuring instrument are called Device Dependent Commands.

This instrument has two types of Device Dependent Commands.

- SCPI
  Commands meeting SCPI standard
- Native
  Commands consisting of at least three ASCII characters

# 2.6 Checking Instrument Status

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



## 2.6.1 Register Structure

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

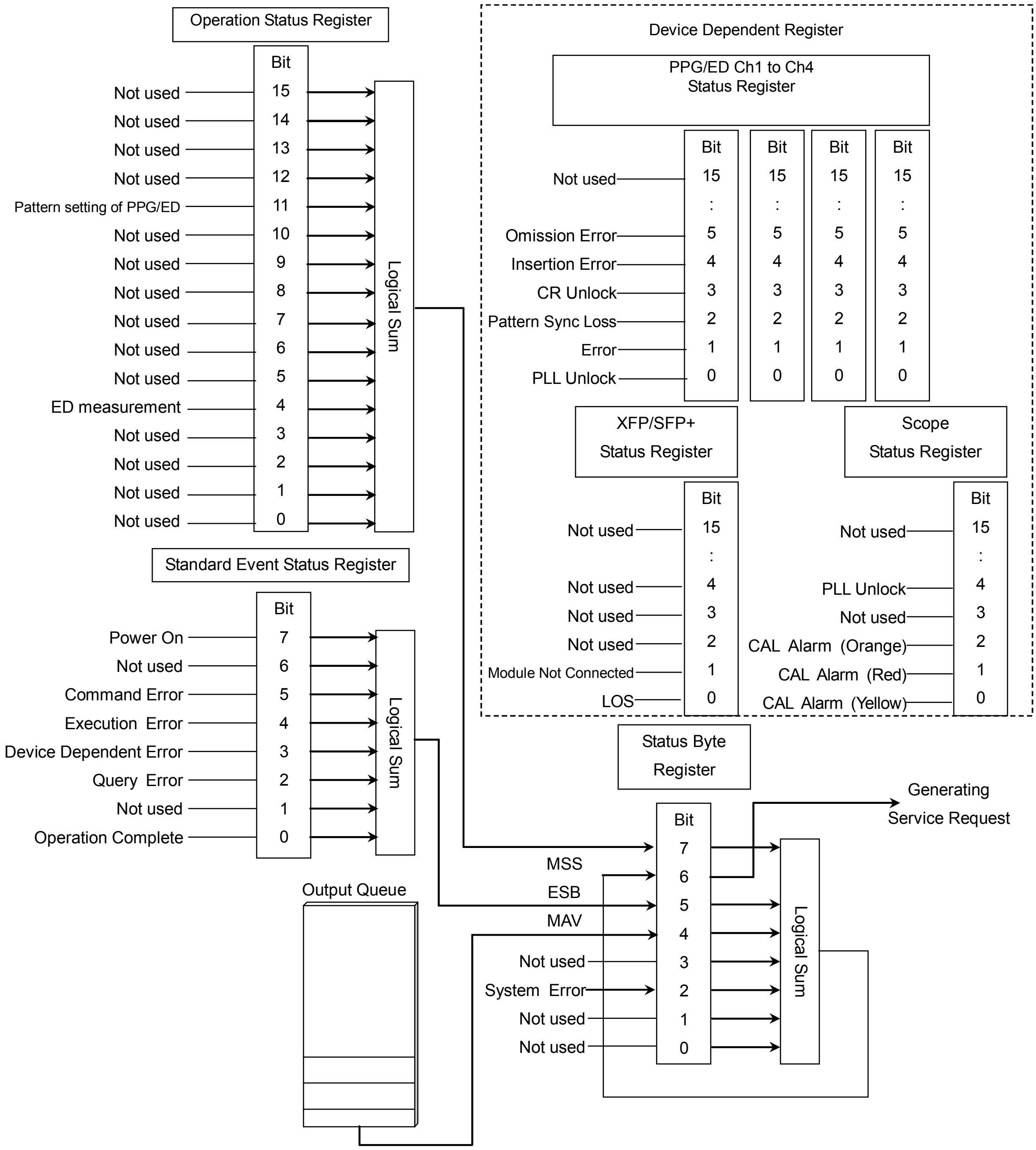



**Figure 2.6.1-1 Register Structure**

Each register uses 8-bit or 16-bit data. The register output values are the decimal totals for each bit shown in Table 2.6.1-1.

**Table 2.6.1-1 Register Bit Decimal Conversion Values**

| Bit | Decimal value | Bit | Decimal value |
|---|---|---|---|
| 0 | 1 | 8 | 256 |
| 1 | 2 | 9 | 512 |
| 2 | 4 | 10 | 1024 |
| 3 | 8 | 11 | 2048 |
| 4 | 16 | 12 | 4096 |
| 5 | 32 | 13 | 8192 |
| 6 | 64 | 14 | 16382 |
| 7 | 128 | 15 | 32764 |

The service request enable register (SRER) has a corresponding status byte register.

## 2.6.2 Status Byte Register

The status byte register (STB) displays the status of equipment defined by the GPIB standards. When the equipment status changes, the value in the STB changes too. It can be used to generate interrupts to the PC controller. These interrupts are called service requests.

There is a service request enable register (SRER) for the STB. The SRER can select the status byte bit generating the service request.

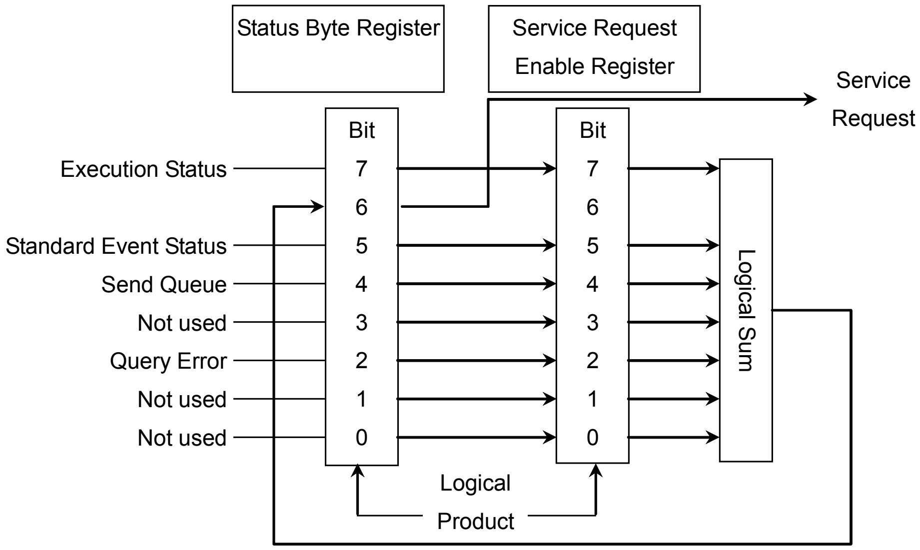



**Figure 2.6.2-1 Configuration of Status Byte Register and Service Request Enable Register**

***Note:***

When using the GPIB interface, the service request is enabled.

The following methods are used to read the status byte register.

- Using common *STB? command
- Using GPIB serial poll (when the Option 030 installed)
  Read the GPIB interface manual for the serial poll method.
  When using serial polling, even if bit 6 is 1, it becomes 0 after reading once.

The `*SRE` and `*SRE?` common commands can be used for setting and reading the SRER for setting reading of the status byte register. To output the STB data, set the bit corresponding to the SRER to 1.

The meaning of each bit of the STB is shown in the following table.

**Table 2.6.2-1 Meaning of Status Byte Register**

| Bit | Explanation |
|---|---|
| 7 | This is the logical sum of each bit of the logical product of the OSR and its event enable register. |
| 6 | MSS (Master Summary Register)<br>It is the logical sum of the bit 5 to 0, bit 7 logical product of the STB and the SRER. |
| 5 | This is the logical sum of each bit of the logical product of the standard event status register and standard event enable register. |
| 4 | MAV (Message Available summary)<br>This is always 1 when there is a response message in the output queue of this instrument |
| 3 | Not used; always 0 |
| 2 | Becomes 1 at System Error |
| 1 | Not used; always 0 |
| 0 | Not used; always 0 |

Bit 7 of the STB indicates information about the OSR.
For details about the information, refer to section 2.6.4 “Operation Status Register”.

Bit 6 of the STB is called the master summary status (MSS) bit. When it is 1, there is a notification from BERTWave to the PC controller. When it changes to 1 from 0, a service request is generated.

Bit 5 of the STB indicates information about the standard status register.
For details about the information, refer to section 2.6.3 “Standard Event Status Register”.

The device dependent register data is not indicated in the STB.

Bits 7 and 5 of the STB can be set to 0 using the `*CLS` common command. When `*CLS` is sent after a command or when a query is sent after `*CLS`, the send queue is cleared and bit 4 is set to 0.

The SRER cannot be set to 0 by `*CLS`, so use `*SRE`.

## 2.6.3 Standard Event Status Register

There is a standard event status enable register (ESE) for the standard event status register (ESR). The logical product of these two registers and the logical sum of each bit of this result is output to bit 5 of the STB.

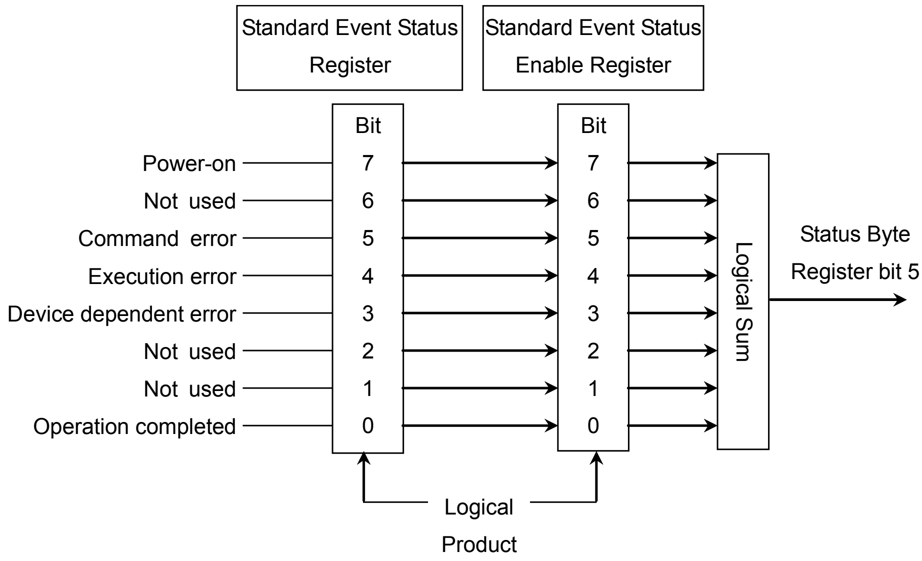



**Figure 2.6.3-1 Configuration of Standard Event Status Register and Standard Event Status Enable Register**

The meaning of each bit of the ESR is listed in the table below.

**Table 2.6.3-1 Meaning of Standard Event Status Register**

| Bit | Explanation |
|---|---|
| 7 | 1 if the BERTWave is powered on. |
| 6 | Not used; always 0 |
| 5 | 1 if a command error occurs.<br>Refer to Appendix B, "Error Codes" for details. |
| 4 | 1if an execution error occurs.<br>Refer to Appendix B, "Error Codes" for details. |
| 3 | 1if a device-dependent error occurs.<br>Refer to Appendix B, "Error Codes" for details. |
| 2 | Not used; always 0 |
| 1 | Not used; always 0 |
| 0 | Operation Complete<br>Becomes 1 when entire command operation completed after *OPC command operation |

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

The ESE can be set and read using the `*ESE` and `*ESE?` commands. To output standard event register data, set the bit corresponding to the enable register to 1.
The bit 0 can be read using the `*OPC` command.
The standard register can be set to 0 using the `*CLS` command.

## 2.6.4 Operation Status Register

The operation status register (OSR) is composed of the following registers:

- Operation status condition register
- Transition filter
- Operation status event register
- Operation status enable register (OSER)

The operation status condition register indicates changes in the status. When the status changes, the value of this register also changes.

The OSER records changes in the value of the execution status condition register. There is a transition filter that defines the write condition before the OSER. The transition filter sets the OSER to 1 under any of the following conditions:

- When bit changes from 0 to 1
- When bit changes from 1 to 0
- When bit changes from 0 to 1 or bit changes from 1 to 0

The OSER sets the OSER output at each bit. The logical product these two registers is obtained and the logical sum of each bit of the result is output at bit 7 of the STB.

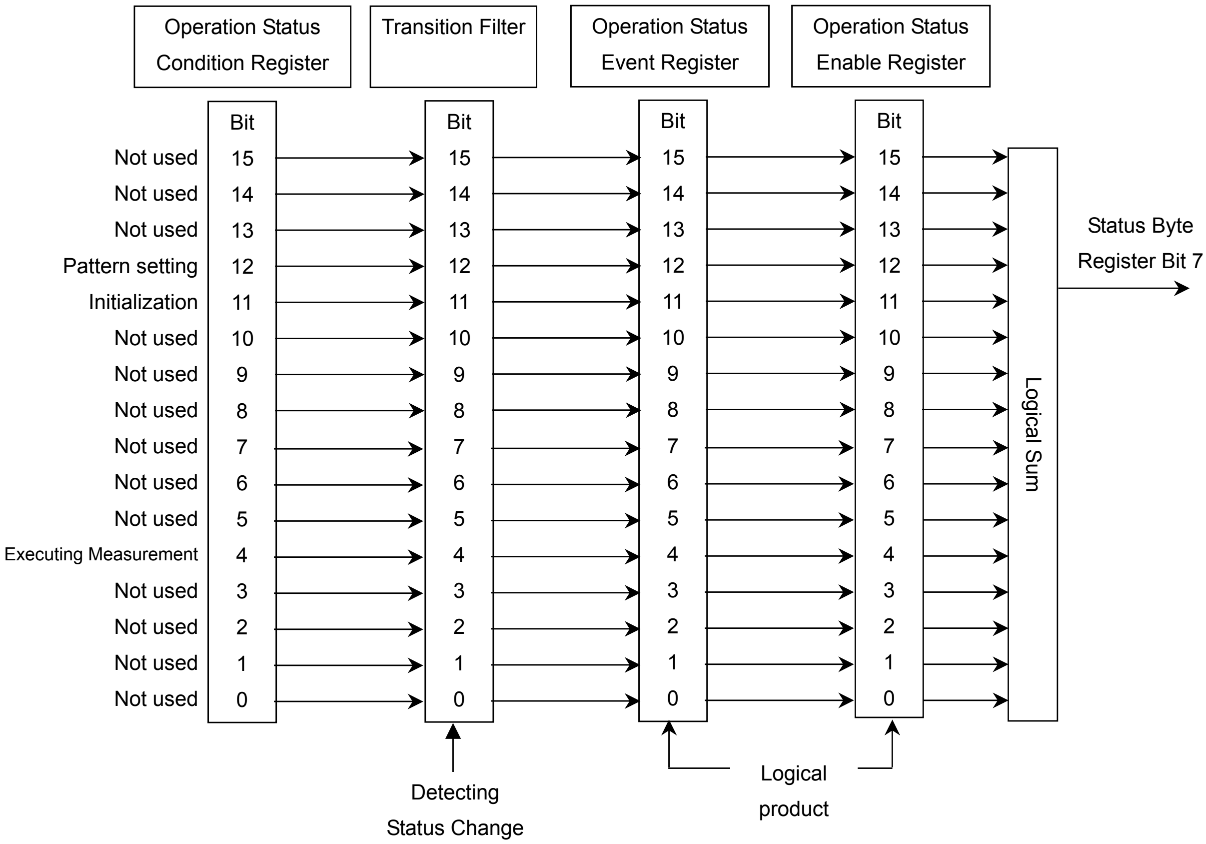



**Figure 2.6.4-1 Configuration of Operation Status Condition Register, Operation Status Event Register, Operation Status Enable Register, and Transition Filter**

Each bit definition of the execution status is as follows.

**Table 2.6.4-1 Bit Definition of Operation Status Register**

| Bit | Explanation |
|---|---|
| 15–12 | Not used; always 0 |
| 11 | 1 during execution of PPG/ED pattern setting. |
| 10–5 | Not used; always 0 |
| 4 | 1 during execution of ED measurement. |
| 3–0 | Not used; always 0 |

The commands for confirming the execution start or end time at the OSR are shown in the following table.

**Table 2.6.4-2 Commands for Confirming Execution of operation at Operation Status Register**

| Operation Status Register Bit | Command |
|---|---|
| 11 | :SENSe:MMEMory:PATTern:RECall<br>:SENSe:PATTern:TYPE<br>:SOURce:MMEMory:PATTern:RECall<br>:SOURce:PATTern:TYPE |
| 4 | [:BERT:ALL]:SENSe:MEASure:STARt<br>[:BERT:ALL]:SENSe:MEASure:STOP |

To detect the execution start, the transition filter bit response to `STATus:OPERation:PTRansition` is set to 1.
To detect the execution end, the transition filter bit response to `:STATus:OPERation:NTRansition`.
The OSER can be read using `:STATus:OPERation:[EVENt]?`. When the register is read, the OSR returns to 0.

The operation status condition register can be read using `:STATus:OPERation:CONDition?`.

To set the OSER, use `:STATus:OPERation:ENBle`. To read the OSER, use `STATus:OPERation:ENBle?`. To output the OSR data, set the bit for the status setting enable register to 1.

When sending `:STATus:OPERation:RESet`, the operation status event register is set to 0.
Even when sending `:STATus:OPERation:RESet`, the OSER is not changed.

## 2.6.5 Device Dependent Registers

The following registers are called the device dependent registers.

- PPG/ED Ch1 to 4 Status Register
- XFP/SFP+ Status Register
- Scope Status Register

The device dependent status register has the same type of condition register, transition filter, and event register. However there is no enable register for switching the output at each bit on/off.

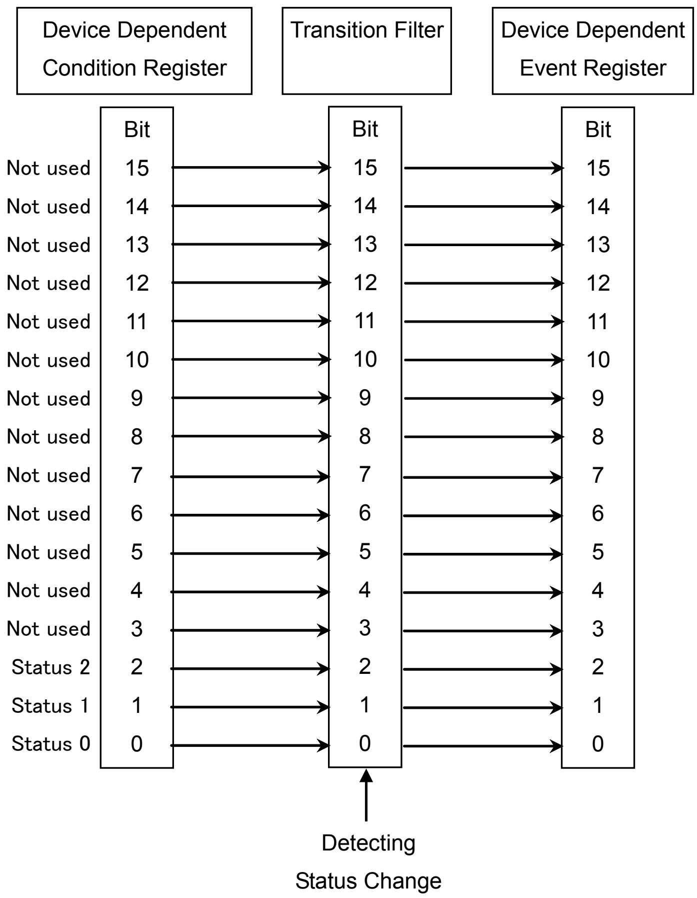



**Figure 2.6.5-1 Configuration of Device Dependent Register**

When the value of the device dependent status register changes, there is no effect on the STB. As a result, a service request is not generated to the PC controller.

Each bit definition of the device dependent register is as follows.

**Table 2.6.5-1 Meaning of PPG/ED Ch1 to Ch4 Status Register**

| Bit | Explanation |
|---|---|
| 15—6 | Not used; always 0 |
| 5 | Indicates Omission Error occurs |
| 4 | Indicates Insertion Error occurs |
| 3 | Indicates CR Unlock occurs |
| 2 | Indicates SYNC Loss occurs |
| 1 | Indicates Bit Error occurs |
| 0 | Indicates PLL Unlock occurs |

**Table 2.6.5-2 Meaning of XFP/SFP+ Status Bit**

| Bit | Explanation |
|---|---|
| 15—2 | Not used; always 0 |
| 1 | Indicates LOS occurs |
| 0 | Indicates Ready status |

**Table 2.6.5-3 Bit Meaning of Scope Status Register**

| Bit | Explanation |
|---|---|
| 15—5 | Not used; always 0 |
| 4 | Indicates PLL Unlock occurs |
| 3 | Not used; always 0 |
| 2 | Indicates CAL alarm (Orange) occurs |
| 1 | Indicates CAL alarm (Red) occurs |
| 0 | Indicates CAL alarm (Yellow) occurs |

To detect the occurrence of these phenomena, set the transition filter bit to 1 using the following commands:

```
:INSTrument:PE{1|2|3|4}:PTRansition
:INSTrument:XSFP:PTRansition
:INSTrument:WAV:PTRansition
```

To detect the end of these phenomena, set the transition filter bit to 1using the following commands:

```
:INSTrument:PE{1|2|3|4}:NTRansition
:INSTrument:XSFP:NTRansition
:INSTrument:WAV:NTRansition
```

The device dependent event register can be read using the following queries:

```
:INSTrument:PE{1|2|3|4}:[EVENt]?
:INSTrument:XSFP:[EVENt]?
:INSTrument:WAV:[EVENt]?
```

The device dependent condition register can be read using the following queries:

```
:INSTrument:PE{1|2|3|4}:CONDition?
:INSTrument:XSFP:CONDition?
:INSTrument:WAV:CONDition?
```

The device dependent event register can be initialized using the following queries:

```
:INSTrument:PE{1|2|3|4}:RESet
:INSTrument:XSFP:RESet
:INSTrument:WAV:RESet
```

# 2.7 Checking If Message Execution Is Completed

Some of the BERTWave program messages may take several seconds to several ten seconds to execute. Depending on the interface (Ethernet or GPIB) you are using, the procedure to check if time-consuming message execution is completed is different.



## 2.7.1 When using Ethernet

Even during execution of a program message sent to the BERTWave, the subsequent message(s) can be sent. However, until execution of the previously sent message(s) is completed, the subsequent message(s) is not processed, and is stored in the buffer of the BERTWave.

Therefore, note that execution of the previously sent program message(s) may not have always been completed, even if the subsequent message(s) can be sent. To check if execution of a program message is completed, send a query to receive a response.

Example of Use:

| | |
|---|---|
| `:CALibrate:AMPLitude` | Starts Level calibration for Scope |
| `:SYSTem:ERRor?` | Query for error code and error message |
| `> 0,"No Error"` | No error |

It may take around fifty seconds to complete execution of the `AMPLitude` command in this example. `AMPLitude` is sent, and subsequently `:SYSTem:ERRor?`, and then execution completion of `AMPLitude` is checked by receiving a response message.

***Note:***

If it may take long time to process the command that precedes the query, set the sufficient response timeout, with respect to the command processing time. (Timeout needs to be longer by at least 10 seconds than the command processing time.)

### 2.7.2 When using GPIB

Sending the subsequent message is forced to wait, on the control PC, until execution of the sent message is completed. Therefore, prevent a communication timeout from occurring during message execution by the BERTWave, when sending a time-consuming command. Set the sufficient timeout for GPIB interface of the control PC, with respect to the command processing time. (Timeout needs to be longer by at least 10 seconds than the command processing time.)

Example: Sending a command that takes about 20 seconds to execute

1. Set the timeout for GPIB interface to 30 seconds.
2. Send a command which takes a time to complete.
3. Reset the timeout to the previous setting.

# Chapter 3 Message List

This chapter describes the message details of remote control commands for BERTWave

# 3.1 Rules for Describing Messages

The following table shows the rules for describing messages.

**Table 3.1-1 Rules for Describing Messages**

<table>
<tr><th>Symbols</th><th>Usage</th></tr>
<tr><td><></td><td>Parameters in angled bracket are input by the programmer.</td></tr>
<tr><td>[]</td><td>Messages or parameters in square brackets can be omitted.</td></tr>
<tr><td>|</td><td>Select one of choices separated by vertical bars. For example, if A | B | C | D are choices, select one of them.</td></tr>
<tr><td>{}</td><td>Group the choices. For example, A | B({C | D}) means that A, B(C), or B(D) is available.</td></tr>
<tr><td><binary_data></td><td>This string is in binary data format.</td></tr>
<tr><td><character></td><td>Alphabet or numeric characters</td></tr>
<tr><td><file_name></td><td>The string indicates file name and path. The double quotation marks or single quotation marks are needed at the beginning and end of the data.<br>\,/,:,*,?,",<,>,| are not used in the file name.<br>Example: "PATTERN005"</td></tr>
<tr><td><integer></td><td>Decimal integer<br>Example: –100, 12500000</td></tr>
<tr><td><numeric></td><td>Decimal number<br>Example: 0, 1.2E–6, 2.35</td></tr>
<tr><td><string></td><td>String data<br>The double quotation marks or single quotation marks are needed at the beginning and end of the data.</td></tr>
<tr><td><enable></td><td>On/Off setting<br>To turn off, specify 0 or OFF. (Response: 0)<br>To turn on, specify 1 or ON. (Response: 1)<br>Example: 0, 1, OFF, ON</td></tr>
<tr><td>...</td><td>Indicates that multiple parameters or responses are omitted.</td></tr>
<tr><td>></td><td>Precedes a response, in Example of Use.</td></tr>
</table>

Some parts of the header strings can be omitted.
The lower-case characters can be omitted, but the upper-case characters cannot be omitted.

Example: `:STATus:OPERation:EVENt?`

The following are also acceptable:

```
:STAT:OPER:EVEN?
:STAT:OPERATION:EVEN?
:STATUS:OPERAT:EVENT?
:STATUS:OPERATION:EVEN?
:STATUS:OPERATION:EVENT?
```

BERTWave interprets them in the same way.

# 3.2 Specifying Module and Channel

There are two methods of specifying one of the function menu items of BERTWave, by using remote commands.

## 3.2.1 Specifying with command

This section describes the method of specifying a function menu item, by using the :MODule:ID command.
The commands sent after the :MODule:ID command will control items on the specified function menu.

### :MODule:ID

**Function**

This command sets and queries the module that is subject to remote control.

**Syntax**

```
:MODule:ID <module_id>
:MODule:ID?
```

**Parameter**

<module_id> = {1|2|3|4|5|6|7|8|9}

| | Function menu items |
|---|---|
| 1 | PPG/ED Ch1 |
| 2 | PPG/ED Ch2 |
| 3 | XFP/SFP+ |
| 4 | O/E |
| 5 | EYE/Pulse Scope |
| 6 | Jitter Analysis |
| 7 | Transmission Analysis |
| 8 | PPG/ED Ch3 |
| 9 | PPG/ED Ch4 |

**Response Data**

<module_id>

**Example of Use**

To start/stop ED Ch1 measurement:

```
:MOD:ID 1
:SENSe:MEASure:STARt
:SENSe:MEASure:STOP
```

To start ED Ch3 measurement:

```
:MOD:ID 8
```

```
:SENSe:MEASure:STARt
```

To start Scope measurement:

```
:MOD:ID 5
:SAMPling:STATus RUN
```

To query the module ID:

```
:MOD:ID?
>5
```

***Notes:***

- The IDs for Ch3 and Ch4 of PPG/ED are respectively 8 and 9, not 3 and 4.
- If the ID is not specified correctly, an Undefined Header error occurs.



### 3.2.2 Specifying in header

This section describes the method of specifying a function menu item, by adding a keyword for module/channel at the beginning of the header.
This method is available if the version number is 3.02 or above.
It is not required to send the :MODule:ID command described in 3.2.1 “Specifying with command”.

| Keyword | Function menu items |
|---|---|
| :BERT[<ch>] | PPG/ED Ch1 to 4<br>If <ch> is omitted, it defaults to Ch1. |
| :PMODule | XFP/SFP+ (Pluggable Module) |
| :OE | O/E |
| :SCOPe | EYE/Pulse Scope |
| :JITTer | Jitter Analysis (MX210001A) |
| :VNA | Transmission Analysis (MX210002A) |

**Example of Use**

To start ED Ch1 measurement:

```
:BERT:SENSe:MEASure:STARt
```

To stop ED Ch1 measurement:

```
:BERT:SENSe:MEASure:STOP
```

To start ED Ch3 measurement:

```
:BERT3:SENSe:MEASure:STARt
```

To start Scope measurement:

```
:SCOPe:SAMPling:STATus RUN
```

***Notes:***

- The above keywords cannot be used in native commands.

- This method is not available if the compatibility with the version 3.02 or earlier is required.
- This method is also applicable to Jitter and VNA by removing :SENSe from the beginning of the command.

**Example of Use**

```
:JITTer:GRAPh:BATHtub:SAMPle
:VNA:WE:SIGNal:SOURce
```

# 3.3 Correspondence Between Panel and Messages

This section explains correspondence between panel and messages.

## 3.3.1 Messages corresponding to common operations

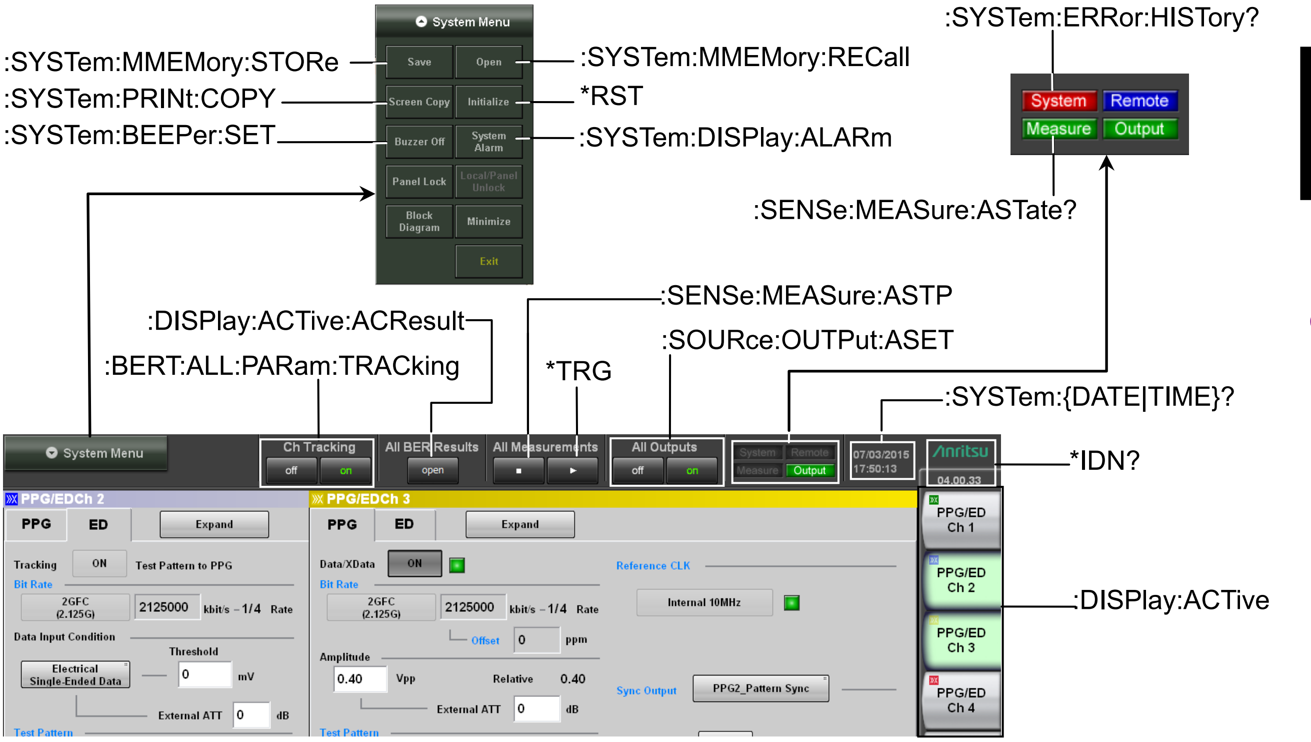



**Figure 3.3.1-1 Messages Corresponding to Common Operations**

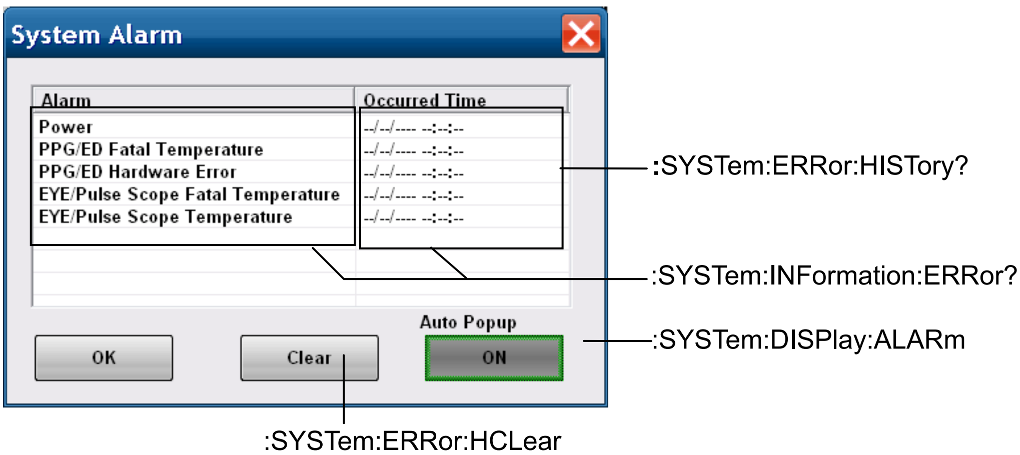



**Figure 3.3.1-2 Messages Corresponding to System Alarm Dialog Box**

## 3.3.2 Messages corresponding to PPG/ED

To control PPG/ED, specify the channel with the :MODule:ID command, in advance.

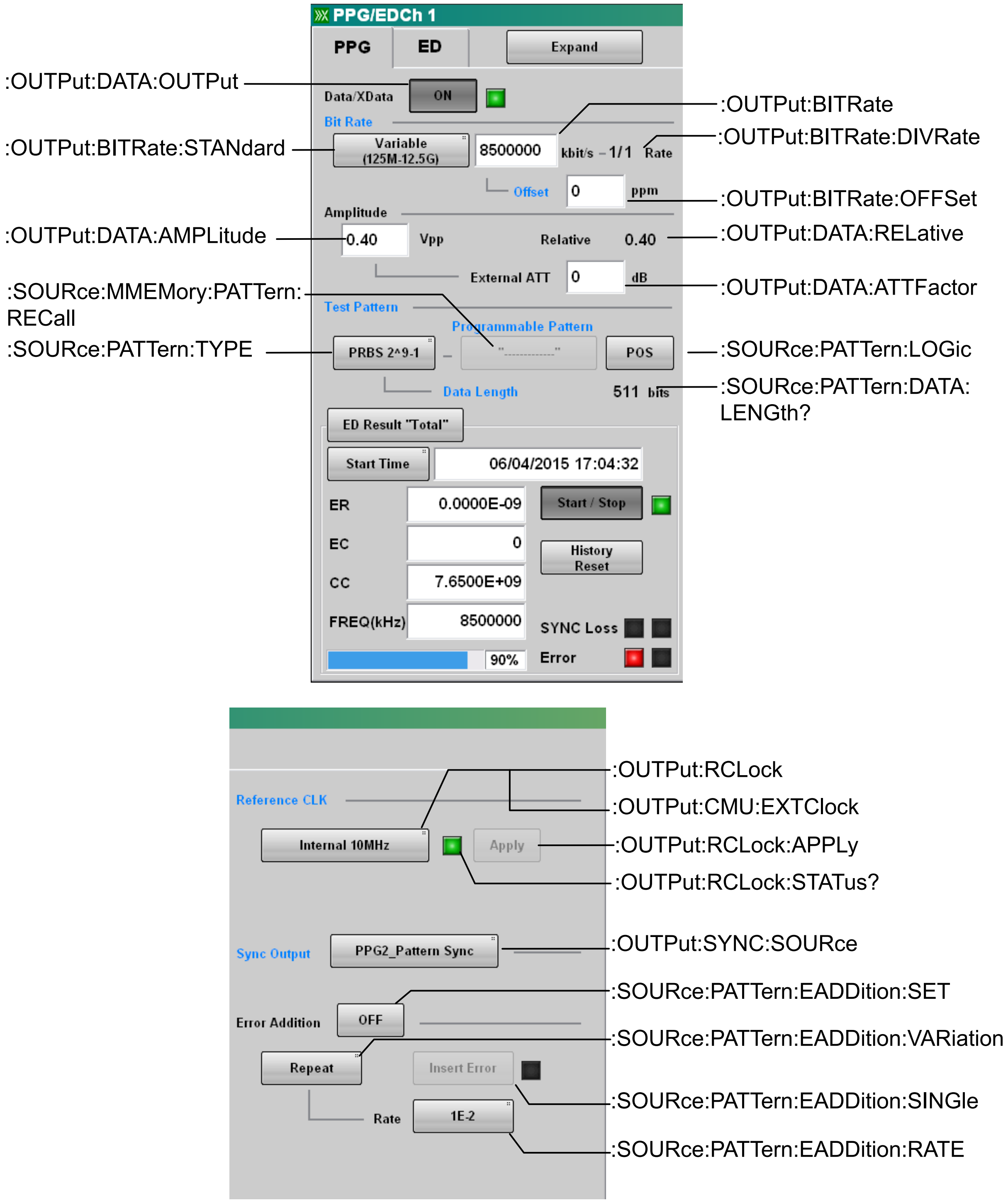
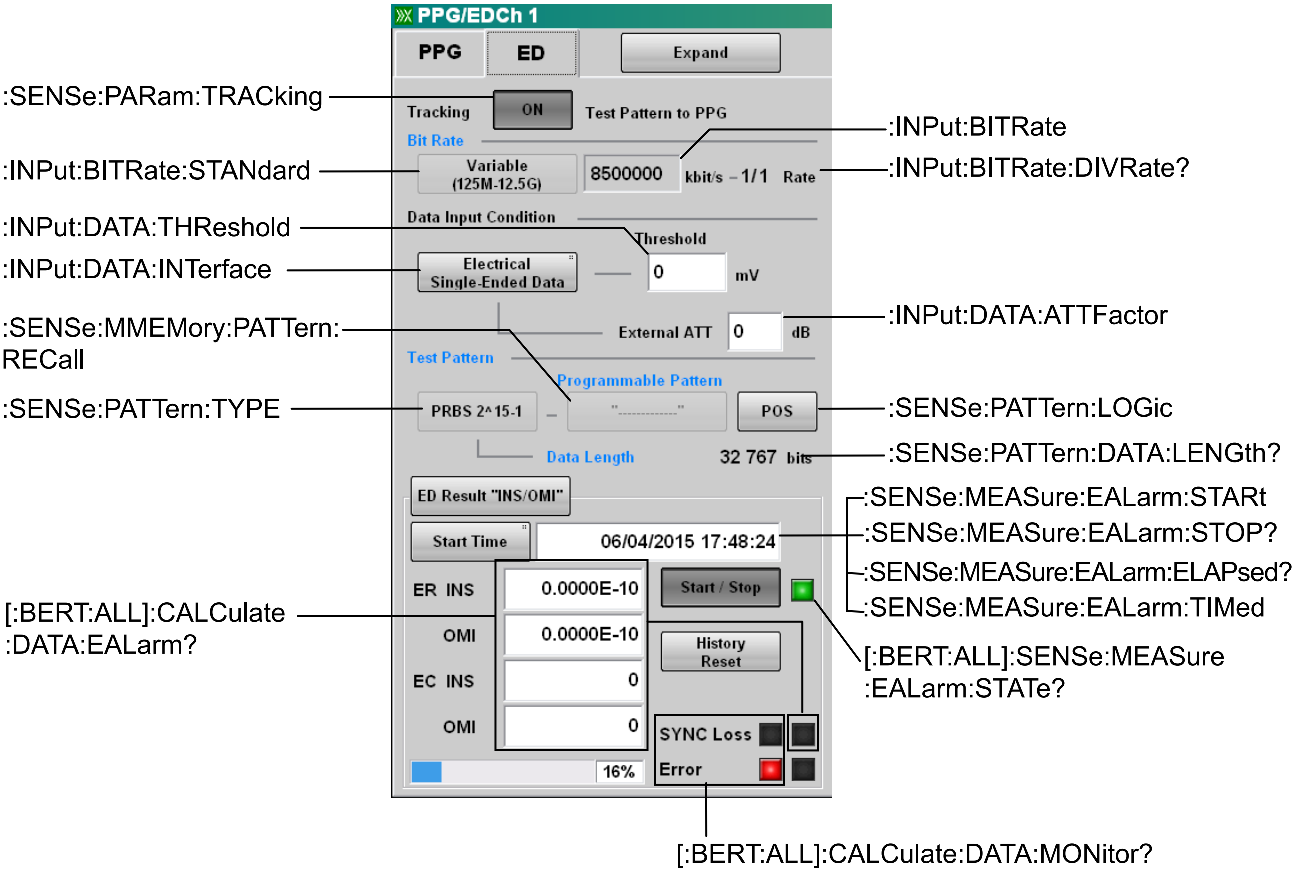



**Figure 3.3.2-1 Messages Corresponding to PPG Panel**



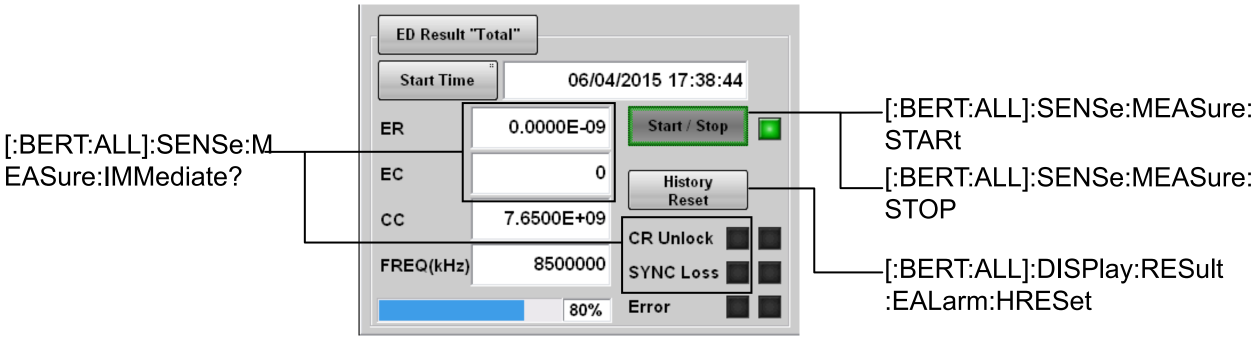




**Figure 3.3.2-2 Messages Corresponding to ED Panel–1**

3

Message List


**Figure 3.3.2-3 Messages Corresponding to ED Panel–2**

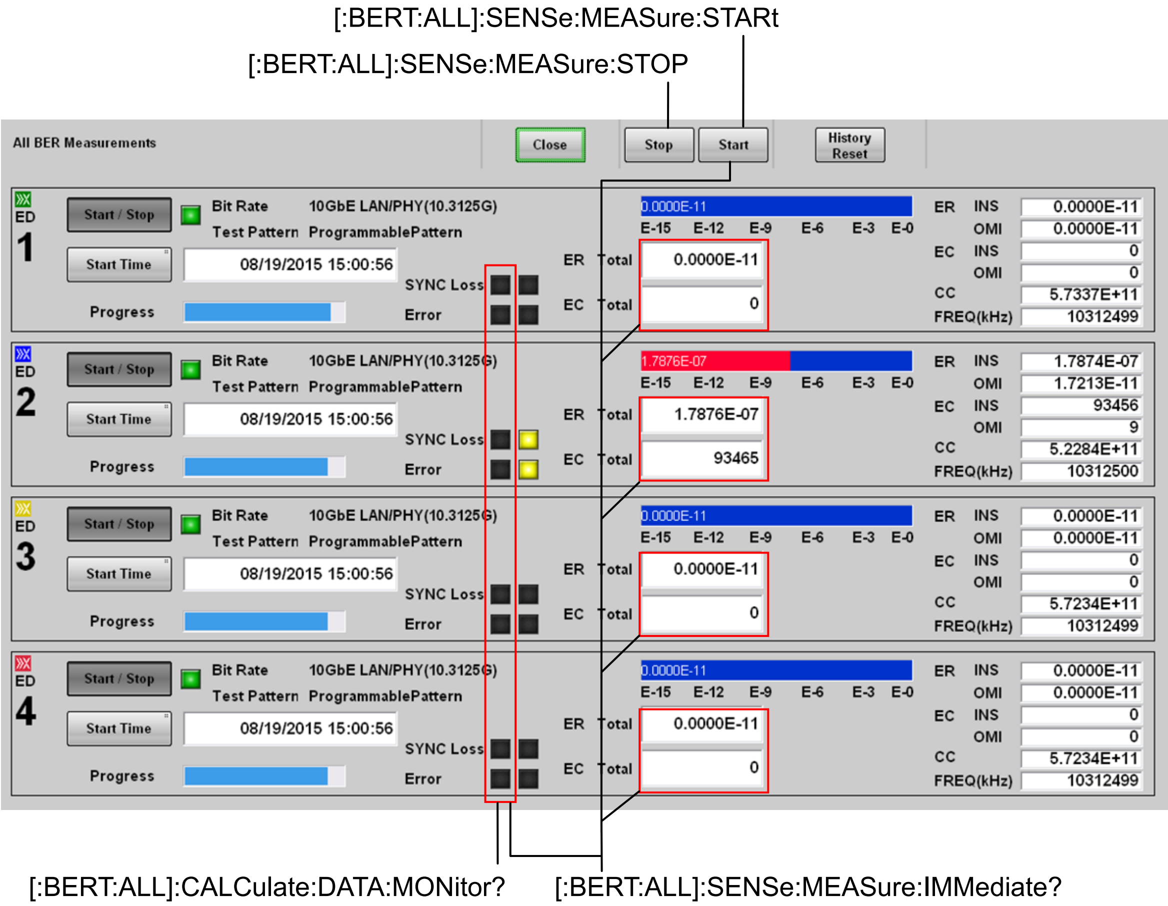



Figure 3.3.2-4 Messages Corresponding to All BER Measurement-1

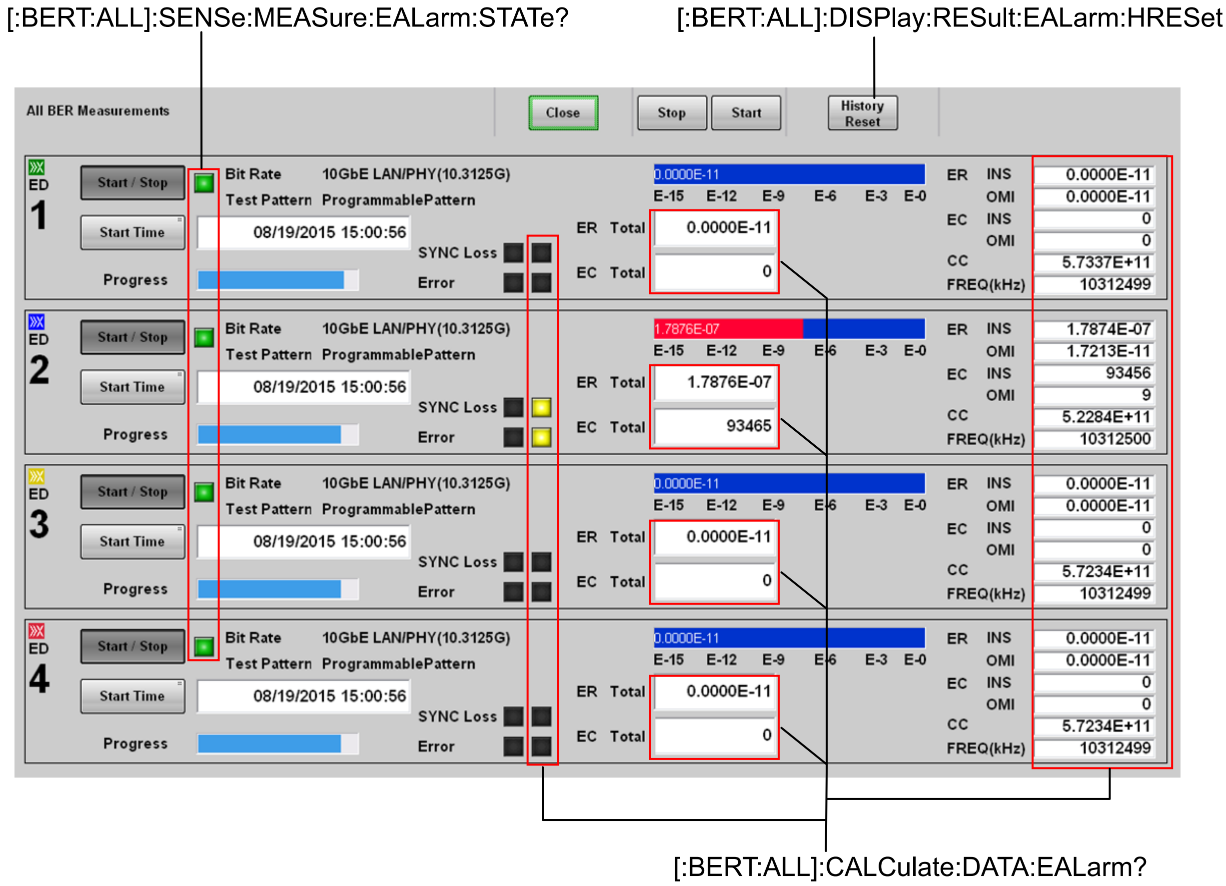



Figure 3.3.2-5 Messages Corresponding to All BER Measurement-2

### 3.3.3 Messages corresponding to XFP/SFP+

When controlling XFP/SFP+, send :MODule:ID 3 at first.

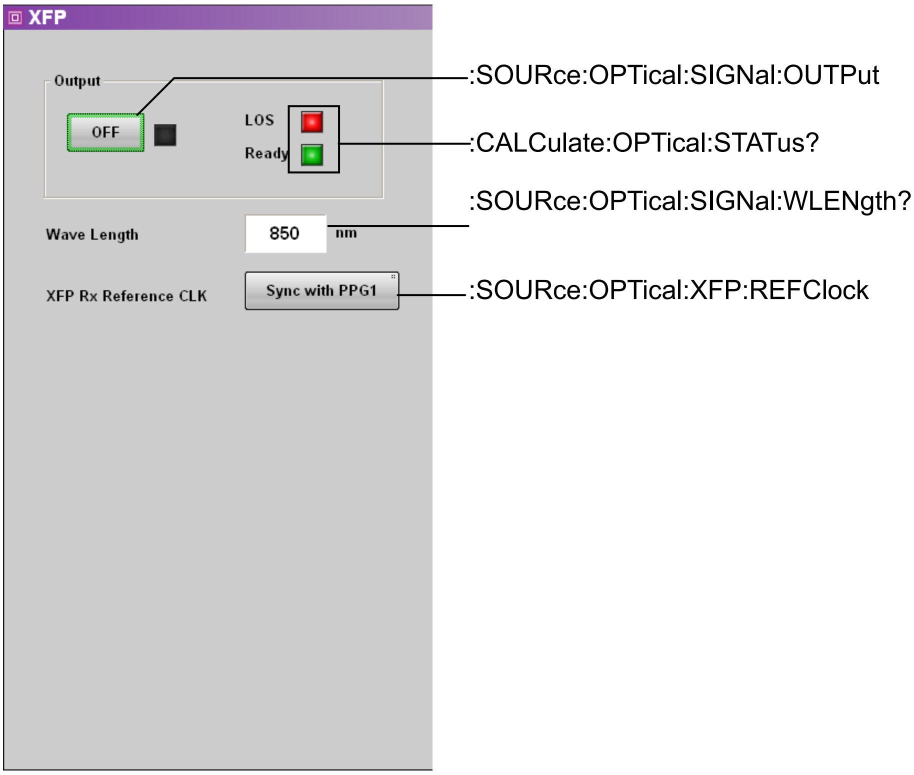



**Figure 3.3.3-1 Messages Corresponding to XFP/SFP+ Panel**

## 3.3.4 Messages corresponding to O/E

When controlling O/E, send :MODule:ID 4 at first.

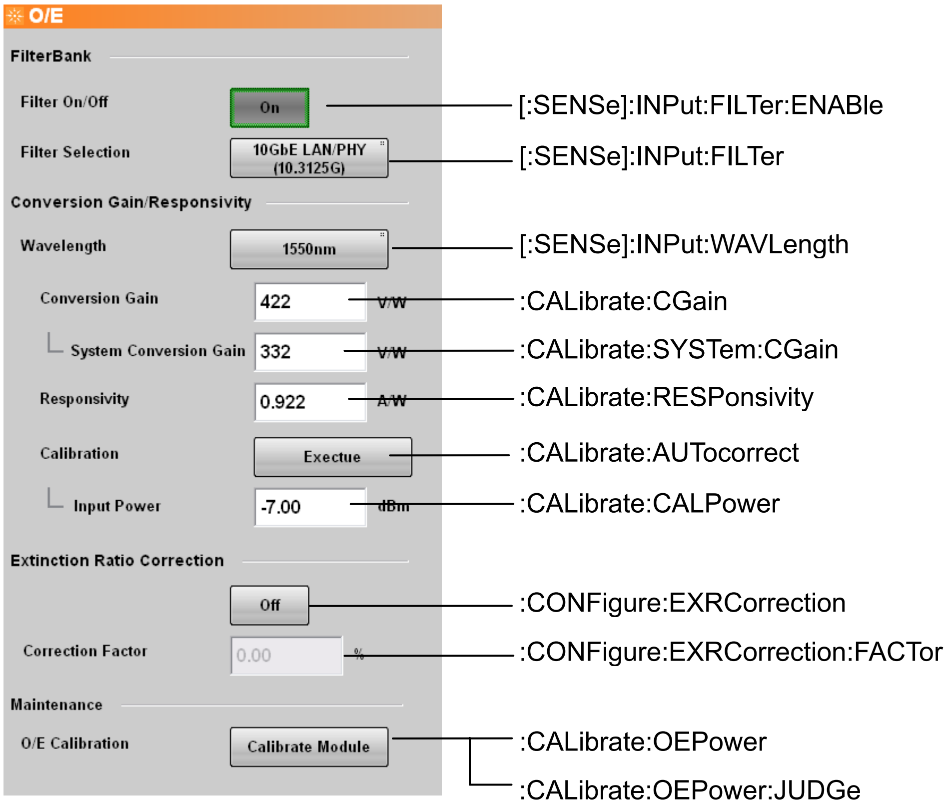



**Figure 3.3.4-1 Messages Corresponding to O/E Panel**

### 3.3.5 Messages corresponding to Scope

When controlling Scope, send :MODule:ID 5 at first

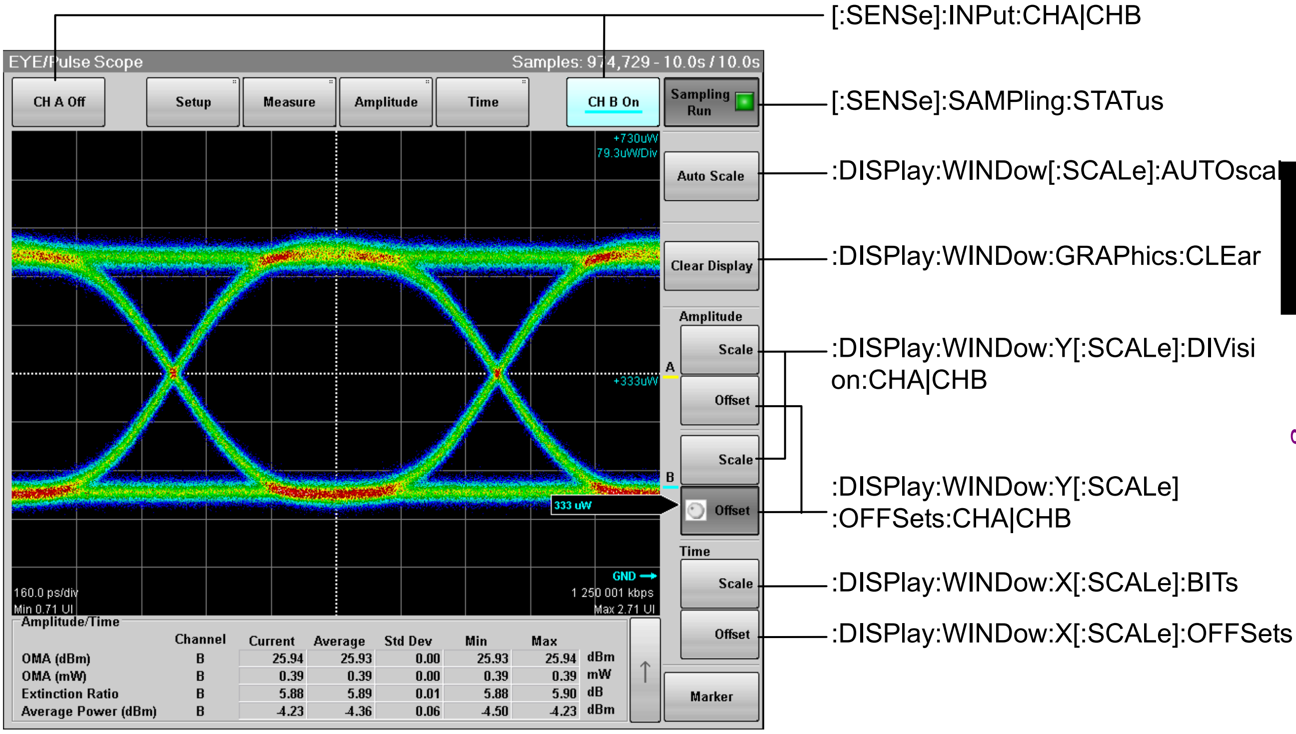



**Figure 3.3.5-1 Messages Corresponding to Scope Panel**

3

Message List

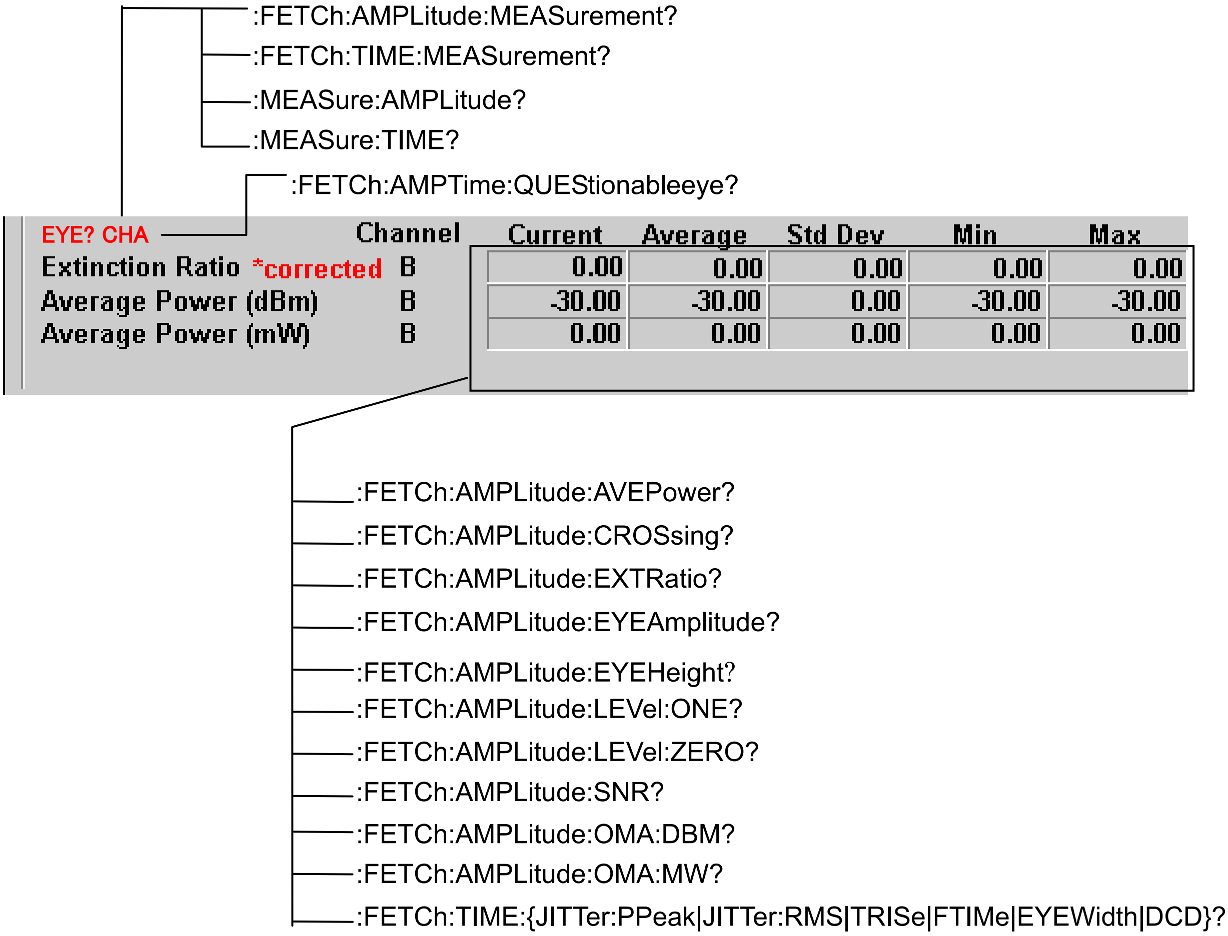



**Figure 3.3.5-2 Messages Corresponding to Amplitude/Time Measurement Result**

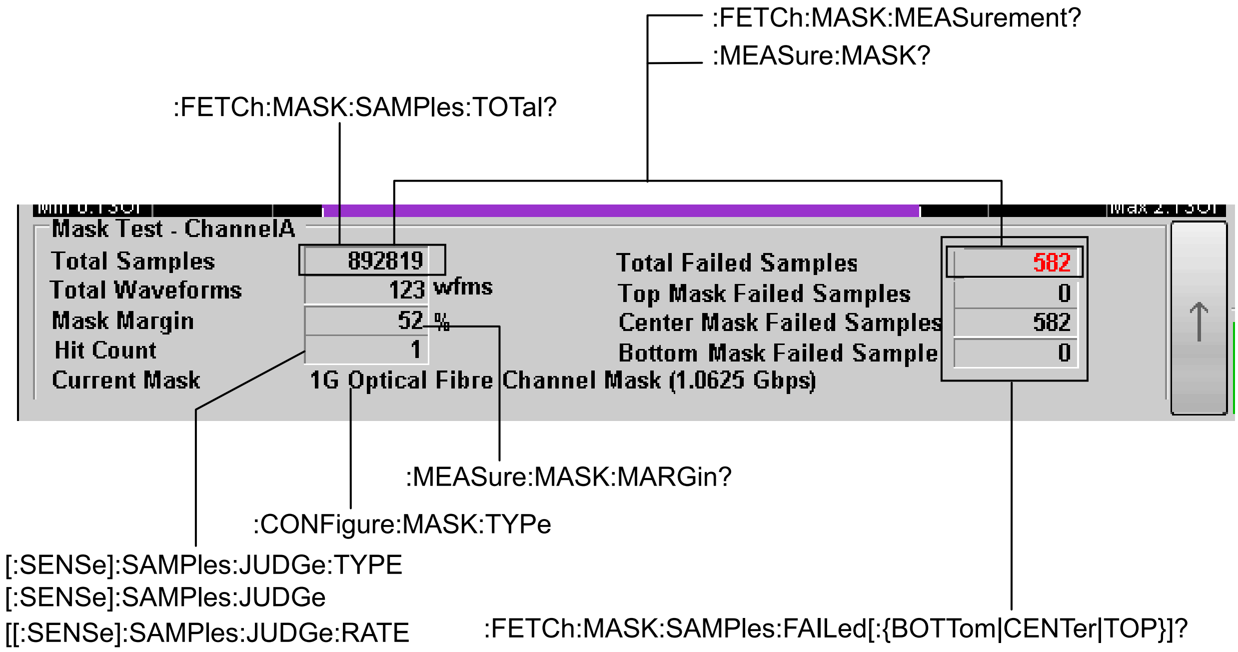



**Figure 3.3.5-3 Messages Corresponding to Mask Test Measurement Result**

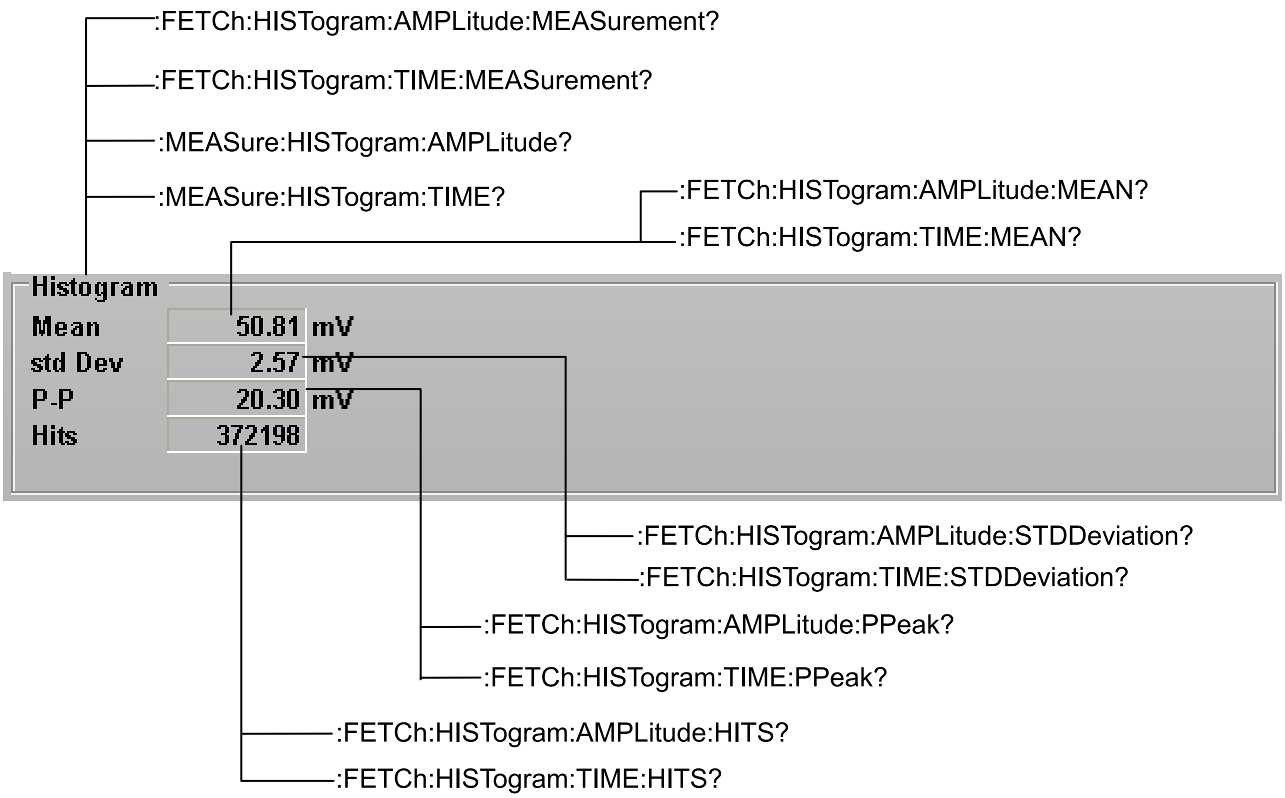



**Figure 3.3.5-4 Messages Corresponding to Histogram Measurement Result**

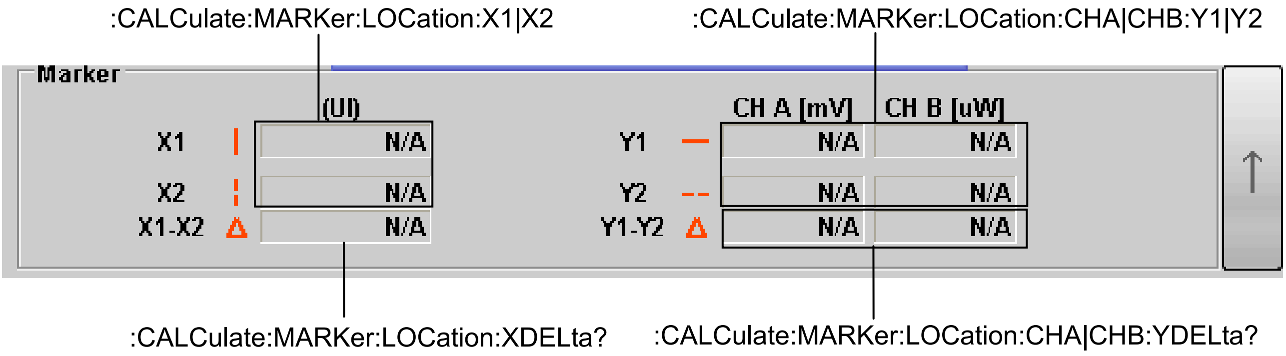



**Figure 3.3.5-5 Messages Corresponding to Marker Display**

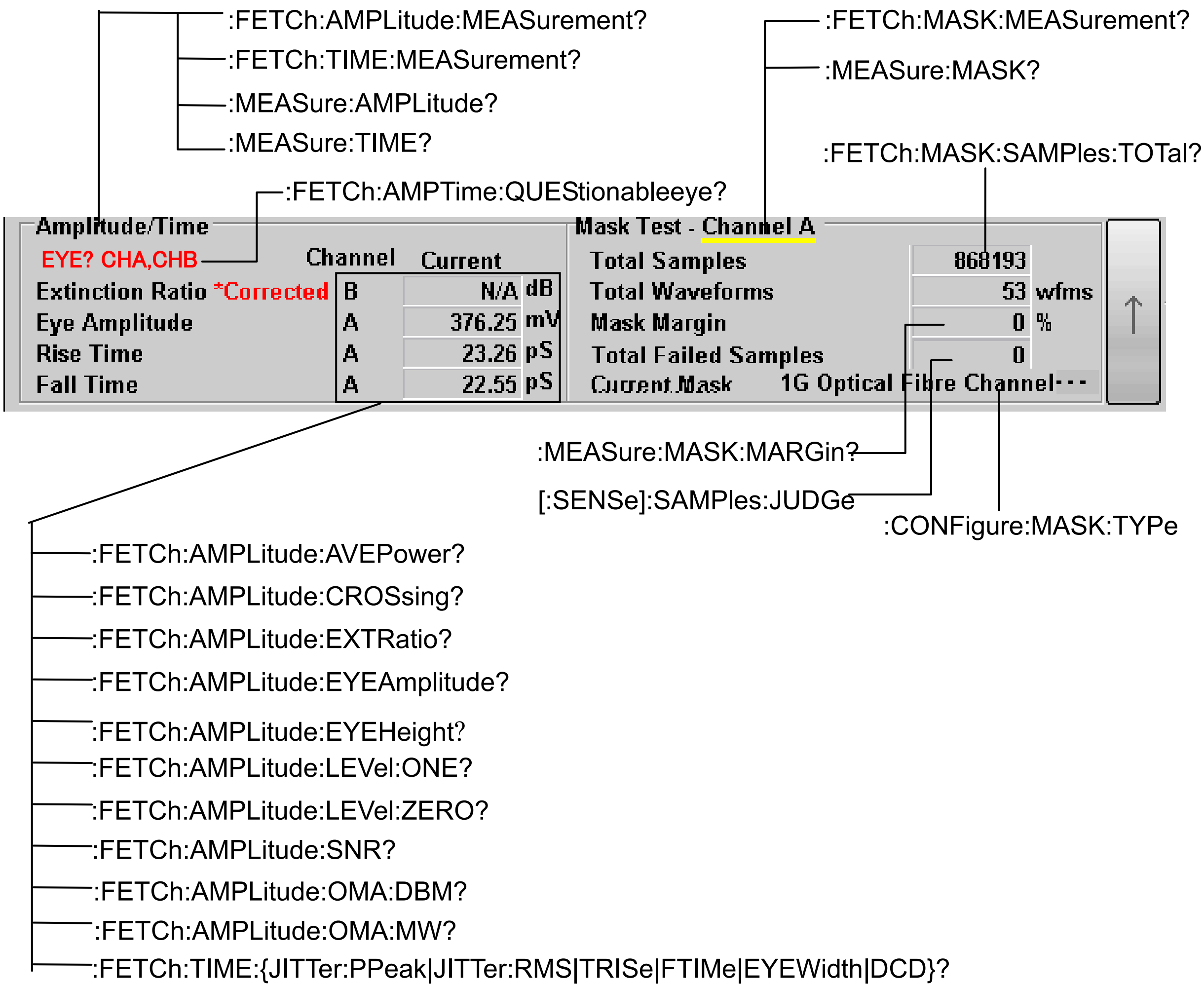



**Figure 3.3.5-6 Messages Corresponding to Amplitude/Time&Mask Measurement Result**

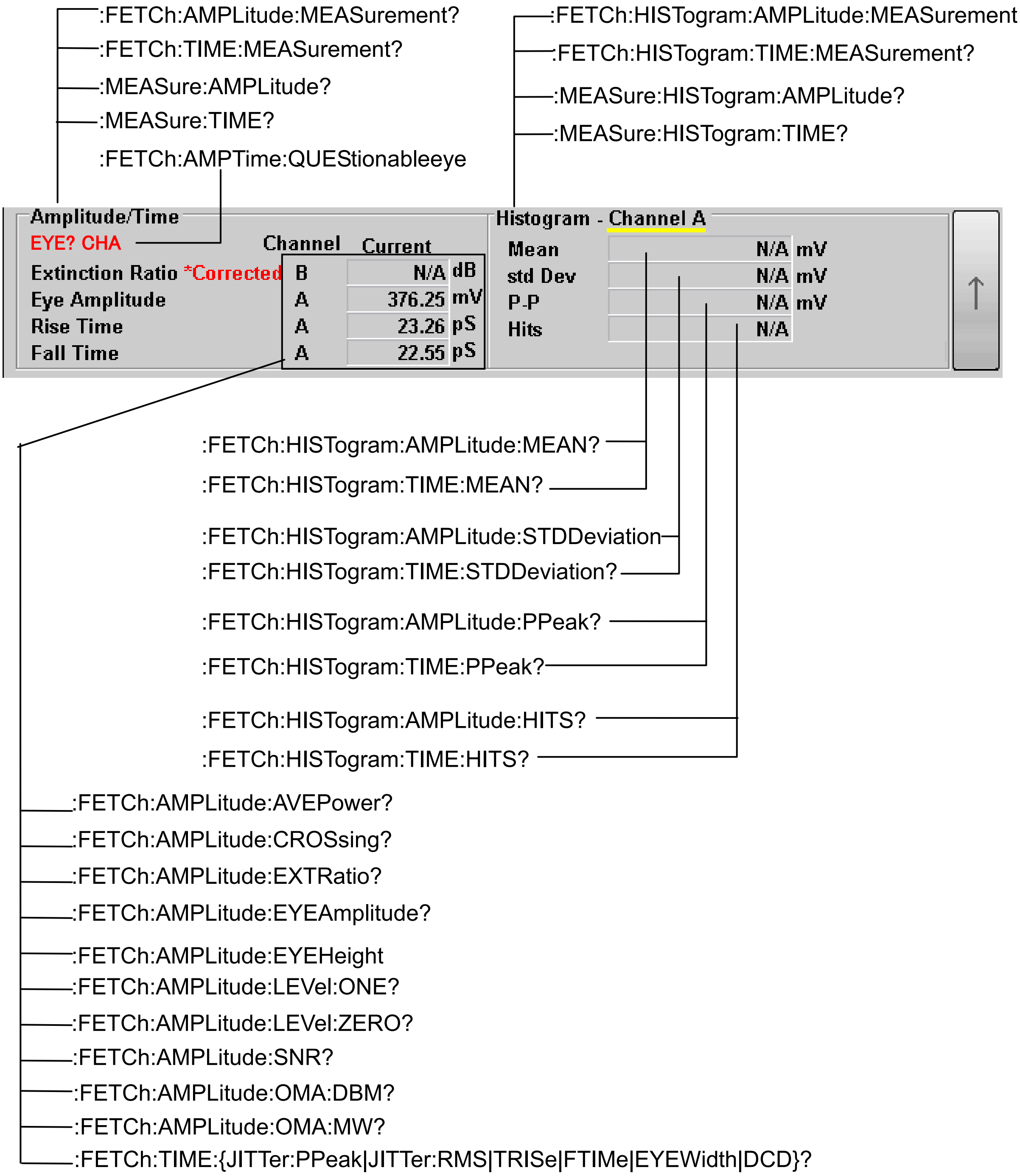



**Figure 3.3.5-7 Messages Corresponding to Amplitude/Time&Histogram Measurement Result**

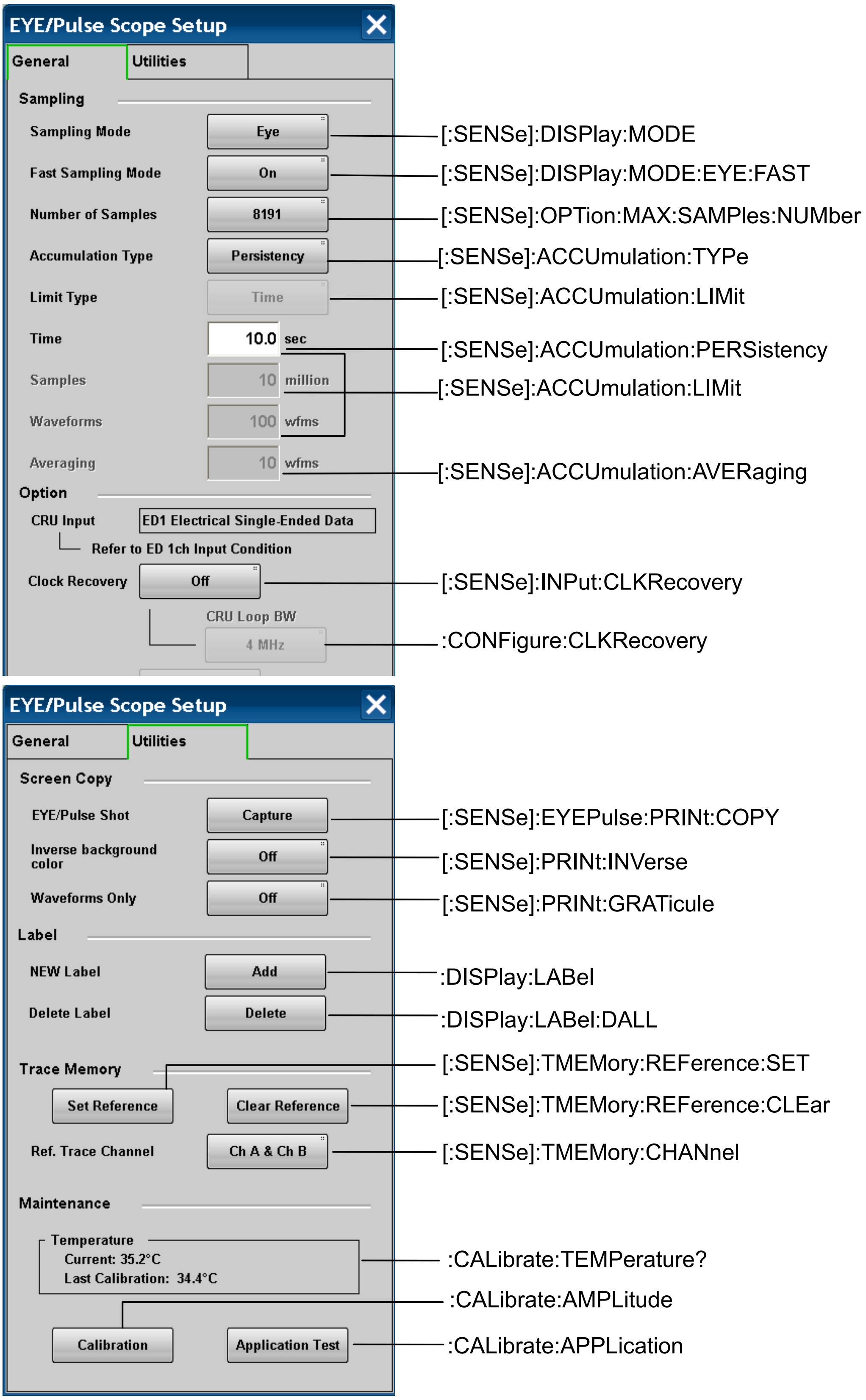



**Figure 3.3.5-8 Messages Corresponding to Setup Dialog Box**

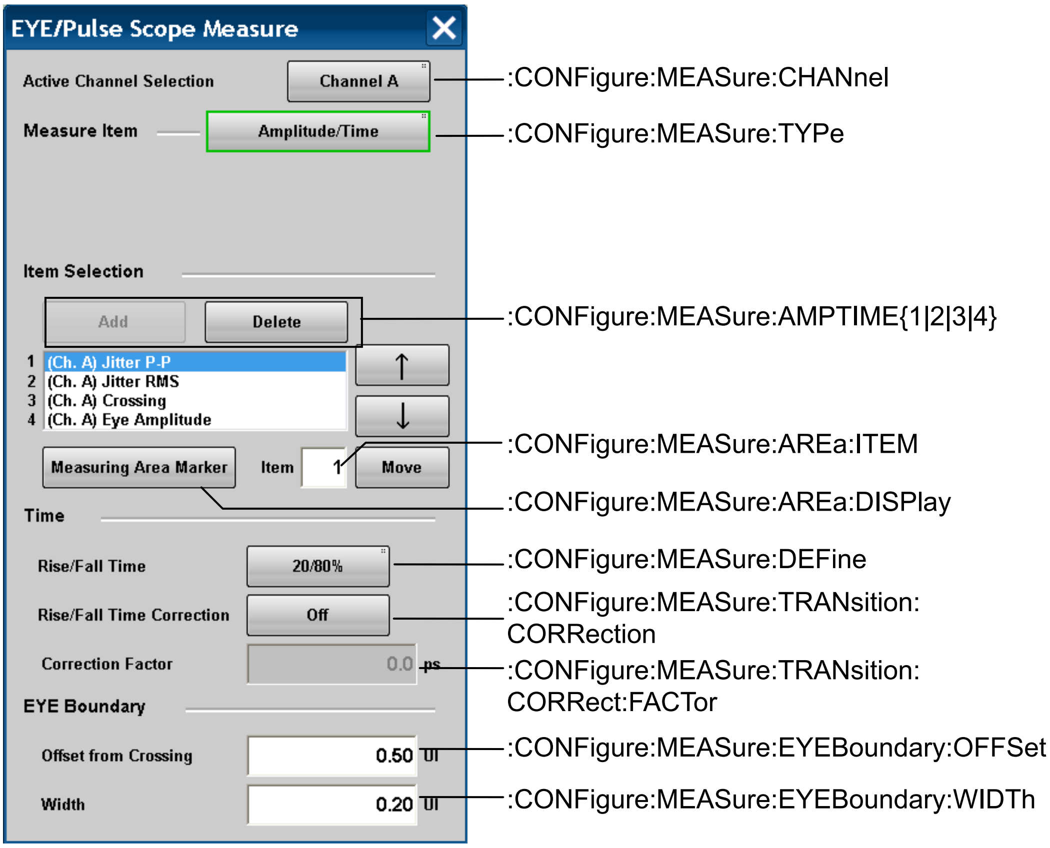



**Figure 3.3.5-9 Messages Corresponding to Measure Dialog Box (Amplitude/Time,Amplitude/Time&Mask,Amplitude/Time&Histogram)**

3

Message List

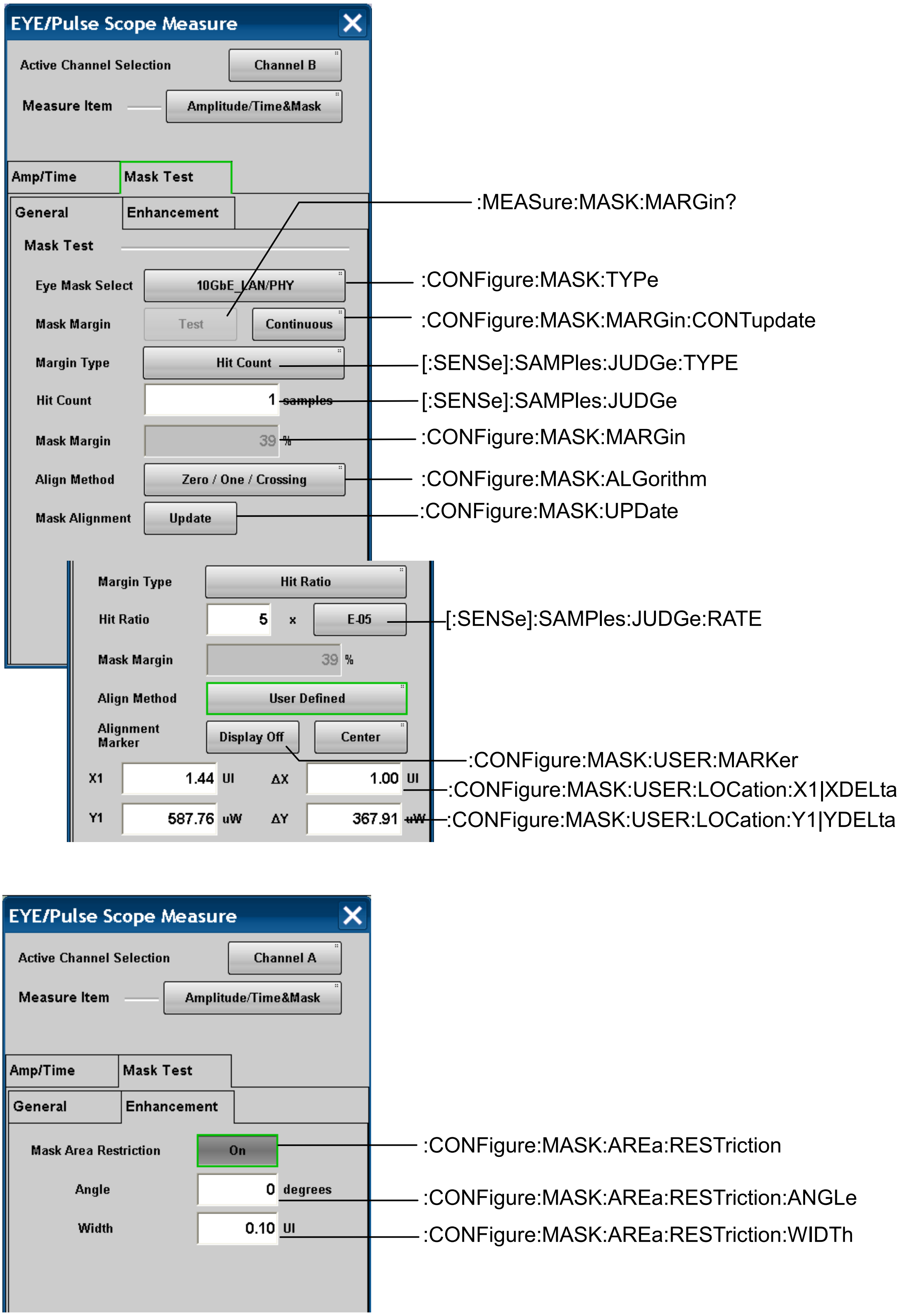



**Figure 3.3.5-10 Messages Corresponding to Measure Dialog Box (Mask Test, Amplitude/Time&Mask)**

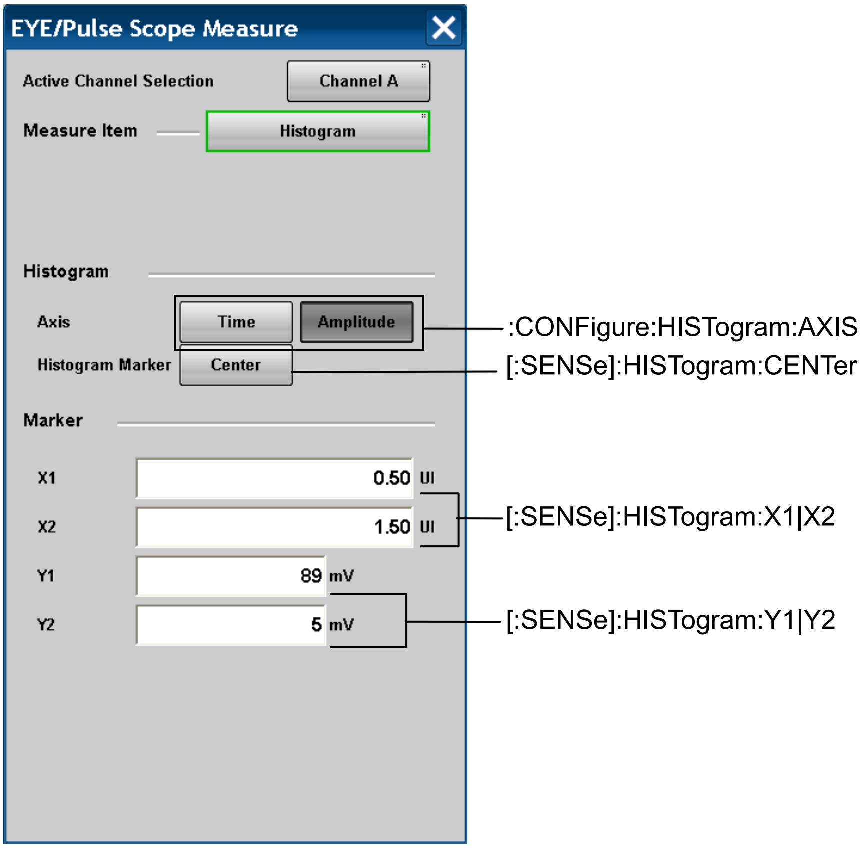
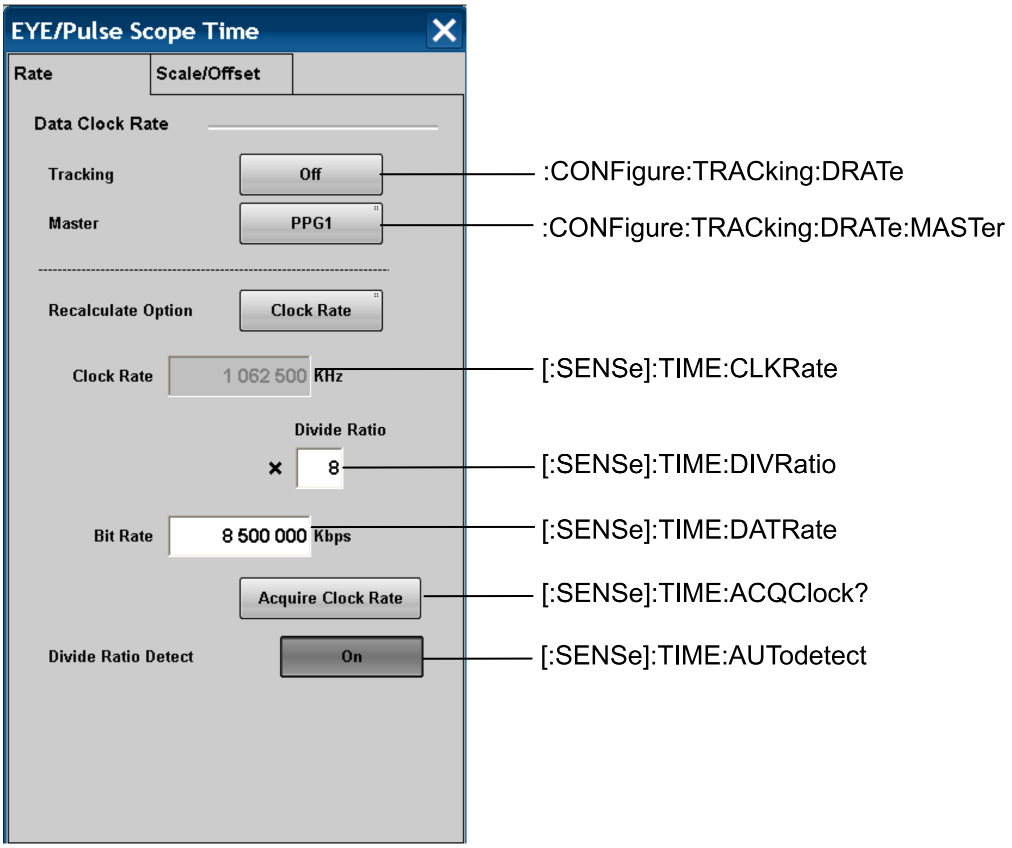



**Figure 3.3.5-11 Messages Corresponding to Measure Dialog Box (Histogram, Amplitude/Time&Histogram)**



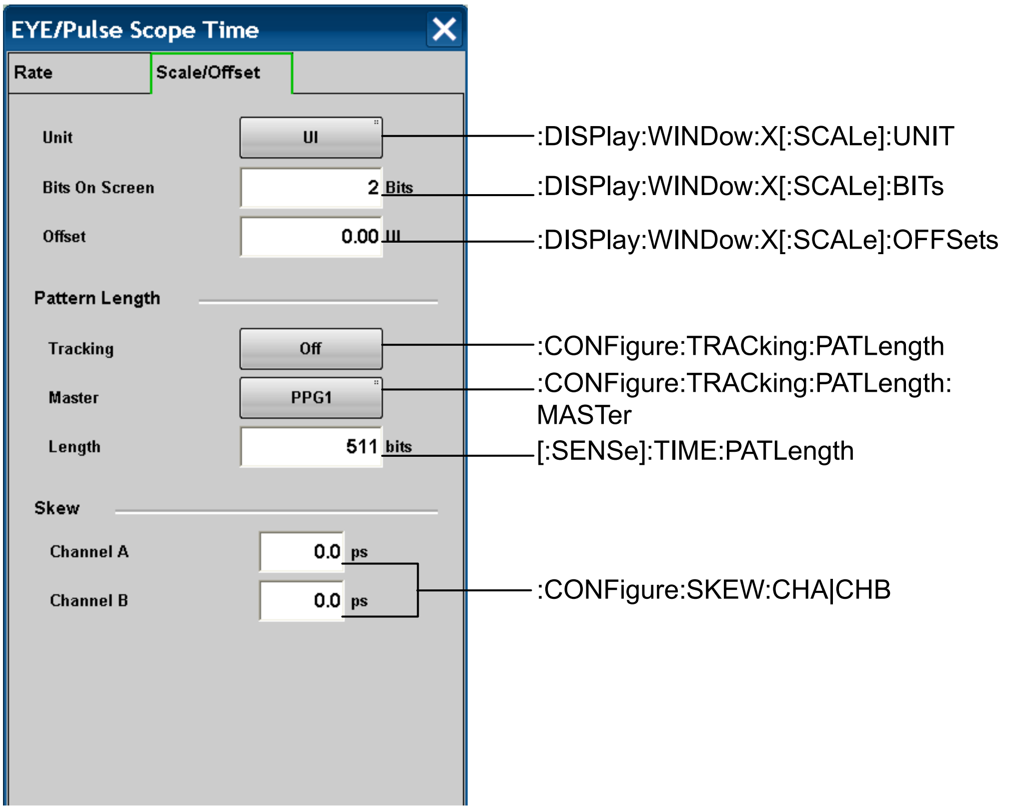




**Figure 3.3.5-12 Messages Corresponding to Time Dialog Box**

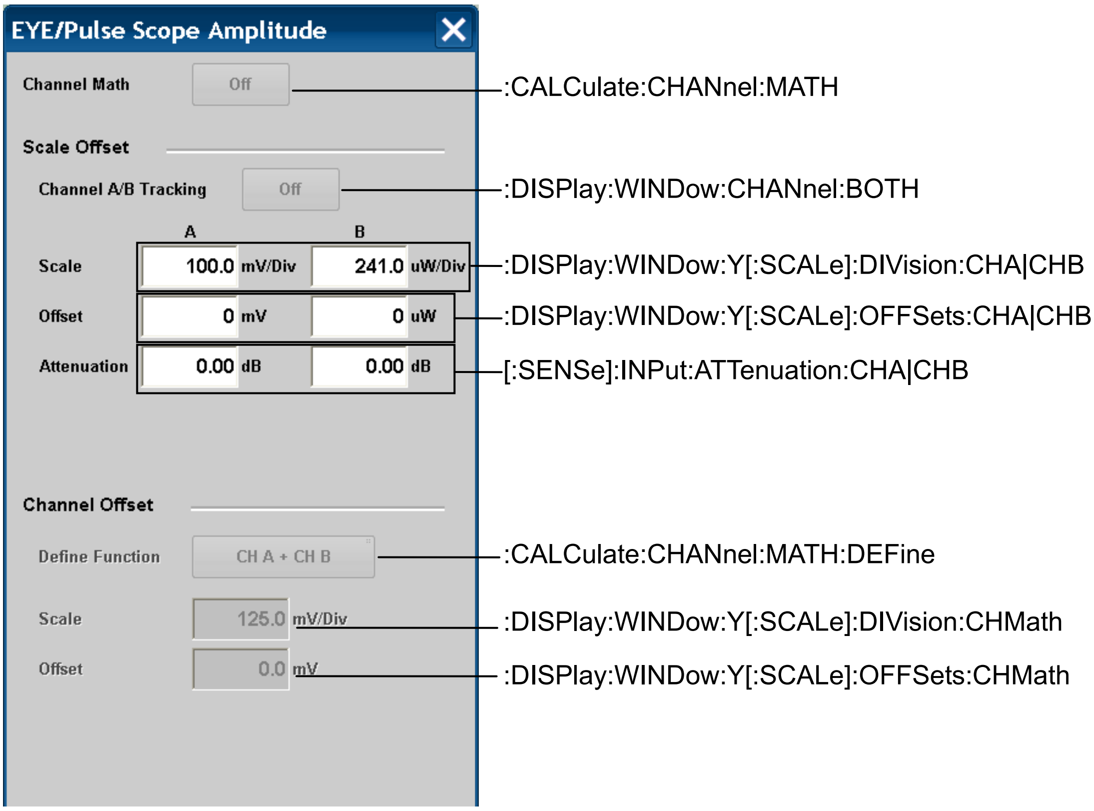



**Figure 3.3.5-13 Messages Corresponding to Amplitude Dialog Box**

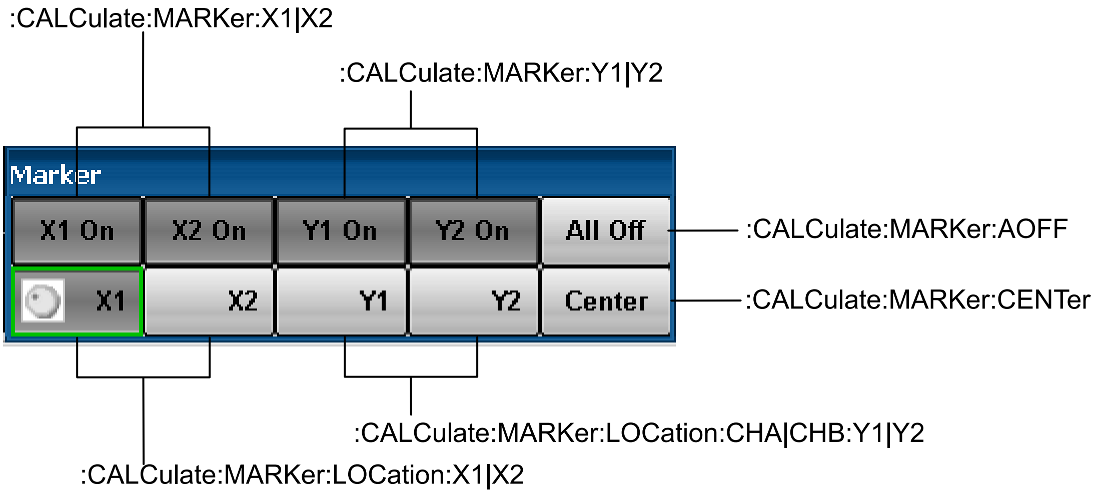



**Figure 3.3.5-14 Messages Corresponding to Marker Dialog Box**

## 3.3.6 Information dialog box for set-up utility

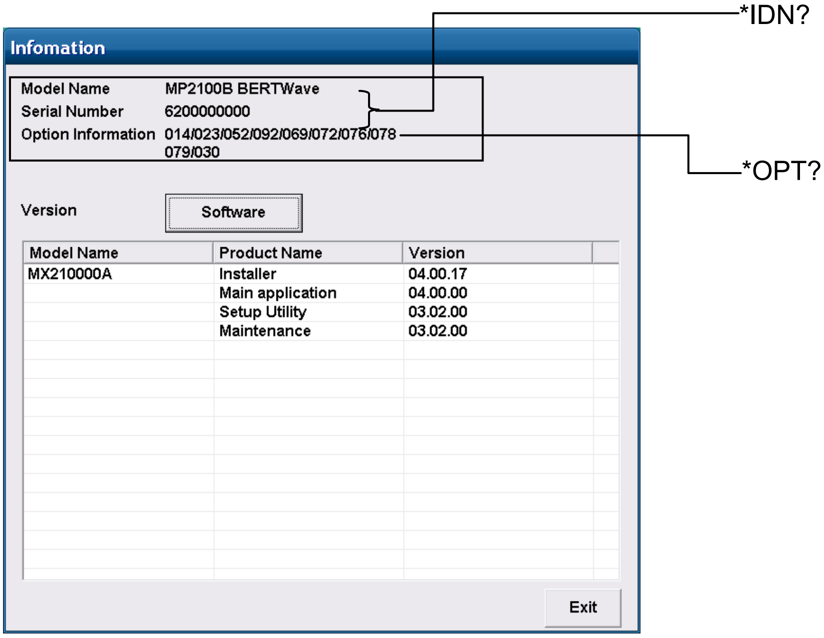



**Figure 3.3.6-1 Messages Corresponding to Information Dialog Box**

### 3.3.7 Messages with no corresponding panel operation

Command messages with no corresponding panel operation are listed below.

For messages corresponding to the status registers, refer to 2.6.4 “Operation Status Register” and 2.6.5 “Device Dependent Register”.

**Table 3.3.7-1 Messages with No Corresponding Panel Operation (SCPI)**

| Command | Details |
|---|---|
| :SYSTem:DISPlay:DATA? | Queries the data of the screenshot image file. |
| :SYSTem:DISPlay:RESult | Sets and queries the On/Off state of the plotting processing of measurement results. |
| :SYSTem:TERMination | Sets and queries the terminator. |
| :SYSTem:VERSion? | Queries the version of SCPI. |
| :TRACe[:DATA]:CHANnelA \| CHANnelB \| CHANnels? | Queries the trace data on the Scope.<br>This message is available only when in trace data readout mode. |
| :TRACe[:DATA]:END | Ends trace data readout mode of the Scope. |
| :TRACe[:DATA]:PREPare | Places the Scope in trace data readout mode. |

# 3.4 Messages Corresponding to Common Operations

## 3.4.1 Setting system configuration

### :SYSTem:TERMination

**Function**

This command sets and queries the type of terminator of the response data sent (or to be sent) from the BERTWave to the Control PC.

**Syntax**

```
:SYSTem:TERMination 0|1
:SYSTem:TERMination?
```

**Parameter**

0 LF+EOI (default)
1 CR+LF+EOI

**Response Data**

0|1

**Example of Use**

To set the terminator type to LF+EOI:

```
:SYST:TERM 0
:SYST:TERM?
>0
```

***Notes:***

- LF (Line Feed) is 0x0A in ASCII.
- CR (Carriage Return) is 0x0D in ASCII.
- EOI (End or Identify) is a GPIB interface signal indicating the end of data.

### :SYSTem:BEEPer:SET

**Function**

This command sets and queries the buzzer ON/OFF.

**Syntax**

```
:SYSTem:BEEPer:SET <enable>
:SYSTem:BEEPer:SET?
```

**Parameter**

0|OFF
1|ON

**Response Data**

0|1

**Example of Use**

To set buzzer ON:

```
:SYST:BEEP:SET ON
:SYST:BEEP:SET?
>1
```

## 3.4.2 Obtaining system information

### *IDN? (Identification)

**Function**

This command queries product supplier name, model name, serial number, and installer version.

**Syntax**

```
*IDN?
```

**Response Data**

Anritsu,MP2100{A|B},<serial_number>,<version>

**Example of Use**

```
*IDN?
>Anritsu,MP2100A,6200123456,03.01.00
```

### *OPT? (Option Identification Query)

**Function**

This command queries what options are installed.

**Syntax**

```
*OPT?
```

**Response Data**

<option_id>[,<option_id>]…

The installed options are returned in the OPT<number> format.

**Table 3.4.2-1 option_id List**

| <option_id> | Option Name |
|---|---|
| OPT001 | Dual Electrical Receiver |
| OPT003 | Optical/Single-ended Electrical Receiver |
| OPT005 | Extended PPG/ED Channel |
| OPT007 | 1ch Electrical BERT and Optical/Single-ended Electrical Scope |
| OPT011 | 1CH BERT |
| OPT012 | 2CH BERT |
| OPT014 | 4CH BERT |
| OPT021 | Dual Electrical Scope |
| OPT023 | Optical and Single-ended Electrical Scope |
| OPT030 | GPIB |
| OPT050 | XFP Slot |
| OPT051 | SFP+ Slot |
| OPT052 | Full Rate Clock Output |
| OPT053 | Clock Recovery (External Data) |
| OPT054 | Clock Recovery (Optical Data) |
| OPT055 | Clock Recovery |
| OPT056 | Low Pass Filter Bank (8.5G/10G/10.7G) |
| OPT057 | Low Pass Filter Bank (2G/4G/8.5G/10G) |
| OPT058 | Low Pass Filter Bank (1.2G/10G/10.7G) |
| OPT059 | Low Pass Filter Bank (1.2G/2.5G/3.1G/6.2G) |
| OPT060 | Low Pass Filter Bank (2G/3.1G/6.2G/10G) |
| OPT061 | 1 High Bit Rate Filter |
| OPT062 | 2 High Bit Rate Filter Bank |
| OPT063 | 4 High Bit Rate Filter Bank |
| OPT064 | 1 to 2 Low Bit Rate Filter Bank |
| OPT065 | 4 Low Bit Rate Filter Bank |
| OPT066 | 1 High Bit Rate and 1 to 2 Low Bit Rate Filter Bank |
| OPT067 | 1 to 2 High Bit Rate and 3 to 4 Low Bit Rate Filter Bank |
| OPT068 | 2 to 3 High Bit Rate and 1 to 2 Low Bit Rate Filter Bank |
| OPT069 | 3 High Bit Rate and 3 Low Bit Rate Filter Bank |

Table 3.4.2-1 option_id List (Cont'd)

| <option_id> | Option Name |
|---|---|
| OPT070 | LPF for 156M (L) |
| OPT071 | LPF for 622M (L) |
| OPT072 | LPF for 1.0G (L) |
| OPT073 | LPF for 1.2G (L) |
| OPT075 | LPF for 2.5G (L) |
| OPT076 | LPF for 2.1G (H) |
| OPT077 | LPF for 2.5G (H) |
| OPT078 | LPF for 2.6G (H) |
| OPT079 | LPF for 3.1G (H) |
| OPT080 | LPF for 4.2G (H) |
| OPT081 | LPF for 5.0G (H) |
| OPT082 | LPF for 6.2G (H) |
| OPT083 | LPF for 8.5G (H) |
| OPT084 | LPF for 9.9G to 10.3G (H) |
| OPT085 | LPF for 10.5G to 11.3G (H) |
| OPT086 | LPF for Multi 10G (8.5G to 11.3G) (H) |
| OPT087 | Filter Bank Set (622M/1.2G/2.5G/4.2G/6.2G/Multi 10G) |
| OPT088 | Filter Bank Set (4.2G/5.0G/6.2G/Multi 10G) |
| OPT089 | Filter Bank Set (156M/622M/1.2G/2.5G) |
| OPT090 | Bit rate Extension for PPG/ED |
| OPT091 | ED High Sensitivity |
| OPT092 | PPG/ED Bit Rate Extension for 125M to 12.5G |

**Example of Use**

```
*OPT?
>OPT001,OPT030,OPT050
```

## :SYSTem:VERSion?

**Function**

This command queries the SCPI version conforming to the software of the BERTWave.

**Response Data**

1999.0

**Example of Use**

```
:SYST:VERS?
>1999.0
```

## :SYSTem:INFormation?

**Function**

This command queries the manufacturer, model name, serial number, and installed option(s) of the BERTWave.

**Syntax**

`:SYSTem:INFormation?`

**Response Data**

Anritsu,MP2100{A | B},<serial_number>,<option_id>[,<option_id>]…

For <option_id>, refer to "*OPT?".

**Example of Use**

```
:SYST:INF?
>Anritsu,MP2100A,6200123456,OPT001,OPT050
```

## :SYSTem:{DATE|TIME}?

**Function**

This command queries the date or time of the BERTWave.

**Syntax**

```
:SYSTem:DATE?
:SYSTem:TIME?
```

**Response Data**

| | |
|---|---|
| When querying the date: | <year>,<month>,<day> |
| When querying the time: | <hour>,<minute>,<second> |

**Example of Use**

```
:SYST:DATE?
>2009,10,24
:SYST:TIME?
>9,50,39
```

## :SYSTem:ERRor?

**Function**

This command queries the error code and error message.

**Syntax**

`:SYSTem:ERRor?`

**Response Data**

<integer>,<string>

<integer>

Range –32768 to 32767

0 is returned if no errors have occurred. For error codes returned by the BERTWave, refer to Appendix B "Error Codes".

<string>

Error message corresponding to the value of <integer> (Up to 255 characters)

**Example of Use**

```
:SYST:ERR?
>0,"No error"
```



## 3.4.3 System alarm

### :SYSTem:ERRor:HCLear

**Function**

This command clears the system alarm history.

**Syntax**

```
:SYSTem:ERRor:HCLear
```

**Example of Use**

```
:SYST:ERR:HCL
```

### :SYSTem:ERRor:HISTory?

**Function**

This command queries if there exists a system alarm history.

**Syntax**

```
:SYSTem:ERRor:HISTory?
```

**Response Data**

| | |
|---|---|
| Not Occurred | No system alarms have occurred. |
| Occurred | A system alarm(s) has occurred. |

**Example of Use**

```
:SYST:ERR:HIST?
>Not occurred
```

## :SYSTem:DISPlay:ALARm

### Function

This command sets and queries the On/Off state of the Auto Popup for system alarm occurrence.

### Syntax

```
:SYSTem:DISPlay:ALARm <enable>
:SYSTem:DISPlay:ALARm?
```

### Parameter

0|OFF
1|ON

### Response Data

0|1

### Example of Use

```
:SYST:DISP:ALAR ON
:SYST:DISP:ALAR?
>1
```

## :SYSTem:INFormation:ERRor?

### Function

This command queries the system alarm information.

### Syntax

```
:SYSTem:INFormation:ERRor?
```

### Response Data

<error_code>[,<error_code>]…
0 is returned if no alarms have occurred.
If multiple alarms have occurred, all error codes are returned in ascending order.

| | |
|---|---|
| 1 | PPG/ED Fatal Temperature |
| 2 | EYE/Pulse Scope Temperature |
| 3 | PPG/ED PLL Unlock (MP210xA)<br>PPG/ED Hardware Error (MP2100B) |
| 4 | Power |
| 5 | EYE/Pulse Scope Fatal Temperature |

### Example of Use

```
:SYST:INF:ERR?
>1,2,3
```

## 3.4.4 Resetting/recalling settings

### *RST (Reset)

**Function**

This command resets the settings to the factory defaults.

**Syntax**

```
*RST
```

**Example of Use**

```
*RST
```

***Notes:***

- Output from PPG/XFP/SFP+ is turned off.
- ED/Scope measurement is stopped; the data obtained so far in the stopped measurement is cleared.
- The time at which the system alarm occurred is cleared.



### :SYSTem:MMEMory:RECall

**Function**

This command recalls the configuration file.

**Syntax**

```
:SYSTem:MMEMory:RECall <setup_file>
```

**Parameter**

<setup_file>

Specify the name and type of the configuration file you want to recall.

| Contents of File | Format |
|---|---|
| Settings for the entire BERTWave | "<file_name>[.CND]",0,ALL |
| Settings for PPG/ED Ch1 | "<file_name>.PE1",1,PE1 |
| Settings for PPG/ED Ch2 | "<file_name>.PE2",2,PE2 |
| Settings for PPG/ED Ch3 | "<file_name>.PE3",8,PE3 |
| Settings for PPG/ED Ch4 | "<file_name>.PE4",9,PE4 |
| Settings for XFP | "<file_name>.XFP",3,XFP |
| Settings for SFP+ | "<file_name>.SFP",3,SFP |
| Settings for O/E | "<file_name>.OES",4,OES |
| Settings for Scope | "<file_name>.WFS",5,WFS |
| Settings for Jitter | "<file_name>.JIT",6,JIT |
| Settings for Transmission | "<file_name>.TAS",7,TAS |

**Example of Use**

```
:SYST:MMEM:REC "settings_all",0,ALL
:SYST:MMEM:REC "settings_ppged1.PE1",1,PE1
```

## 3.4.5 Storing settings and results

### :SYSTem:MMEMory:STORe

**Function**

This command stores the settings or measurement results to a file.

**Syntax**

```
:SYSTem:MMEMory:STORe <setup_file>|<results_file>
```

**Parameter**

Specify the name and type of a file to which you want to store setting/results, as follows:

<setup_file>

Refer to “:SYSTem:MMEMory:RECall

<results_file>

<table>
<tr><th>Result</th><th>Format[*1,*2,*3]</th></tr>
<tr><td>EDCh1 measurement</td><td>"<file_name>.{CSV|TXT}",1,ER1,{CSV|TXT}</td></tr>
<tr><td>EDCh2 measurement</td><td>"<file_name>.{CSV|TXT}",2,ER2,{CSV|TXT}</td></tr>
<tr><td>EDCh3 measurement</td><td>"<file_name>.{CSV|TXT}",8,ER3,{CSV|TXT}</td></tr>
<tr><td>EDCh4 measurement</td><td>"<file_name>.{CSV|TXT}",9,ER4,{CSV|TXT}</td></tr>
<tr><td>Scope measurement</td><td>"<file_name>[.{CSV|TXT}]",5,WFR[,{CSV|TXT}]</td></tr>
<tr><td>Jitter measurement</td><td>"<file_name>.{CSV|TXT}",6,JIR,{CSV|TXT}</td></tr>
<tr><td>Transmission measurement (TransmissionAnalysis)</td><td>"<file_name>.{S2P|TXT}",7,TAR,{S2P|TXT}</td></tr>
<tr><td>Transmission measurement (WaveformEstimation)</td><td>"<file_name>.WFE",7,WER,WFE</td></tr>
</table>

*1: CSV, TXT, S2P and WFE indicate file formats.

*2: If CSV|TXT is omitted when storing Scope measurement results, the file is saved in binary format.

*3: File extensions, and third and fourth parameters are not case-sensitive.

**Example of Use**

To store settings:

```
:SYST:MMEM:STOR "settings_all",0,ALL
:SYST:MMEM:STOR "settings_ppged1.PE1",1,PE1
:SYST:MMEM:STOR "settings_jitter.JIT",6,JIT
```

To store measurement results:

```
:SYST:MMEM:STOR "results_ed1.TXT",1,ER1,TXT
```

```
:SYST:MMEM:STOR "results_scope.CSV",5,WFR,CSV
:SYST:MMEM:STOR
"results_transmission_analysis.S2P",7,TAR,S2P
:SYST:MMEM:STOR
"results_waveform_estimation.WFE",7,WER,WFE
```

***Note:***

If the file extension of the saved configuration file is changed, it cannot be recalled.



## 3.4.6 Turning on/off plotting processing

### :SYSTem:DISPlay:RESult

**Function**

This command sets and queries the On/Off state of the plotting processing of measurement results.

**Syntax**

```
:SYSTem:DISPlay:RESult {{0|OFF}[,ED]|{1|ON}}
:SYSTem:DISPlay:RESult?
```

**Parameter**

| | |
|---|---|
| 0\|OFF | Plotting processing Off |
| ED | *Added in Version 4:*<br>Plotting processing Off (Only ED measurement results) |
| 1\|ON | Plotting processing On |

**Response Data**

0|1

**Example of Use**

To stop the plotting processing of ED/Scope measurement results:

```
:SYST:DISP:RES OFF
:SYST:DISP:RES?
>0
```

To stop the plotting processing of only ED measurement results:

```
:SYST:DISP:RES OFF,ED
:SYST:DISP:RES?
>0
```

To resume the plotting processing:

```
:SYST:DISP:RES ON
:SYST:DISP:RES?
>1
```

***Note:***

The response time for remote control can be reduced by setting the plotting processing to Off. If set to Off, the screen displays the following message, instead of measurement results:

*The measurement results display cannot be updated during remote control.*
*Press the [Local/Panel Unlock] button to return to local control and re-open the updated measurement results display.*

## 3.4.7 Saving screenshot to file

### :SYSTem:PRINt:COPY

**Function**

This command saves the full screenshot to a file.

**Syntax**

```
:SYSTem:PRINt:COPY [<file_name>,<directory>][,PNG|JPEG]
```

**Parameter**

<file_name>,<directory>

Specify the file name and folder to save the screenshot. If the specified folder is not found, it is created automatically.

If the file name and folder are omitted, the screenshot is saved as mmddyyyy_hhmmssmmm.png|jpeg in the following directory:

C:\Program Files\Anritsu\MP2100A\MX210000A\UserData\Screen Copy

PNG|JPEG

Specify the image file format. If omitted, it defaults to PNG.

**Example of Use**

```
:SYST:PRIN:COPY "screen_copy_full","C:\screen_copy"
```

***Notes:***

- If the file name is not specified, an image file is created newly each time this command is sent. Make sure there is sufficient disk space.
- To save the screenshot of the Scope screen, execute :EYEP:PRIN:COPY.
- To obtain image file data, execute :SYST:DISP:DATA?.

### :SYSTem:DISPlay:DATA?

**Function**

This command queries the last screenshot image file saved by :{SYST|EYEP}:PRIN:COPY.

**Syntax**

```
:SYSTem:DISPlay:DATA?
```

**Response Data**

#<digit><data_size><binary_data><terminator>

<digit> is a one-digit number that indicates the number of digits of the value of <data_size>.
<data_size> indicates the data size of <binary_data>.
<binary_data> is image file data of the screenshot.
<terminator> indicates the terminator (LF or CR/LF) specified by the :SYST:TERM command.

**Example of Use**

To save the screenshot to an image file:

```
:SYST:PRIN:COPY "screen_copy_full","C:\screen_copy"
```

To query the screenshot image file:

```
:SYST:DISP:DATA?
>#541056Avdl-*;E4"as…
```

***Note:***

The end of response data cannot be detected by the terminator, because <binary_data> contains CR/LF. The procedure for obtaining the response data for this command is described below.

1. Disable the terminator detection function of the control interface.
2. Send :SYSTem:DISPlay:DATA?.
3. Read the first byte that follows "#" in the response data. It indicates the number of digits of the data size.
4. Read the bytes by the number of digits indicated. The bytes indicate the data size (byte).
5. Read the binary data of the indicated data size.
6. Read the terminator.
7. Enable the terminator detection function of the control interface.
8. Output the received binary data as-is to a file to create a screenshot image file.

Example: For binary data of 2002 bytes

#42002an%*qe4445+\...

1. Read the first byte "4" that follows "#". It indicates the data size is a four-digit number.
2. Read the four-digit string "2002" that follows "4". It indicates the data size is 2002 bytes.
3. Disable the terminator detection function of the control interface.
4. Read the binary data of 2002 bytes.
5. Output the read binary data of 2002 bytes to a file, which is called a screenshot image file.

## 3.4.8 Specifying screen display

### :DISPlay:ACTive

**Function**

This command activates the screen of the specified function.

**Syntax**

`:DISPlay:ACTive <module_id>`

**Parameter**

<module_id>

Refer to ":MODule:ID".

**Example of Use**

To activate the PPG/ED Ch1 screen:

`:DISP:ACT 1`

***Note:***

The purpose of this command is only to activate the specified screen. To specify the module you want to remotely control, execute :MODule:ID.

### :DISPlay:ACTive:ACResult

**Function**

*Added in Version 4:* This command displays all BER measurement results on the screen.

**Syntax**

`:DISPlay:ACTive:ACResult`

**Example of Use**

`:DISP:ACT:ACR`

## 3.4.9 Operations to all modules

### *OPC (Operation Complete)

**Function**

*OPC sets bit 0 of the standard event status register to be changed from 0 to 1 when execution of the message being processed has been completed.
*OPC? queries whether execution of the message being processed has been completed.

**Syntax**

```
*OPC
*OPC?
```

**Response Data**

0|1

**Example of Use**

```
*OPC?
>1
```

***Note:***

For the BERTWave, the response data for *OPC? is always 1.

Each of all the messages, including *OPC?, for the BERTWave cannot be executed, unless the previously sent message has been completed. Therefore, the response data for *OPC? is always 1 (Operation Complete).

### *WAI (Wait to Continue)

**Function**

This command holds execution of the next message until processing of the message sent before *WAI is completed.

**Syntax**

```
*WAI
```

**Example of Use**

```
*WAI
```

***Note:***

It is not required to use *WAI, because no overlapping commands are provided for the BERTWave. Each of the messages for the BERTWave cannot be executed, unless the previous message has been completed.

## :SOURce:OUTPut:ASET

### Function

This command sets and queries the On/Off state of signal outputting from all PPG channels and optical outputting from XFP/SFP+.

### Syntax

```
:SOURce:OUTPut:ASET <enable>
:SOURce:OUTPut:ASET?
```

### Parameter

0|OFF
1|ON

### Response Data

0|1
0 All outputting (PPG/Optical) Off
1 At least one outputting (PPG/Optical) On

### Example of Use

```
:SOUR:OUTP:ASET ON
:SOUR:OUTP:ASET?
>1
```

## *TRG (Trigger)

### Function

This command triggers the measurement for all modules (all ED channels and Scope).

### Syntax

```
*TRG
```

### Example of Use

```
*TRG
```

## :SENSe:MEASure:ASTP

### Function

This command stops the measurement for all modules (all ED channels and Scope).

### Syntax

```
:SENSe:MEASure:ASTP
```

### Example of Use

```
:SENS:MEAS:ASTP
```

## :SENSe:MEASure:ASTate?

### Function

This command queries the measurement status for all modules (all ED channels and Scope).

### Syntax

```
:SENSe:MEASure:ASTate?
```

### Response Data

0|1

0 Measurement stops for all modules.

1 At least one module, measurement in progress

### Example of Use

```
:SENS:MEAS:AST?
>0
```

# 3.5 Status Register

## 3.5.1 Clearing register

### *CLS (Clear Status)

**Function**

This command clears the standard event status register and output queue.

**Syntax**

```
*CLS
```

**Example of Use**

```
*CLS
```

***Notes:***

- *CLS common command clears the following registers.
  - Standard event status register
  - Output queue

  Therefore, bits 5 of status byte register became 0.

  The setting value of each enable register does not vary depending on *CLS.

  - Standard event status enable register
  - Service request enable register
  - Operation status register
  - Device dependent status register
- The *CLS common command clears the status byte register when sending *CLS command before the query after the program message terminator.
  All unread messages in the output queue are cleared at this time.

  The relevant message example indicates below.

  ```
  SENS:BIT 8500000
  *CLS;SENS:BIT?
  ```

  When receiving SENS:BIT? after *CLS, the status byte register is cleared.

### :STATus:PRESet

**Function**

This command presets the event registers and transition filters of the operation status register and device dependent (PPG/ED/XFP/SFP+/Scope) registers.

**Syntax**

```
:STATus:PRESet
```

**Example of Use**

```
:STAT:PRES
```

***Note:***

All bits in the event registers and negative transition filters are set to 0, and all bits in the positive transition filters are set to 1.

## 3.5.2 Status byte register

### *STB? (Status Byte)

**Function**

This command queries the value of the status byte register.

**Syntax**

```
*STB?
```

**Response Data**

<integer> = bit0 + bit1 + bit2 + bit3 + bit4 + bit5 + bit6 + bit7

| | |
|---|---|
| bit7 : $2^7 = 128$ | Operation status register |
| bit6 : $2^6 = 64$ | Always 0 |
| bit5 : $2^5 = 32$ | Standard event status register |
| bit4 : $2^4 = 16$ | MAV |
| bit3 : $2^3 = 8$ | Not used |
| bit2 : $2^2 = 4$ | System Alarm |
| bit1 : $2^1 = 2$ | Not used |
| bit0 : $2^0 = 1$ | Not used |

**Example of Use**

```
*STB?
>0
```

## *SRE (Service Request Enable)

### Function

This command sets and queries the value of the service request enable register.

### Syntax

```
*SRE <integer>
*SRE?
```

### Parameter

<integer> = bit0 + bit1 + bit2 + bit3 + bit4 + bit5 + bit6 + bit7
The mask bits of the status byte register are set to 0.
The meanings of the bits are the same as those of *STB?.

### Response Data

<integer>

### Example of Use

The following example shows how to mask bits 7,6,3,1 and 0 and permit bits 5, 4, and 2.

```
*SRE 52
*SRE?
>52
```

## 3.5.3 Standard event status register

### *ESR? (Standard Event Status Register)

#### Function

This command queries the value of the standard event status register.

#### Syntax

```
*ESR?
```

#### Response Data

<integer> = bit0 + bit1 + bit2 + bit3 + bit4 + bit5 + bit6 + bit7

| | |
|---|---|
| bit7: $2^7 = 128$ | Power-on |
| bit6: $2^6 = 64$ | Not used |
| bit5: $2^5 = 32$ | Command error |
| bit4: $2^4 = 16$ | Operation error |
| bit3: $2^3 = 8$ | Device Dependent error |
| bit2: $2^2 = 4$ | Not used |
| bit1: $2^1 = 2$ | Not used |
| bit0: $2^0 = 1$ | Completion of operation |

For more information about errors, refer to Appendix B “Error Codes”.

This value (sum of standard event status register, 0 to 255) is obtained by ANDing with 8 bits set by *ESE.

#### Example of Use

To query the value of the standard event status register at the time the operation error and command error have occurred:

```
*ESR?
>48
```

***Note:***

The standard event status register is cleared by executing *ESR?.

## *ESE (Event Status Enable)

### Function

This command sets and queries the value of the standard event status enable register.

### Syntax

```
*ESE <integer>
*ESE?
```

### Parameter

<integer> = bit0 + bit1 + bit2 + bit3 + bit4 + bit5 + bit6 + bit7

Set 0 to the standard event status register bit(s) when masking it (them).

The meanings of the bits are the same as those of *ESR?.

### Response Data

<integer>

### Example of Use

To mask bits 4, 5, 6 and 7 and enable bits 0, 1, 2 and 3:

```
*ESE 15
*ESE?
>15
```

## 3.5.4 Operation status register

### :STATus:OPERation:CONDition?

#### Function

This command queries the details of the operation status condition register.

#### Syntax

```
:STATus:OPERation:CONDition?
```

#### Response Data

<integer> = bit4 + bit11 + bit12

bit4: $2^4 = 16$ Measurement (ED)

bit11: $2^{11} = 2048$ Pattern Setting (PPG/ED)

#### Example of Use

```
:STAT:OPER:COND?
>16
```

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

#### Function

This command queries the operation status event register.

#### Syntax

```
:STATus:OPERation[:EVENt]?
```

#### Response Data

<integer> = bit4 + bit11 + bit12

The meanings of the bits are the same as those of :STATus:OPERation:CONDition?.

#### Example of Use

```
:STAT:OPER?
>16
```

### :STATus:OPERation:ENABle

#### Function

This command sets and queries the operation status enable register.

#### Syntax

```
:STATus:OPERation:ENABle <integer>
:STATus:OPERation:ENABle?
```

**Parameter**

<integer> = bit4 + bit11 + bit12

Specify the value for the bit(s) you want to enable.

The meanings of the bits are the same as those of :STATus:OPERation:CONDition?.

**Response Data**

<integer>

**Example of Use**

To set only bit 4 of the operation status event register to be queried:

At this time, $2^4 = 16$ is set in the operation status enable register.

```
:STAT:OPER:ENAB 16
```

To query the value of the operation status enable register:

```
:STAT:OPER:ENAB?
>16
```



## :STATus:OPERation:NTRansition

**Function**

This command sets and queries the transition filter (negative transition) of the operation status register.

**Syntax**

```
:STATus:OPERation:NTRansition <integer>
:STATus:OPERation:NTRansition?
```

**Parameter**

<integer> = bit4 + bit11 + bit12

To set the event register to 1 when the condition register has changed from 0 to 1, set the bit to 1.

The meanings of the bits are the same as those of :STATus:OPERation:CONDition?.

**Response Data**

<integer> = bit4 + bit11 + bit12

**Example of Use**

To set bit 4 of operation status event register to bit 1 when bit 4 of operation status condition register changed from 1 to 0:

At this time, $2^4 = 16$ is set to the transition filter (negative transition).)

```
:STAT:OPER:NTR 16
```

To query transition filter (negative transition) of operation status register:

```
:STAT:OPER:NTR?
>16
```

## :STATus:OPERation:PTRansition

### Function

This command sets and queries the transition filter (positive transition) of the operation status register.

### Syntax

```
:STATus:OPERation:PTRansition <integer>
:STATus:OPERation:PTRansition?
```

### Parameter

<integer> = bit4 + bit11 + bit12

If the event register is set to 1 when the condition register is changed from 0 to 1, the bit is set to 1.

The meanings of the bits are the same as those of :STATus:OPERation:CONDition?.

### Response Data

<integer> = bit4 + bit11 + bit12

### Example of Use

To set bit 11 of operation status event register to bit 1 when bit 11 of operation status condition register changes from 0 to 1:

At this time, $2^{11}$ = 2048 is set in the transition filter (positive transition).

```
:STAT:OPER:PTR 2048
```

To query transition filter (positive transition) of operation status register:

```
:STAT:OPER:PTR?
>2048
```

## 3.5.5 PPG/ED status register

### :INSTrument:PE<ch_num>:RESet

**Function**

This command initializes the PPG/ED status event register.

**Syntax**

```
:INSTrument:PE<ch_num>:RESet
```

**Parameter**

<ch_num>

Channel number of PPG/ED

Range 1 to 4

**Example of Use**

```
:INST:PE1:RES
```

### :INSTrument:PE<ch_num>:CONDition?

**Function**

This command queries the details of the PPG/ED condition register.

**Syntax**

```
:INSTrument:PE<ch_num>:CONDition?
```

**Parameter**

<ch_num>

Channel number of PPG/ED

Range 1 to 4

**Response Data**

<integer> = bit0 + bit1 + bit2 + bit3 + bit4 + bit5

| | |
|---|---|
| bit0: $2^0 = 1$ | PLL Unlock |
| bit1: $2^1 = 2$ | Total Error |
| bit2: $2^2 = 4$ | Pattern Sync Loss |
| bit3: $2^3 = 8$ | CR Unlock |
| bit4: $2^4 = 16$ | Insertion Error |
| bit5: $2^5 = 32$ | Omission Error |

**Example of Use**

```
:INST:PE1:COND?
>1
```

## :INSTrument:PE<ch_num>[:EVENt]?

**Function**

This command queries the details of the PPG/ED event register.

**Syntax**

```
:INSTrument:PE<ch_num>:[EVENt]?
```

**Parameter**

<ch_num>
Channel number of PPG/ED
Range 1 to 4

**Response Data**

<integer>
The meanings of the bits are the same as those of :INSTrument:PE<ch_num>:CONDition?.

**Example of Use**

```
:INST:PE1?
>1
```

## :INSTrument:PE<ch_num>:NTRansition

**Function**

This command sets and queries the transition filter (negative transition) of the PPG/ED status.

**Syntax**

```
:INSTrument:PE<ch_num>:NTRansition <integer>
:INSTrument:PE<ch_num>:NTRansition?
```

**Parameter**

<ch_num>
Channel number of PPG/ED
Range 1 to 4
<integer>
To set the event register to 1 when the condition register has changed from 0 to 1, set the bit to 1.
The meanings of the bits are the same as those of :INSTrument:PE<ch_num>:CONDition?.

**Response Data**

<integer>

**Example of Use**

```
:INST:PE1:NTR 15
:INST:PE1:NTR?
>15
```

## :INSTrument:PE<ch_num>:PTRansition

**Function**

This command sets and queries the transition filter (positive transition) of the PPG/ED status.

**Syntax**

```
:INSTrument:PE<ch_num>:PTRansition <integer>
:INSTrument:PE<ch_num>:PTRansition?
```

**Parameter**

<ch_num>

Channel number of PPG/ED

Range 1 to 4

<integer>

To set the event register to 1 when the condition register has changed from 0 to 1, set the bit to 1.

The meanings of the bits are the same as those of :INSTrument:PE<ch_num>:CONDition?.

**Response Data**

<integer>

**Example of Use**

```
:INST:PE1:PTR?
>3
```

## 3.5.6 Scope status register

### :INSTrument:WAV:RESet

**Function**

This command initializes the Scope status event register.

**Syntax**

```
:INSTrument:WAV:RESet
```

### :INSTrument:WAV:CONDition?

**Function**

This command queries the details of the Scope status condition register.

**Syntax**

```
:INSTrument:WAV:CONDition?
```

**Response Data**

<integer>

Total value of the condition register bits

The correspondence between register bits and decimal numbers is described below:

| Valid bit | Description |
|---|---|
| 1 Bit0 | CAL Alarm (Yellow: Temperature deviation of ±2.5°C or more) |
| 2 Bit1 | CAL Alarm (Red: Temperature deviation of ±5.0°C or more) |
| 4 Bit2 | CAL Alarm (Orange: Different sampling rate) |
| 8 Bit3 | Not used |
| 16 Bit4 | PLL Unlock |

**Example of Use**

```
:INST:WAV:COND?
>1
```

### :INSTrument:WAV[:EVENt]?

**Function**

This command queries the details of the Scope status event register.

**Syntax**

```
:INSTrument:WAV:[EVENt]?
```

**Response Data**

<integer>

Total value of the event register bits

Range 0 to 7, 16 to 23

The correspondence between register bits and decimal numbers is described below:

| Valid bit | Description |
|---|---|
| 1 Bit0 | CAL Alarm (Yellow: Temperature deviation of ±2.5°C or more) |
| 2 Bit1 | CAL Alarm (Red: Temperature deviation of ±5.0°C or more) |
| 4 Bit2 | CAL Alarm (Orange: Different sampling rate) |
| 8 Bit3 | Not used |
| 16 Bit4 | PLL Unlock |

#### Example of Use

```
:INST:WAV?
>1
```

### :INSTrument:WAV:NTRansition

#### Function

This command sets and queries the transition filter (negative transition) of the Scope status register.

#### Syntax

```
:INSTrument:WAV:NTRansition <integer>
:INSTrument:WAV:NTRansition?
```

#### Parameter

<integer>

To set the event register to 1 when the condition register has changed from 0 to 1, set the bit to 1.

The meanings of the bits are the same as those of :INSTrument:WAV:CONDition?.

#### Response Data

<integer>

#### Example of Use

```
:INST:WAV:NTR?
>1
```

### :INSTrument:WAV:PTRansition

**Function**

This command sets and queries the transition filter (positive transition) of the Scope status register.

**Syntax**

```
:INSTrument:WAV:PTRansition <integer>
:INSTrument:WAV:PTRansition?
```

**Parameter**

<integer>

If the event register is set to 1 when the condition register is changed from 0 to 1, the bit is set to 1.

The meanings of the bits are the same as those of :INSTrument:WAV:CONDition?.

**Response Data**

<integer>

**Example of Use**

```
:INST:WAV:PTR?
>3
```

## 3.5.7 XFP/SFP+ status register

### :INSTrument:XSFP:RESet

**Function**

This command initializes the XFP/SFP+ status event register.

**Syntax**

```
:INSTrument:XSFP:RESet
```

### :INSTrument:XSFP:CONDition?

**Function**

This command queries the condition register details on the XFP/SFP+ status register.

**Syntax**

```
:INSTrument:XSFP:CONDition?
```

**Response Data**

<integer> = bit0 + bit1

bit0: $2^0 = 1$ Ready
bit1: $2^1 = 2$ LOS

**Example of Use**

```
:INST:XSFP:COND?
>0
```

## :INSTrument:XSFP[:EVENt]?

**Function**

This command queries the details on the XFP/SFP+ status event register.

**Syntax**

```
:INSTrument:XSFP:[EVENt]?
```

**Response Data**

<integer>
The meanings of the bits are the same as those of :INSTrument:XSFP:CONDition?.

**Example of Use**

```
:INST:XSFP?
>0
```

## :INSTrument:XSFP:NTRansition

**Function**

This command sets and queries the transition filter (negative transition) of the XFP/SFP+ status.

**Syntax**

```
:INSTrument:XSFP:NTRansition <integer>
:INSTrument:XSFP:NTRansition?
```

**Parameter**

<integer>
To set the event register to 1 when the condition register has changed from 0 to 1, set the bit to 1.
The meanings of the bits are the same as those of :INSTrument:XSFP:CONDition?.

**Response Data**

<integer>

**Example of Use**

```
:INST:XSFP:NTR?
>3
```

## :INStrument:XSFP:PTRansition

**Function**

This command sets and queries the transition filter (positive transition) of the XFP/SFP+ status.

**Syntax**

```
:INSTrument:XSFP:PTRansition <integer>
:INSTrument:XSFP:PTRansition?
```

**Parameter**

<integer>

If the event register is set to 1 when the condition register is changed from 0 to 1, the bit is set to 1.

The meanings of the bits are the same as those of :INSTrument:XSFP:CONDition?.

**Response Data**

<integer>

**Example of Use**

```
:INST:XSFP:PTR?
>3
```

# 3.6 PPG/ED messages

## 3.6.1 Common commands

### :OUTPut:RCLock

**Function**

*Added in Version 4:* This command queries the Reference CLK for PPG/ED.

**Syntax**

```
:OUTPut:RCLock INT|EXT10M|EXT1_16|CH1
:OUTPut:RCLock?
```

**Parameter**

| | |
|---|---|
| INT | Internal 10MHz |
| EXT10ME | 10MHz In |
| EXT1_16 | Ext 1/16 In |
| CH1 | Dependent on the Reference Clock for PPG1<br>(Available if the remote control target is Ch2.) |

**Response Data**

INT|EXT10M|EXT1_16|CH1

**Example of Use**

```
:OUTP:RCL INT
:OUTP:RCL?
>INT
```

***Note:***

If :MODule:ID is set to Ch3 or Ch4, execution of the command results in an error (–220 Parameter error), because the clock of Ch3/4 is dependent on Ch1/2.

### :OUTPut:RCLock:SELect

**Function**

This command is compatible with version 4 or later, and sets and queries which clock (internal or external) is used as Reference CLK.

**Syntax**

```
:OUTPut:RCLock:SELect
INTernal|EXTernal|CH1External|CH2External|SYNChronize
:OUTPut:RCLock:SELect?
```

3

Message List

**Parameter**

| | |
|---|---|
| INTernal | Ch1/2: Internal clock |
| EXTernal | Ch1/2: External clock |
| CH1External | Ch1: External clock, Ch2: Internal clock |
| CH2External | Ch1:Internal clock, Ch2: External clock |
| SYNChronize | Ch1: Internal clock,<br>Ch2: Dependent on Reference Clock for PPG1 |

**Response Data**

INT|EXT|CH1E|CH2E|SYNC

**Example of Use**

```
:OUTP:RCL:SEL INT
:OUTP:RCL:SEL?
>INT
```

***Notes:***

- For MP210xA, SYNChronize is available only when the Option 052 is installed. (For MP2100B, it is available even when the Option 052 is not installed.)
- If :MODule:ID is set to Ch3 or Ch4, execution of the command results in an error (–220 Parameter error), because the bit rate of Ch3/4 is dependent on Ch1/2.

## :OUTPut:RCLock:STATus?

**Function**

This command queries the state of the Reference CLK status indicator.

**Syntax**

```
:OUTPut:RCLock:EXTernal:STATus?
```

**Response Data**

NONE|UNLOCK|LOCKED

| | |
|---|---|
| NONE | Red |
| NOT_READY | Yellow |
| READY | Green |

**Example of Use**

```
:OUTP:RCL:STAT?
>READY
```

## :OUTPut:RCLock:APPLy

### Function

This command starts synchronization of the input clock when the external clock is used as Reference CLK.
This command is available when the response to :OUTPut:RCLock:EXTernal:STATus? is NOT_READY.

### Syntax

```
:OUTPut:RCLock:APPLy
```

## :OUTPut:CMU:EXTClock

### Function

This command is compatible with version 4 or later, and sets and queries the connector to input Reference CLK when the external clock is used as Reference CLK.

### Syntax

```
:OUTPut:CMU:EXTClock 10M|1_16
:OUTPut:CMU:EXTClock?
```

### Parameter

| | |
|---|---|
| 10M | Ext 10MHz In |
| 1_16 | Ext 1/16 In |

### Response Data

10M|1_16

### Example of Use

To set the **Ext Clk In** connector for inputting an external clock:

```
:OUTP:CMU:EXTC 1_16
:OUTP:CMU:EXTC?
>1_16
```

## :OUTPut:SYNC:SOURce

### Function

This command sets and queries the signal source to be output to the **Sync Out** connector.

### Syntax

```
:OUTPut:SYNC:SOURce <character>
:OUTPut:SYNC:SOURce?
```

**Parameter**

<character>

For PPG Pattern Sync: PPG{1|2|3|4}PATT

When using PPG as the signal source: PPG{1|2}CLOC{1|2|4|8|16|64}

When using ED as the signal source: ED{1|2|3|4}CLOC{4|8|16}

**Response Data**

PPG{1|2|3|4}PATT

PPG{1|2}CLOC{1|2|4|8|16|64}

ED{1|2|3|4}CLOC{4|8|16}

**Example of Use**

To set the 1/16 divided clock that synchronizes with the data output from PPG1, as the signal to be output to the **Sync Out** connector:

```
:OUTP:SYNC:SOUR PPG1CLOC16
:OUTP:SYNC:SOUR?
>PPG1CLOC16
```

## :BERT:ALL:PARam:TRACking

**Function**

*Added in Version 4:* This command sets and queries the On/Off state of Ch Tracking.

**Syntax**

```
:BERT:ALL:PARam:TRACking <enable>
:BERT:ALL:PARam:TRACking?
```

**Parameter**

0|OFF

1|ON

**Response Data**

0|1

**Example of Use**

```
:BERT:ALL:PAR:TRAC ON
:BERT:ALL:PAR:TRAC?
>1
```

***Notes:***

- If Reference CLK for PPG2 is set to other than Ch1 when Ch Tracking is set to On, an error (–220 Parameter error) occurs.
- If Ch Tracking is set to On, ED Tracking is set to On as well.

## :SENSe:PARam:TRACking

### Function

This command sets and queries the On/Off state of ED Tracking.

### Syntax

```
:SENSe:PARam:TRACking <enable>
:SENSe:PARam:TRACking?
```

### Parameter

0|OFF
1|ON

### Response Data

0|1

### Example of Use

```
:SENS:PAR:TRAC ON
:SENS:PAR:TRAC?
>1
```

## :OUTPut:BITRate:STANdard

### Function

This command sets and queries the bit rate standard for the PPG.

### Syntax

```
:OUTPut:BITRate:STANdard <bitrate_standard>
:OUTPut:BITRate:STANdard?
```

### Parameter

<bitrate_standard> = <string>
The following strings can be used for <bitrate_standard>.

Table 3.6.1-1 Bit Rate Standards

| <string> | Standard | Bit Rate (bit/s) | Remarks |
|---|---|---|---|
| "VARIABLE" | Variable-1/1 | 6.25G to 12.5G | |
| "1G_FC" | 1GFC | 1.0625G | *1 |
| "2G_FC" | 2GFC | 2.125G | *1 |
| "4G_FC" | 4GFC | 4.25G | |
| "8G_FC" | 8GFC | 8.5G | |
| "10G_FC" | 10GFC | 10.518G | |
| "10G_FC_FEC" | 10GFC FEC | 11.3168G | |
| "1GBE" | 1GbE | 1.25G | *1 |
| "2GBE" | 2GbE | 2.5G | *1 |
| "INF" | Infiniband | 2.5G | *1 |
| "10G_WAN" | 10GbE WAN | 9.95328G | |
| "10G_LAN" | 10GbE LAN/PHY | 10.3125G | |
| "10G_OTU1E" | 10GbE OTU1e | 11.049G | |
| "10G_OTU2E" | 10GbE OTU2e | 11.095G | |
| "OC-3" | OC-3/STM-1 | 155.22M | *1 |
| "OC-12" | OC-12/STM-4 | 622.08M | *1 |
| "OC-48" | OC-48/STM16 | 2.488G | *1 |
| "OTU-1" | OTU-1 | 2.666057G | *1 |
| "OC-192" | OC-192/STM-64 | 9.95328G | |
| "OC-192FEC" | G.975 FEC | 10.664G | |
| "VARIABLE-1/2" | Variable-1/2 | 6.25G to 3.125G | |
| "VARIABLE-1/4" | Variable-1/4 | 3.125G to 1.5625G | *1 |
| "VARIABLE-1/8" | Variable-1/8 | 1.5625G to 781.25M | *1 |
| "VARIABLE-1/16" | Variable-1/16 | 781.25M to 390.625M | *1 |
| "VARIABLE-1/32" | Variable-1/32 | 390.625M to 195.312M | *1 |
| "VARIABLE-1/64" | Variable-1/64 | 195.312M to 125M | *1 |
| "INF5G" | Infiniband x2 | 5G | |
| "INF10G" | Infiniband x4 | 10G | |
| "OC-24" | OC-24 | 1.244G | *1 |
| "CPRI" | CPRI | 614.4M | *1 |

*1: If the Option 090 is not installed, the bit rate that is equal to or less than Variable-1/4 cannot be selected for the ED.

**Table 3.6.1-1 Bit Rate Standards (Cont'd)**

| <string> | Standard | Bit Rate | Remarks |
|---|---|---|---|
| "CPRI-2" | CPRI x2 | 1.2288G | *1 |
| "CPRI-4" | CPRI x4 | 2.4576G | *1 |
| "CPRI-5" | CPRI x5 | 3.072G | *1, *2 |
| "CPRI-10" | CPRI x10 | 6.144G | *2 |
| "OBSAIRP3" | OBSAI RP3 | 768M | *1, *2 |
| "OBSAIRP3-2" | OBSAI RP3 x2 | 1.536G | *1, *2 |
| "OBSAIRP3-4" | OBSAI RP3 x4 | 3.072G | *1, *2 |
| "OBSAIRP3-8" | OBSAI RP3 x8 | 6.144G | *2 |

*2: If the Option 090/092 is not installed, the bit rate cannot be selected because the bit rate range is limited.





**Response Data**

<bitrate_standard> = <string>

**Example of Use**

```
:OUTP:BITR:STAN "10G_LAN"
:OUTP:BITR:STAN?
>"10G_LAN"
```

***Note:***

*Added in Version 4:* If :MODule:ID is set to Ch3 or Ch4, execution of the command results in an error (–220 Parameter error), because the bit rate of Ch3/4 is dependent on Ch1/2.

## :INPut:BITRate:STANdard

**Function**

This command is compatible with version 4 or later, and sets and queries the bit rate standard for the ED.

**Syntax**

```
:INPut:BITRate:STANdard <bitrate_standard>
:INPut:BITRate:STANdard?
```

**Parameter**

<bitrate_standard>

Refer to Table 3.6.1-1 "Bit Rate Standards".

**Response Data**

<bitrate_standard>

**Example of Use**

```
:INP:BITR:STAN "10G_LAN"
:INP:BITR:STAN?
>"10G_LAN"
```

***Note:***

*Modified in Version 4:* If the BERTWave you are using is MP2100B, execution of the command results in an error (–220 Parameter error), because the bit rate of the ED is dependent on the PPG.

## :OUTPut:BITRate

**Function**

This command sets and queries the bit rate of the PPG if the bit rate standard is Variable.

**Syntax**

```
:OUTPut:BITRate <bitrate>
:OUTPut:BITRate?
```

**Parameter**

<bitrate> = <numeric>

Range 12500000 to 125000, 1 kbit/s step

If the Option 092 is not installed, the range is limited as follows.

**Table 3.6.1-2 Bit Rate Range**

| <bitrate_standard> | | With the Option 090 Installed (kbit/s) | Without the Options 090 and 092 Installed (kbit/s) |
|---|---|---|---|
| Variable-(1/1) | (6.250001G to 12.5G) | 8000000 to 12500000 | 8500000 to 11320000 |
| Variable-1/2 | (3.125001G to 6.25G) | 4000000 to 6250000 | 4250000 to 5660000 |
| Variable-1/4 | (1.562501G to 3.125G) | 2000000 to 3125000 | 2125000 to 2830000 |
| Variable-1/8 | (781.251M to 1.5625G) | 1000000 to 1562500 | 1062500 to 1415000 |
| Variable-1/16 | (390.626M to 781.25M) | 500000 to 781250 | 531250 to 707500 |
| Variable-1/32 | (195.313M to 390.625M) | 250000 to 390625 | 265625 to 353750 |
| Variable-1/64 | (125M to 195.312M) | 125000 to 195312 | 132813 to 176875 |

**Response Data**

<bitrate> = <numeric>

**Example of Use**

```
:OUTP:BITR:STAN "VARIABLE"
:OUTP:BITR 8500000
:OUTP:BITR?
>8500000
```

***Note:***

*Added in Version 4:* If :MODule:ID is set to Ch3 or Ch4, execution of the command results in an error (–220 Parameter error), because the bit rate of Ch3/4 is dependent on Ch1/2.



## :INPut:BITRate

**Function**

This command is compatible with version 4 or later, and sets and queries the bit rate of the ED if the bit rate standard is Variable.

**Syntax**

```
:INPut:BITRate <bitrate>
:INPut:BITRate?
```

**Parameter**

<bitrate> = <numeric>

**Response Data**

<bitrate> = <integer>

**Example of Use**

```
:INP:BITR:STAN "VARIABLE"
:INP:BITR 8500000
:INP:BITR?
>8500000
```

***Note:***

*Modified in Version 4:* If the BERTWave you are using is MP2100B, execution of the command results in an error (–220 Parameter error), because the bit rate of the ED is dependent on the PPG.

## :OUTPut:CMU:RESolution

**Function**

This command is compatible with existing products, and sets and queries the unit of bit rate specified by the :OUTPut:CMU:FREQuency command.

**Syntax**

```
:OUTPut:CMU:RESolution KHZ|MHZ
:OUTPut:CMU:RESolution?
```

**Parameter**

KHZ|MHZ

**Response Data**

KHZ|MHZ

**Example of Use**

```
:OUTP:CMU:RES KHZ
:OUTP:CMU:RES?
>KHZ
```

***Note:***

Even if MHZ is specified by this command, the Bit Rate display in the screen will not change.

## :OUTPut:BITRate:OFFSet

**Function**

This command sets and queries the bit rate offset for the PPG.

**Syntax**

```
:OUTPut:BITRate:OFFSet <numeric>
:OUTPut:BITRate:OFFSet?
```

**Parameter**

<numeric>

Range –100 to 100, 1 ppm step

**Response Data**

<integer>

**Example of Use**

```
:OUTP:BITR:OFFS 100
:OUTP:BITR:OFFS?
>100
```

***Note:***

If :MODule:ID is set to Ch3/4, execution of this command results in an error (–220 Parameter error) because the bit rate of Ch3/4 is dependent on Ch1/2.

## :OUTPut:BITRate:DIVRate

### Function

This command sets and queries the clock divide ratio (1/n) of the PPG.

### Syntax

```
:OUTPut:BITRate:DIVRate <character>
:OUTPut:BITRate:DIVRate?
```

### Parameter

<character>

1_{1|2|4|8|16|32|64}

### Response Data

1_{1|2|4|8|16|32|64}

### Example of Use

To set the clock divide ratio of the PPG to 1/2:

```
:OUTP:BITR:DIVR 1_8
:OUTP:BITR:DIVR?
>1_8
```

***Note:***

This command is executed when the Reference CLK is Ext 1/16 In.

## :INPut:BITRate:DIVRate?

### Function

This command queries the clock divide ratio (1/n) of the error detector.

### Syntax

```
:INPut:BITRate:DIVRate?
```

### Response Data

1_{1|2|4|8|16|32|64}

### Example of Use

```
:INP:BITR:DIVR?
>1_2
```

## :SOURce:PATTern:TYPE

### Function

This command sets and queries the test pattern of the PPG.

**Syntax**

```
:SOURce:PATTern:TYPE <character>
:SOURce:PATTern:TYPE?
```

**Parameter**

<character> = PRBS{7|9|15|23|31}|USER

**Response Data**

PRBS{7|9|15|23|31}|USER

**Example of Use**

To set the test pattern of the PPG to PRBS2^23–1:

```
:SOUR:PATT:TYPE PRBS23
:SOUR:PATT:TYPE?
>PRBS23
```

To set the test pattern of the PPG to ProgrammablePattern

```
:SOUR:PATT:TYPE USER
:SOUR:PATT:TYPE?
>USER
```

## :SENSe:PATTern:TYPE

**Function**

This command sets and queries the test pattern of the ED.

**Syntax**

```
:SENSe:PATTern:TYPE <character>
:SENSe:PATTern:TYPE?
```

**Parameter**

<character> = PRBS{7|9|15|23|31}|USER

**Response Data**

PRBS{7|9|15|23|31}|USER

**Example of Use**

To set the test pattern of the ED to PRBS2^23–1:

```
:SENS:PATT:TYPE PRBS23
:SENS:PATT:TYPE?
>PRBS23
```

To set the test pattern of the ED to ProgrammablePattern

```
:SENS:PATT:TYPE USER
:SENS:PATT:TYPE?
>USER
```

***Note:***

If ED Tracking is set to On, this command results in an error.

## :SOURce:PATTern:LOGic

**Function**

This command sets and queries the test pattern logic (positive/negative logic) of the PPG.

**Syntax**

```
:SOURce:PATTern:LOGic POSitive|NEGative
:SOURce:PATTern:LOGic?
```

**Parameter**

POSitive|NEGative

**Response Data**

POS|NEG

**Example of Use**

```
:SOUR:PATT:LOG POS
:SOUR:PATT:LOG?
>POS
```

## :SENSe:PATTern:LOGic

**Function**

This command sets and queries pattern logic (negative/positive logic) of the ED.

**Syntax**

```
:SENSe:PATTern:LOGic POSitive|NEGative
:SENSe:PATTern:LOGic?
```

**Parameter**

POSitive|NEGative

**Response Data**

POS|NEG

**Example of Use**

```
:SENS:PATT:LOG POS
:SENS:PATT:LOG?
>POS
```

## :SOURce:MMEMory:PATTern:RECall

**Function**

This command sets the programmable pattern file for the PPG.

**Syntax**

```
:SOURce:MMEMory:PATTern:RECall <file_name>,{BIN|TXT}
```

**Parameter**

<file_name>
Name of pattern file in the following folder (including file extension)
C:\Program Files\Anritsu\MP2100A\MX210000A\UserData\Pattern

BIN|TXT
BIN Binary file
TXT Text file

**Example of Use**

```
:SOUR:PATT:TYPE USER
:SOUR:MMEM:PATT:REC "10101010.dat",BIN
```

## :SENSe:MMEMory:PATTern:RECall

**Function**

This command sets the programmable pattern file for the ED.

**Syntax**

```
:SENSe:MMEMory:PATTern:RECall <file_name>,BIN|TXT
```

**Parameter**

Same as :SOURce:MMEMory:PATTern:RECall.

**Example of Use**

```
:SENS:PATT:TYPE USER
:SENS:MMEM:PATT:REC "10101010.dat",BIN
```

***Note:***

If ED Tracking is set to On, this command results in an error.

## :SOURce:PATTern:DATA:LENGth?

**Function**

This command queries the pattern length when the test pattern of the PPG is Programmable Pattern.

**Syntax**

`:SOURce:PATTern:DATA:LENGth?`

**Response Data**

<integer>

2 to 1305600

**Example of Use**

```
:SOUR:PATT:DATA:LENG?
>16384
```

## :SENSe:PATTern:DATA:LENGth?

**Function**

This command queries the pattern length when the test pattern of the ED is Programmable Pattern.

**Syntax**

`:SENSe:PATTern:DATA:LENGth?`

**Response Data**

Same as :SOURce:PATTern:DATA:LENGth?.

**Example of Use**

```
:SENS:PATT:DATA:LENG?
>16384
```

## 3.6.2 PPG-specific commands

### :OUTPut:DATA:OUTPut

**Function**

This command sets and queries the On/Off state of PPG signal output.

**Syntax**

```
:OUTPut:DATA:OUTPut <enable>
:OUTPut:DATA:OUTPut?
```

**Parameter**

0|OFF
1|ON

**Response Data**

0|1

**Example of Use**

```
:OUTP:DATA:OUTP ON
:OUTP:DATA:OUTP?
>1
```

***Note:***

To turn on and off the signal output from all channels, use :SOUR:OUTP:ASET.

### :OUTPut:DATA:AMPLitude

**Function**

This command sets and queries the amplitude voltage of the signal to be output from the **Data Out** and $\overline{\text{Data}}$ **Out** connectors of PPG.

**Syntax**

```
:OUTPut:DATA:AMPLitude DATA,<numeric>
:OUTPut:DATA:AMPLitude? DATA
```

**Parameter**

<numeric>

Range 0.10 to 0.80, 0.01 Vpp step

**Response Data**

<numeric>

**Example of Use**

To set the output amplitude of the PPG to 0.5 Vpp:

```
:OUTP:DATA:AMPL DATA,0.5
:OUTP:DATA:AMPL? DATA
>0.5
```

## :OUTPut:DATA:ATTFactor

**Function**

This command sets and queries the External Attenuator Factor value of PPG.

**Syntax**

```
:OUTPut:DATA:ATTFac tor DATA,<numeric>
:OUTPut:DATA:ATTFactor? DATA
```

**Parameter**

<numeric>

Range 0 to 30, 1 dB step

**Response Data**

<integer>

**Example of Use**

```
:OUTP:DATA:ATTF DATA,20
:OUTP:DATA:ATTF? DATA
>20
```

## :OUTPut:DATA:RELative?

**Function**

This command queries the Relative value displayed on PPG panel.

**Syntax**

```
:OUTPut:DATA:RELative? DATA
```

**Response Data**

<numeric>

Range 0.00 to 0.80, 0.01 Vpp step

**Example of Use**

```
:OUTP:DATA:REL? DATA
>0.4
```

## :SOURce:PATTern:EADDition:SET

**Function**

This command sets and queries whether to add a bit error(s) to the test pattern to be generated by PPG.

**Syntax**

```
:SOURce:PATTern:EADDition:SET <enable>
:SOURce:PATTern:EADDition:SET?
```

**Parameter**

0|OFF
1|ON

**Response Data**

0|1

**Example of Use**

```
:SOUR:PATT:EADD:SET ON
:SOUR:PATT:EADD:SET?
>1
```

## :SOURce:PATTern:EADDition:VARiation

**Function**

This command sets and queries the error addition mode (Repeat/Single) for PPG.

**Syntax**

```
:SOURce:PATTern:EADDition:VARiation REPeat|SINGle
:SOURce:PATTern:EADDition:VARiation?
```

**Parameter**

REPeat|SINGle

**Response Data**

REP|SING

**Example of Use**

```
:SOUR:PATT:EADD:VAR REP
:SOUR:PATT:EADD:VAR?
>REP
```

## :SOURce:PATTern:EADDition:SINGle

### Function

This command generates a single error in the test pattern when the error addition mode for PPG is Single.

### Syntax

```
:SOURce:PATTern:EADDition:SINGle
```

### Example of Use

```
:SOUR:PATT:EADD:SING
```

## :SOURce:PATTern:EADDition:RATE

### Function

This command sets and queries the rate of adding bit errors when the error addition mode for PPG is Repeat.

### Syntax

```
:SOURce:PATTern:EADDition:RATE <character>[,1]
:SOURce:PATTern:EADDition:RATE?
```

### Parameter

<character>
E_{2|3|4|5|6|7|8|9|10|11|12}
Exponent of error addition rate (2 to 12)

[,1]
Indicates a mantissa of the error addition rate is 1.

### Response Data

<character>,1

### Example of Use

To set the error addition rate to 1E–9:

```
:SOUR:PATT:EADD:RATE E_9,1
:SOUR:PATT:EADD:RATE?
>E_9,1
```

## 3.6.3 ED-specific configuration commands

### :INPut:DATA:INTerface

**Function**

This command sets and queries the connector inputting signal into the ED.

**Syntax**

```
:INPut:DATA:INTerface DATA|DIFF|OPT|XDATa
:INPut:DATA:INTerface?
```

**Parameter**

| | |
|---|---|
| DATA | Electrical Single-Ended Data (Inputting to the **Data In** connector) |
| DIFF | Differential 50 Ohm (Inputting to the **Data In** and $\overline{\text{Data}}$ **In** connectors) |
| OPT | Optical (Inputting to the **O/E Data In** connector) |
| XDATa | Electrical Single-Ended $\overline{\text{Data}}$ (Inputting to the $\overline{\text{Data}}$ **In** connector) |

**Response Data**

DATA | DIFF | OPT | XDAT

**Example of Use**

```
:INP:DATA:INT DATA
:INP:DATA:INT?
>DATA
```

***Notes:***

- OPT is available if the Option 003 or 023 is installed.
- XDATa and DIFF cannot be selected for Ch1 of the Option 003/007.

### :INPut:DATA:ATTFactor

**Function**

This command sets and queries the external attenuation factor for the ED in dB unit.

**Syntax**

```
:INPut:DATA:ATTFactor DATA,<numeric>
:INPut:DATA:ATTFactor? DATA
```

**Parameter**

<numeric>

Range 0 to 30, 1 dB step

**Response Data**

<integer>

**Example of Use**

```
:INP:DATA:ATTF DATA,10
:INP:DATA:ATTF? DATA
>10
```

## :INPut:DATA:THReshold

**Function**

This command sets and queries the input threshold for ED, in mV units.

**Syntax**

```
:INPut:DATA:THReshold <numeric>
:INPut:DATA:THReshold?
```

**Parameter**

<numeric>

The range of the input threshold, assuming External ATT is A (dB), is as follows:

Range $-85*10^{\frac{A}{20}}$ to $85*10^{\frac{A}{20}}$, $10^{\frac{A}{20}}$ mV step

**Response Data**

<integer>

***Notes:***

- The setting needs to be made according to the specified step.
- The fractional portions of minimum and maximum values are truncated, and the step is rounded off to the nearest integer.

**Example of Use**

```
:INP:DATA:ATTF DATA,10
:INP:DATA:THR -270
:INP:DATA:THR?
>-270
```

## :SENSe:PATTern:SYNC:ASYNc

**Function**

This command sets and queries the On/Off state of Auto SYNC (auto pattern resynchronization processing) of ED.

**Syntax**

```
:SENSe:PATTern:SYNC:ASYNc <enable>
:SENSe:PATTern:SYNC:ASYNc?
```

**Parameter**

0|OFF
1|ON

**Response Data**

0|1

**Example of Use**

```
:SENS:PATT:SYNC:ASYN ON
:SENS:PATT:SYNC:ASYN?
>1
```

## :SENSe:PATTern:SYNC:THReshold

**Function**

This command sets and queries the threshold for Auto SYNC of ED.

**Syntax**

```
:SENSe:PATTern:SYNC:THReshold <character>
:SENSe:PATTern:SYNC:THReshold?
```

**Parameter**

<character>
INT or E_{2|3|4|5|6|7|8} (1E–2 to 1E–8)

**Response Data**

<character>
INT|E_{2|3|4|5|6|7|8}

**Example of Use**

```
:SENS:PATT:SYNC:THR E_2
:SENS:PATT:SYNC:THR?
>E_2
```

### :SENSe:PATTern:SYNC:PSMode

**Function**

This command sets and queries the On/Off state of SYNC Control when the test pattern for ED is Programmable Pattern.

**Syntax**

```
:SENSe:PATTern:SYNC:PSMode FRAMe|NORMal
:SENSe:PATTern:SYNC:PSMode?
```

**Parameter**

| | |
|---|---|
| FRAMe | SYNC Control On |
| NORMal | SYNC Control Off |

**Response Data**

FRAM|NORM

**Example of Use**

To set SYNC Control to On:

```
:SENS:PATT:SYNC:PSM FRAM
:SENS:PATT:SYNC:PSM?
>FRAM
```

### :SENSe:PATTern:SYNC:FPOSition

**Function**

This command sets and queries the frame position when the test pattern for ED is Programmable Pattern and Sync Control is On.

**Syntax**

```
:SENSe:PATTern:SYNC:FPOSition <numeric>
:SENSe:PATTern:SYNC:FPOSition?
```

**Parameter**

<numeric>

| | |
|---|---|
| Range | 1 to <Data Length> –64, 1 bit step |

**Response Data**

<integer>

**Example of Use**

```
:SENS:PATT:SYNC:FPOS 1
:SENS:PATT:SYNC:FPOS?
>1
```

## 3.6.4 ED status-related commands

### [:BERT:ALL]:DISPlay:RESult:EALarm:HRESet

**Function**

This command resets the error alarm history of ED.

**Syntax**

```
[:BERT:ALL]:DISPlay:RESult:EALarm:HRESet
```

**Example of Use**

To reset the error alarm history of the channel specified by :MODule:ID:

```
:DISP:RES:EAL:HRES
```

*Added in Version 4:* To reset the error alarm history of all ED channels:

```
:BERT:ALL:DISP:RES:EAL:HRES
```

### [:BERT:ALL]:CALCulate:DATA:MONitor?

**Function**

This command queries if the specified error/alarm has occurred at ED.

**Syntax**

```
[:BERT:ALL]:CALCulate:DATA:MONitor?
"BIT:TOTal"|"CRUNlock"|"PSLoss"
```

**Parameter**

| | |
|---|---|
| "BIT:TOTal" | Bit Error |
| "CRUNlock" | CR Unlock |
| "PSLoss" | SYNC Loss |

**Response Data**

| | |
|---|---|
| "Occur" | Error/alarm has occurred. |
| "Not Occur" | Error/alarm has not occurred. |

**Example of Use**

To query if Bit Error has occurred at the channel specified by :MODule:ID:

```
:CALC:DATA:MON? "BIT:TOT"
>"Occur"
```

*Added in Version 4:* If :BERT:ALL is specified in the header, the data on all channels are returned in the order of Ch1, Ch2, Ch3, and Ch4.

```
:BERT:ALL:CALC:DATA:MON? "BIT:TOT"
>"Occur","Occur","Occur","Occur"
```

## 3.6.5 ED fast measurement command

### [:BERT:ALL]:SENSe:MEASure:IMMediate?

**Function**

*Added in Version 4:* This command immediately executes BER measurement and returns results.

**Syntax**

```
[:BERT:ALL]:SENSe:MEASure:IMMediate? <time>[,<item>]
```

**Parameter**

<time>

Measurement time

Range 10 to 3000 step, 10 ms

<item>

Measurement items

| | |
|---|---|
| "ER:TOTal" | Total Bit Error Rate |
| "EC:TOTal" | Total Bit Error Count |
| "PSLoss" | SYNC Loss |

**Response Data**

If <item> is specified, only the specified measurement item(s) is returned. If <item> is omitted, results of all the measurement items are returned in the order of Total ER, Total EC, and SYNC Loss, as a comma-separated string.

If :BERT:ALL is specified in the header, the data of all channels are returned in the order of Ch1, Ch2, Ch3, and Ch4.

Response data for each measurement item is returned in the following format:

- "ER:TOTal": "0.0001E-18" to "1.0000E-00" (Form2)
- "EC:TOTal": "0" to "9999999" or "1.0000E+07" to "9.9999E+17" (Form1)
- "PSLoss": "Not Occur" or "Occur"
- "-------" is returned if there is no data to return.

**Example of Use**

To perform 10 ms measurement on the specified channel, and query the result of Bit Error Rate:

```
:SENS:MEAS:IMM? 10,"ER:TOTal"
>"1.0000E-02"
```

To perform 1 s measurement on the specified channel, and query the results of all measurement items:

```
:SENS:MEAS:IMM? 1000
>"1.0000E-02","850001","Not Occur"
```

To perform 10 ms measurement on all channels, and query the results of Bit Error Rate (The response example below is for the BERTWave that consists of two channels.):

```
:ALL:SENS:MEAS:IMM? 10,"ER:TOTal"
>"1.0000E-02","1.0000E-02"
```

To perform 1 s measurement on all channels, and query the results of all measurement items (The response example below is for the BERTWave that consists of two channels.):

```
:ALL:SENS:MEAS:IMM? 1000
>"1.0000E-02","850001","Not
Occur","1.0000E-02","850001","Not Occur"
```

***Notes:***

- This command is effective in reducing the time required to perform a measurement sequence. (The shorter measurement period than GUI can be specified. The measurement sequence can automatically be started and stopped. Measurement results of multiple channels can be queried at a time.)
- This command is available on MP2100B only, but not available on MP210xA.
- Execution of this command stops measurement on all ED channels.
- Measurement results of this command are not displayed in the PPG/ED screen, because fast processing is performed. The screen display of the scope is updated.

## 3.6.6 ED measurement commands

### [:BERT:ALL]:CALCulate:DATA:EALarm?

**Function**

This command queries the measurement results of the error detector (ED).

**Syntax**

```
[:BERT:ALL]:CALCulate:DATA:EALarm? "<period>:<item>"
```

**Parameter**

<period> = CURRent | LAST

| | |
|---|---|
| CURRent | To query the current measurement results. |
| LAST | To query the results of the last measurement performed as specified by Gating Time. |

<item>

Select a measurement item(s) from the following:

| | |
|---|---|
| AINTerval:CRUNlock | CR Unlock Seconds |
| AINTerval:PSLoss | SYNC Loss Seconds |
| EC:TOTal | Bit Error Count Total |
| EC:INSertion | Bit Error Count Insertion |
| EC:OMISsion | Bit Error Count Omission |
| ER:TOTal | Bit Error Rate Total |
| ER:INSertion | Bit Error Rate Insertion |
| ER:OMISsion | Bit Error Rate Omission |
| CC:TOTal | Clock Count Total |
| FREQuency | FREQ(kHz) |

**Response Data**

The response format varies, depending on the item specified for <item>, as follows.

AINTerval:{CRUNlock | PSLoss}
EC:{TOTal | INSertion | OMISsion}
CC:TOTal
Range: "0" to "9999999" or "1.0000E+07" to "9.9999E+17" (Form1)
ER:{TOTal | INSertion | OMISsion}
Range: "0.0001E-18" to "1.0000E-00" (Form2)
FREQuency
"<integer>" (Form3)
"-------" is returned if there is no data to return.

**Example of Use**

To query the Bit Error Rate with the channel specified by :MODule:ID:

```
:CALC:DATA:EAL? "CURR:ER:TOT"
>"0.0000E-12"
```

*Added in Version 4:* (When :BERT:ALL is specified in the header)

To return the data of all channels in the order of Ch1, Ch2, Ch3, and Ch4:

```
:BERT:ALL:CALC:DATA:EAL? "CURR:ER:TOT"
>"0.0000E-12","0.0000E-12","0.0000E-12","0.0000E-12"
```

## :SENSe:MEASure:EALarm:MODE

### Function

This command sets and queries the gating cycle of ED.

### Syntax

```
:SENSe:MEASure:EALarm:MODE REPeat|SINGle|UNTimed
:SENSe:MEASure:EALarm:MODE?
```

### Parameter

| | |
|---|---|
| REPeat | Repeatedly performs measurement for the period of time set for Gating Time. |
| SINGle | Performs single measurement for the period of time set for Gating Time. |
| UNTimed | Finishes measurement using panel operation or continues measurement until the :SENSe:MEASure:STOP is sent. |

### Response Data

REP|SING|UNT

### Example of Use

```
:SENS:MEAS:EAL:MODE REP
:SENS:MEAS:EAL:MODE?
>REP
```

## :SENSe:MEASure:EALarm:PERiod

### Function

This command sets and queries the gating time of ED.

### Syntax

```
:SENSe:MEASure:EALarm:PERiod
<days>,<hours>,<minutes>,<seconds>
:SENSe:MEASure:EALarm:PERiod?
```

### Parameter

<days>,<hours>,<minutes>,<seconds>

### Response Data

<days>,<hours>,<minutes>,<seconds>

#### Example of Use

To set the gating time to 1 minute:

```
:SENS:MEAS:EAL:PER 0,0,1,0
:SENS:MEAS:EAL:PER?
>0,0,1,0
```

### :DISPlay:RESult:EALarm:MODE

#### Function

This command sets and queries the On/Off state of real-time update of ED measurement results.

#### Syntax

```
:DISPlay:RESult:EALarm:MODE <enable>
:DISPlay:RESult:EALarm:MODE?
```

#### Parameter

0|OFF
1|ON

#### Response Data

0|1

#### Example of Use

```
:DISP:RES:EAL:MODE ON
:DISP:RES:EAL:MODE?
>1
```

***Note:***

If set to Off, measurement results are updated when the measurement progress reaches 100%.

### [:BERT:ALL]:SENSe:MEASure:STARt

#### Function

This command starts the ED measurement.

#### Syntax

```
[:BERT:ALL]:SENSe:MEASure:STARt
```

#### Example of Use

To start measurement on the channel specified by :MODule:ID :

```
:SENS:MEAS:STAR
```

*Added in Version 4:* To start measurement on all ED channels:

`:BERT:ALL:SENS:MEAS:STAR`

***Notes:***

- If measurement is running, execution of this command clears the current data, and restarts measurement.
- To start measurement of all modules including Scope, use :SENS:MEAS:ASTR.

## [:BERT:ALL]:SENSe:MEASure:STOP

**Function**

This command stops the ED measurement.

**Syntax**

`[:BERT:ALL]:SENSe:MEASure:STOP`

**Example of Use**

To stop measurement on the channel specified by :MODule:ID:

`:SENS:MEAS:STOP`

*Added in Version 4:* To stop measurement on all ED channels:

`:BERT:ALL:SENS:MEAS:STOP`

***Note:***

To stop measurement of all modules including Scope, use :SENS:MEAS:ASTP.

## [:BERT:ALL]:SENSe:MEASure:EALarm:STATe?

**Function**

This command queries measurement status of the ED.

**Syntax**

`[:BERT:ALL]:SENSe:MEASure:EALarm:STATe?`

**Response Data**

0|1

| | |
|---|---|
| 0 | None of the channels is being measured. |
| 1 | At least one of the channels is being measured. |

**Example of Use**

To query the measurement status of the channel specified by :MODule:ID:

`:SENS:MEAS:EAL:STAT?`

`>0`

*Added in Version 4:* To query the measurement status of all ED channels:
`:BERT:ALL:SENS:MEAS:EAL:STAT?`
`>0`

***Note:***

To query the measurement status of all modules including Scope, use :SENS:MEAS:AST?.



## :SENSe:MEASure:EALarm:STARt?

**Function**

This command queries the measurement start time of the ED.

**Syntax**

`:SENSe:MEASure:EALarm:STARt?`

**Response Data**

"<year>,<month>,<day>,<hour>,<minute>,<second>"
If time data is not available, "0,0,0,0,0,0" is returned.

**Example of Use**

To query the measurement start time of the ED:
```
:SENS:MEAS:EAL:STAR?
>"2009,10,05,16,25,40"
```

## :SENSe:MEASure:EALarm:STOP?

**Function**

This command queries the measurement end time of the ED.

**Syntax**

`:SENSe:MEASure:EALarm:STOP?`

**Response Data**

"<year>,<month>,<day>,<hour>,<minute>,<second>"
If time data is not available, "0,0,0,0,0,0" is returned.

**Example of Use**

When Gating Cycle is Single or Repeat
```
:SENS:MEAS:EAL:STOP?
>"2009,10,05,16,25,40"
```
When Gating Cycle is Untimed
```
:SENS:MEAS:EAL:STOP?
>"0,0,0,0,0,0"
```

## :SENSe:MEASure:EALarm:ELAPsed?

### Function

This command queries the period of time elapsed from the start time of the ED measurement.

### Syntax

```
:SENSe:MEASure:EALarm:ELAPsed?
```

### Response Data

"<days>,<hours>,<minutes>,<seconds>"

### Example of Use

```
:SENS:MEAS:EAL:ELAP?
>"0,0,2,10"
```

## :SENSe:MEASure:EALarm:TIMed?

### Function

This command queries the measurement remaining time until the ED completes the measurement.

### Syntax

```
:SENSe:MEASure:EALarm:TIMed?
```

### Response Data

"<days>,<hours>,<minutes>,<seconds>"

### Example of Use

```
:SENS:MEAS:EAL:TIM?
>"0,0,2,10"
```

# 3.7 XFP/SFP+ specific messages

## :CALCulate:OPTical:STATus?

**Function**

This command queries the status of the XFP/SFP+ slot optical transceiver.

**Syntax**

`:CALCulate:OPTical:STATus? "READy"|"LOS"`

**Parameter**

| | |
|---|---|
| "READy" | Detection of XFP/SFP+ |
| "LOS" | Occurrence of LOS |

**Response Data**

If "READy" is specified:

| | |
|---|---|
| "None" | XFP/SFP+ not detected |
| "Occur" | XFP/SFP+ detected |

If "LOS" is specified:

| | |
|---|---|
| "None" | No LOS occurs or XFP/SFP+ not detected |
| "Occur" | LOS occurs |

**Example of Use**

```
:CALC:OPT:STAT? "LOS"
>"Occur"
```

## :SOURce:OPTical:SIGNal:WLENgth?

**Function**

This command queries the wavelength of the optical transceiver (XFP/SFP+).

**Syntax**

`:SOURce:OPTical:SIGNal:WLENgth?`

**Response Data**

<string>

| | |
|---|---|
| "xxxx" | Four-digit wavelength (Unit: nm)<br>(Right-justified if it is less than four digits.) |
| "-------" | XFP/SFP+ not detected |

**Example of Use**

```
:SOUR:OPT:SIGN:WLEN?
>" 850"
```

3

Message List

## :SOURce:OPTical:SIGNal:OUTPut

### Function

This command sets and queries the optical output of the optical transceiver (XFP/SFP+).

### Syntax

```
:SOURce:OPTical:SIGNal:OUTPut <enable>
:SOURce:OPTical:SIGNal:OUTPut?
```

### Parameter

0|OFF
1|ON

### Response Data

0|1

| | |
|---|---|
| 0 | Optical output Off or XFP/SFP+ not detected |
| 1 | Optical output On |

### Example of Use

To set the optical output to On:

```
:SOUR:OPT:SIGN:OUTP ON
:SOUR:OPT:SIGN:OUTP?
>1
```

## :SOURce:OPTical:XFP:REFClock

### Function

This command sets and queries the reference clock of the optical transceiver (XFP).

### Syntax

```
:SOURce:OPTical:XFP:REFClock <character>
:SOURce:OPTical:XFP:REFClock?
```

### Parameter

<character>

| | |
|---|---|
| ED1Sync | Sync with ED1 |
| ED2Sync | Sync with ED2 |
| PPG1Sync | Sync with PPG1 |
| PPG2Sync | Sync with PPG2 |

### Response Data

ED1Sync|ED2Sync|PPG1Sync|PPG2Sync

**Example of Use**

```
:SOUR:OPT:XFP:REF PPG1Sync
:SOUR:OPT:XFP:REF?
>PPG1Sync
```

# 3.8 O/E messages

## 3.8.1 CALibrate subsystem

### :CALibrate:AUTocorrect

**Function**

This command automatically adjusts the values of Conversion Gain, Responsivity, and System Conversion Gain so that they become equal to the values measured by the optical power meter.

**Syntax**

```
:CALibrate:AUTocorrect
```

**Example of Use**

```
:CAL:AUT
```

***Note:***

Before using Auto Correct, measure the unmodulated optical signal power by the optical power meter and set the measured value.

### :CALibrate:CALPower

**Function**

This command sets the optical signal power input to the device, and queries the set value.

**Syntax**

```
:CALibrate:CALPower <numeric>
:CALibrate:CALPower?
```

**Parameter**

<numeric>

Input Power

Range –2 to 10 dBm

**Example of Use**

```
:CAL:CALP -2
:CAL:CALP?
>-2
```

***Note:***

Measure the unmodulated optical signal power by the external optical power meter in advance, and set the measured value.

## :CALibrate:SYSTem:CGain

### Function

This command sets and queries the system conversion gain (V/W) for the O/E converter.

### Syntax

```
:CALibrate:SYSTem:CGain <numeric>
:CALibrate:SYSTem:CGain?
```

### Parameter

<numeric>
System Conversion Gain
Range 1 to 9999 V/W

### Response Data

<integer>

### Example of Use

```
:CAL:SYST:CG?
>160
```

## :CALibrate:CGain

### Function

This command sets and queries the O/E converter gain (V/W).

### Syntax

```
:CALibrate:CGain <numeric>
:CALibrate:CGain?
```

### Parameter

<numeric>
Conversion Gain
Range 1 to 9999 V/W

### Response Data

<integer>
1 to 9999

### Example of Use

```
:CAL:CG?
>320
```

## :CALibrate:RESPonsivity

**Function**

This command sets and queries the responsivity (A/W) of photo diode in O/E converter. This value is used to calculate the average optical power in amplitude measurement.

**Syntax**

```
:CALibrate:RESPonsivity <numeric>
:CALibrate:RESPonsivity?
```

**Parameter**

<numeric>

Responsivity

Range 0.001 to 9999

**Response Data**

<numeric>

**Example of Use**

```
:CAL:RESP?
>0.853
```

## :CALibrate:OEPower:JUDGe

**Function**

This command judges if the O/E converter can be calibrated, and queries the result of judgment.

**Syntax**

```
:CALibrate:OEPower:JUDGe
:CALibrate:OEPower:JUDGe?
```

**Response Data**

Pass | Fail

| | |
|---|---|
| Pass | The O/E converter can be calibrated. (Optical input level ≤ –30 dBm) |
| Fail | The O/E converter cannot be calibrated. (Optical input level > –30 dBm) |

**Example of Use**

```
:CAL:OEP:JUDG
:CAL:OEP:JUDG?
>Pass
```

### :CALibrate:OEPower

**Function**

This command starts the calibration of the O/E converter.

**Syntax**

```
:CALibrate:OEPower
```

**Example of Use**

```
:CAL:OEP
```



## 3.8.2 SENSe subsystem

### [:SENSe]:INPut:FILTer

**Function**

This command sets and queries the internal low-pass filter.

**Syntax**

```
[:SENSe]:INPut:FILTer <integer>
[:SENSe]:INPut:FILTer?
```

**Parameter**

| <integer> | Standards | Cut-off frequency |
|---|---|---|
| 0 | None | |
| 1 | 2GFC | 2.125G |
| 2 | 4GFC | 4.25G |
| 3 | fc = 6.3GHz | 6.3G |
| 4 | 10GFC | 10.51875G |
| 5 | 10GbE WAN | 9.95328G |
| 6 | 10GbE LAN/PHY | 10.3125G |
| 7 | OC192/STM-64 | 9.95328G |
| 8 | G.975 FEC | 10.664228G |
| 9 | OTU-2 | 10.709225G |
| 10 | 1GFC | 1.0625G |
| 11 | 10GFC FEC | 11.3168G |
| 12 | 1GbE | 1.25G |
| 13 | 2GbE | 2.5G |
| 14 | 10GbE FEC | 11.095728G |
| 15 | Infiniband Optical | 2.5G |
| 16 | Infiniband Optical x2 | 5G |
| 17 | Infiniband Optical x4 | 10G |
| 18 | OC3/STM-1 | 155.52M |
| 19 | OC12/STM-4 | 622.08M |

| <integer> | Standards | Cut-off frequency |
|---|---|---|
| 20 | OC24 | 1.244G |
| 21 | OC48/STM-16 | 2.488G |
| 22 | OTU-1 | 2.66648G |
| 23 | CPRI | 614.4M |
| 24 | CPRI x2 | 1.2288G |
| 25 | CPRI x4 | 2.4576G |
| 26 | CPRI x5 | 3.072G |
| 27 | CPRI x8 | 4.9515G |
| 28 | CPRI x10 | 6.144G |
| 29 | 10GBASE-LX4 | 3.125G |
| 30 | 10GFC-LX4 | 3.1875G |
| 31 | XAUI Optical x2 | 6.25G |
| 32 | 8GFC | 8.5G |

**Response Data**

`<integer>`

**Example of Use**

To set the filter to OC192/STM64:

```
:INP:FILT 7
:INP:FILT?
>7
```

## [:SENSe]:INPut:FILTer:ENABle

**Function**

This command sets and queries the On/Off state of the filter of the filter bank option.

**Syntax**

```
[:SENSe]:INPut:FILTer:ENABle 0|1
[:SENSe]:INPut:FILTer:ENABle?
```

**Parameter**

0 Sets the Filter to Off.

1 Sets the Filter to On.

**Response Data**

0|1

**Example of Use**

To set the filter to Off.

```
:INPut:FILTer:ENABle 0
```

```
:INPut:FILTer:ENABle?
>0
```

***Note:***

This command is available only when the Option 086 is installed.

Execute amplitude calibration before using this command.

### [:SENSe]:INPut:WAVLength

**Function**

This command sets and queries the wavelength of the O/E converter.

**Syntax**

```
[:SENSe]:INPut:WAVLength 850|1310|1550|USER
[:SENSe]:INPut:WAVLength?
```

**Parameter**

| | |
|---|---|
| 850 | Sets wavelength to 850 nm |
| 1310 | Sets wavelength to 1310 nm |
| 1550 | Sets wavelength to 1550 nm |
| USER | Sets wavelength to User |

**Response Data**

850|1310|1550|USER

**Example of Use**

To set the O/E converter wavelength to 1550 nm

```
:INP:WAVL 1550
:INP:WAVL?
>1550
```

## 3.8.3 CONFigure subsystem

### :CONFigure:EXRCorrection

**Function**

This command sets and queries the On/Off state of the extinction ratio correction of the O/E converter.

**Syntax**

```
:CONFigure:EXRCorrection 0|1
:CONFigure:EXRCorrection?
```

**Parameter**

0 Off

1 On

**Response Data**

0|1

**Example of Use**

```
:CONF:EXRC 1
:CONF:EXRC?
>1
```

## :CONFigure:EXRCorrection:FACTor

**Function**

This command sets and queries the extinction ratio correction factor for the O/E converter.

**Syntax**

```
:CONFigure:EXRCorrection:FACTor <numeric>
:CONFigure:EXRCorrection:FACTor?
```

**Parameter**

<numeric>

Extinction Ratio Correction Factor

Range –9.99 to 9.99%, 0.01 step

**Response Data**

<numeric>

**Example of Use**

To set extinction ratio correction factor to 1.2%

```
:CONF:EXRC:FACT 1.20
:CONF:EXRC:FACT?
>1.20
```

# 3.9 Scope-specific messages

## 3.9.1 CALibrate subsystem

### :CALibrate:AMPLitude

**Function**

This command initiates an amplitude calibration for Scope Channel A and B.

**Syntax**

```
:CALibrate:APPLitude[?]
```

For Version 3.2 or later, this command can be used as a query by appending “?”.

**Response Data**

One of the following results is returned when the calibration ends.

| | |
|---|---|
| "Calibration complete." | Calibration has successfully completed. |
| "Calibration Failed.(CHA)" | Calibration of channel A has failed. |
| "Calibration Failed.(CHB)" | Calibration of channel B has failed. |
| "Calibration Failed.(CHA&CHB)" | Calibration of channels A and B has failed. |

**Example of Use**

```
:CAL:AMPL
```

(Time-waiting process of approx. 60 s)

```
>"Caribration Failed.(CHB)"
```

***Note:***

In EYE mode, calibration takes about 50 s. Therefore, the timeout for the interface must be set to 60 s. If Timeout is set to less than 60 s, a calibration result may not be read due to a timeout error occurring before outputting the result to an output queue.

### :CALibrate:APPLication

**Function**

This command starts the self-test of the Scope.
Also, this command queries the self-test result of the Scope.

**Syntax**

```
:CALibrate:APPLication
:CALibrate:APPLication?
```

**Response Data**

"Self Test Passed!"

"Self Test Passed!" The self-test ended normally.

When the self-test error occurs, the response message is not returned.

**Example of Use**

```
:CAL:APPL?
>"Self Test Passed!"
```

### :CALibrate:TEMPerature?

**Function**

This command queries the current temperature and temperature during the calibration on the Scope module.

**Syntax**

```
:CALibrate:TEMPerature?
```

**Response Data**

<numeric>,<numeric>

Current temperature (°C), Temperature during calibration (°C)

**Example of Use**

```
:CAL:TEMP?
>39.6,24.4
```

## 3.9.2 DISPlay subsystem

### :DISPlay:LABel

**Function**

This command sets a label to display in the Scope screen.

**Syntax**

```
:DISPlay:LABel <string>
```

**Parameter**

<string>

Label

Up to 1024 alphanumeric characters can be specified.

**Example of Use**

To display 8 characters “BERTWave” when [Sampling Run] is displayed. This example waits until the label is displayed, and then saves the screenshot to a file.

```
:DISPlay:LABel "BERTWave"
```
(Fixed-time waiting (for example, 200 ms))
```
:EYEPulse:PRINt:COPY
```

***Notes:***

- Depending on the button displayed in the Scope screen, the command operation will vary as follows:
  If [Sampling Hold] is displayed, a label is displayed immediately after sending the command.
  If [Sampling Run] is displayed, a label is not displayed immediately after sending the command.

  In this case, set the waiting time on the Control PC according to “Example of Use” given below, and then send the next command. The waiting time varies depending on the length of a string to display.

- Even if the *WAI command is used, this command will not wait.



## :DISPlay:LABel:DALL

**Function**

This command deletes the label displayed in the Scope screen.

**Syntax**

```
:DISPlay:LABel:DALL
```

**Example of Use**

```
:DISP:LAB:DALL
```

## :DISPlay:WINDow:CHANnel:BOTH

**Function**

This command sets and queries the On/Off state of Channel A/B Tracking of Scope.

**Syntax**

```
:DISPlay:WINDow:CHANnel:BOTH <enable>
:DISPlay:WINDow:CHANnel:BOTH?
```

**Parameter**

0|OFF
1|ON

**Response Data**

0|1

**Example of Use**

```
:DISP:WIND:CHAN:BOTH ON
:DISP:WIND:CHAN:BOTH?
>1
```

## :DISPlay:WINDow:GRAPhics:CLEar

**Function**

This command erases the trace on the Scope screen.

**Syntax**

```
:DISPlay:WINDow:GRAPhics:CLEar
```

**Example of Use**

```
:DISP:WIND:GRAP:CLE
```

## :DISPlay:WINDow[:SCALe]:AUTOscale

**Function**

This command automatically adjusts the vertical and horizontal axes so that the waveform is displayed in the center of the EYE/Pulse Scope screen.

**Syntax**

```
:DISPlay:WINDow[:SCALe]:AUTOscale
[BOTH|HORIzontal|VERTical]
```

**Parameter**

In EYE mode, the parameter can be specified.

| | |
|---|---|
| BOTH | Scale on the vertical axis and offset on the horizontal axis auto-adjusted |
| HORizontal | Offset on the horizontal axis auto-adjusted |
| VERTical | Scale on the vertical axis auto-adjusted |

**Example of Use**

```
:DISP:WIND:AUTO
```

***Note:***

If the parameter is specified when in EYE mode, execution of this command only adjusts the offset on the horizontal axis and the scale on the vertical axis, without measuring the waveform frequency. Additionally, when this parameter is specified, Acquire

Clock (measuring frequency of the clock signal input to the **Trigger Clk In** connector) is not executed during Auto Scale, which provides an advantage of shorter duration of automatic adjustment by Auto Scale. If the clock frequency is already known, specify this parameter as needed.

## :DISPlay:WINDow:X[:SCALe]:BITs

### Function

This command sets and queries the horizontal scale of Scope, by the number of bits.

### Syntax

```
:DISPlay:WINDow:X[:SCALe]:BITs <integer>
:DISPlay:WINDow:X[:SCALe]:BITs?
```

### Parameter

<integer>

Range 1 to 1000, 1 bit step
(If Fast Sampling Mode is On, the range is 1 to 100.)

### Response Data

<integer>

### Example of Use

```
:DISP:WIND:X:BIT 2
:DISP:WIND:X:BIT?
>2
```

## :DISPlay:WINDow:X[:SCALe]:OFFSets

### Function

This command sets and queries the offset value of the horizontal scale of the Scope.

### Syntax

```
:DISPlay:WINDow:X[:SCALe]:OFFSets <numeric>
:DISPlay:WINDow:X[:SCALe]:OFFSets?
```

### Parameter

<numeric>
Offset value
Unit UI or picosecond (ps)
The unit can be set using the :DISPlay:WINDow:X[:SCALe]:UNIT.
When unit is UI: 0 to 16777215

When unit is ps: Offset (UI) / Date-Rate (Tbps or 1000 Gbps)

**Response Data**

<numeric>

**Example of Use**

```
:DISP:WIND:X:OFFS 150
```

## :DISPlay:WINDow:X[:SCALe]:UNIT

**Function**

This command sets and queries the unit of the horizontal scale of Scope.

**Syntax**

```
:DISPlay:WINDow:X[:SCALe]:UNIT PS|UI
:DISPlay:WINDow:X[:SCALe]:UNIT?
```

**Parameter**

PS: picosecond ($10^{-12}$ seconds)

UI: Unit Interval.

**Response Data**

PS|UI

**Example of Use**

```
:DISP:WIND:X:UNIT UI
:DISP:WIND:X:UNIT?
>UI
```

## :DISPlay:WINDow:Y[:SCALe]:DIVision:CHA|CHB

**Function**

This command sets and queries the unit of the vertical scale of Scope.

**Syntax**

```
:DISPlay:WINDow:Y[:SCALe]:DIVision:CHA|CHB <numeric>
:DISPlay:WINDow:Y[:SCALe]:DIVision:CHA|CHB?
```

**Parameter**

CHA|CHB

| | |
|---|---|
| CHA | Channel A |
| CHB | Channel B |

<numeric>

| | |
|---|---|
| Electrical: | 1.0 mV to 250.0 × [10 (attenuation/20)] mV |

Optical: Divided by Gain (V/W) (Conversion gain)
The range is the same as electrical.

**Response Data**

<numeric>

**Example of Use**

```
:DISP:WIND:Y:DIV:CHB?
>100
```

***Notes:***

- The vertical scale can set either of electrical interface or optical interface using this message.
- The units, mV and μW, are used for the electrical and optical interface, respectively.
- The maximum value for the scale depends on the attenuation set for the specified channel.
- The attenuation can be set using [:SENSe]:INPut:ATTenuation:CHA | CHB.
- For the optical interface, the settable maximum values vary with the O/E conversion gain.
- The O/E conversion gain can be set using :CALibrate:Cgain.

## :DISPlay:WINDow:Y[:SCALe]:DIVision:CHMath

**Function**

This command sets and queries the value (mV) of the vertical scale of Scope when Channel Math is On.

**Syntax**

```
:DISPlay:WINDow:Y[:SCALe]:DIVision:CHMath <numeric>
:DISPlay:WINDow:Y[:SCALe]:DIVision:CHMath?
```

**Parameter**

<numeric>

Range 1 to 200 mV, 0.1 mV step

**Response Data**

<numeric>

**Example of Use**

```
:DISP:WIND:Y:DIV:CHM?
>100
```

***Notes:***

- The maximum value of the settable scale varies with the attenuation set at each channel.
- The attenuation can be set using [:SENSe]:INPut:ATTenuation:CHA|CHB.

## :DISPlay:WINDow:Y[:SCALe]:OFFSets:CHA|CHB

**Function**

This command sets and queries the offset value for the vertical scale of Scope.

**Syntax**

```
:DISPlay:WINDow:Y[:SCALe]:OFFSets:CHA|CHB <numeric>
:DISPlay:WINDow:Y[:SCALe]:OFFSets:CHA|CHB?
```

**Parameter**

CHA|CHB

| | |
|---|---|
| CHA | Channel A |
| CHB | Channel B |

<numeric>

| | |
|---|---|
| Electrical interface: | –500.0 mV to +500.0 mV |
| Optical interface: | Divided value (mW) (divides –500.0 mV to +500.0 mV by Gain (V/W) (conversion gain)) |

**Response Data**

<numeric>

**Example of Use**

```
:DISP:WIND:Y:OFFS:CHB?
>-50
```

***Notes:***

- The vertical scale can set either of electrical or optical interfaces using this message.
- The units, mV and μW, are used for the electrical and optical interfaces, respectively.
- The maximum value for the offset depends on the attenuation

set for the specified channel.

- The attenuation can be set using [:SENSe]:INPut:ATTenuation:CHA | CHB.
- For the optical interface, the settable maximum values vary with the O/E conversion gain.
- The O/E conversion gain can be set using: CALibrate:Cgain.





## :DISPlay:WINDow:Y[:SCALe]:OFFSets:CHMath

### Function

This command sets and queries the offset value (mV) of the vertical scale of Scope when Channel Math is On.

### Syntax

```
:DISPlay:WINDow:Y[:SCALe]:OFFSets:CHMath <numeric>
:DISPlay:WINDow:Y[:SCALe]:OFFSets:CHMath?
```

### Parameter

<numeric>

Range    –1000.0 to +1000.0 mV, 0.1 mV step

### Response Data

<numeric>

### Example of Use

```
:DISP:WIND:Y:OFFS:CHM?
>-50
```

***Note:***

The maximum value for the offset depends on the attenuation set for the specified channel.

The attenuation can be set using [:SENSe]:INPut:ATTenuation:CHA | CHB.

## 3.9.3 SENSe subsystem

### [:SENSe]:ACCUmulation:AVERaging

**Function**

This command sets and queries the averaging process count of Scope.

**Syntax**

```
[:SENSe]:ACCUmulation:AVERaging <integer>
[:SENSe]:ACCUmulation:AVERaging?
```

**Parameter**

<integer>

Range 1 to 9999

**Response Data**

<integer>

1 to 9999

**Example of Use**

```
:ACCU:AVER?
>1000
```

### [:SENSe]:ACCUmulation:LIMit

**Function**

This command sets and queries the limit (time/number) for the data collection, when Accumulation Type of Scope is Limited.

**Syntax**

```
[:SENSe]:ACCUmulation:LIMit
TIME|SAMPle|WAVeform,<numeric>
[:SENSe]:ACCUmulation:LIMit?
```

**Parameter**

TIME | SAMPLe | WAVeform

TIME The data collection is limited at time.

SAMPle The data collection is limited by the number of samples.

WAVeform The data collection is limited by the number of waveforms.

<numeric>

When specifying TIME, set the limit by the time in seconds.

When specifying SAMPle, set the limit by the number of samples in million units, in the range of 1 to 99999.

When specifying WAVeform, set the limit by the number of waveforms, in the range of 1 to 999999.

**Response Data**

TIME | SAMPLe | WAVeform,<integer>

**Example of Use**

```
:ACCU:LIM SAMPLe,10
:ACCU:LIM?
>SAMPLe,10
```

***Note:***

The unit (second or million) does not need to be specified by the command. The unit is determined automatically depending on the specified limit type, as follows:
- sec (when the limit type is Time)
- million (when the limit type is Sample)

When this message is transmitted while correcting the data (when Sampling of the screen is [RUN]), the displayed waveform is deleted and the data correction is done over again.



## [:SENSe]:ACCUmulation:PERSistency

**Function**

This command sets and queries the data display time when the data collection process of Scope is Persistency.

**Syntax**

```
[:SENSe]:ACCUmulation:PERSistency <numeric>
[:SENSe]:ACCUmulation:PERSistency?
```

**Parameter**

<numeric>
Time to display collected data (seconds)

**Response Data**

<integer>

**Example of Use**

```
:ACCU:PERS?
>10.0
```

## [:SENSe]:ACCUmulation:TYPe

**Function**

This command sets and queries the data collection process of Scope.

**Syntax**

```
[:SENSe]:ACCUmulation:TYPe <character>
[:SENSe]:ACCUmulation:TYPe?
```

**Parameter**

<character>

| | |
|---|---|
| NONe | The data collection is not overwritten. When the fresh data is collected, the displayed data is deleted. |
| INFinite | The data collection is overwritten.<br>The acquired data does not go out of the screen. |
| LIMited | The data collection is limited by the number of samples and time.<br>When it reaches the limited conditions, the data collection is ended.<br>The acquired data does not go out of the screen. |
| PERSistency | The data collection is overwritten.<br>After the fixed time, the acquired data goes out of the screen. |
| AVERaging | The average of the collected data is displayed. Only when Sampling Mode is set to Pulse, this can be used. |

**Response Data**

NONe | INFinite | LIMited | PERSistency | AVERaging

**Example of Use**

```
:ACCU:TYP LIMited
:ACCU:TYP?
>LIMited
```

## [:SENSe]:DISPlay:MODE

**Function**

This command sets and queries the Sampling Mode of Scope.

**Syntax**

```
[:SENSe]:DISPlay:MODE COHErenteye|EYE|PULSe
[:SENSe]:DISPlay:MODE?
```

**Parameter**

| | |
|---|---|
| COHErenteye | Coherent eye mode |
| EYE | Eye mode |
| PULSe | Pulse mode |

**Response Data**

COHErenteye | EYE | PULSe

**Example of Use**

```
:DISP:MODE PULSe
:DISP:MODE?
>PULSe
```

## [:SENSe]:DISPlay:MODE:EYE:FAST

**Function**

*Added in Version 4:* This command sets and queries the On/Off state of the Fast Sampling Mode of Scope.

**Syntax**

```
[:SENSe]:DISPlay:MODE:EYE:FAST <enable>
[:SENSe]:DISPlay:MODE:EYE:FAST?
```

**Parameter**

0|OFF

1|ON

**Response Data**

0|1

**Example of Use**

```
:DISP:MODE EYE
:DISP:MODE:EYE:FAST ON
:DISP:MODE:EYE:FAST?
>1
```

## [:SENSe]:EYEPulse:PRINt:COPY

**Function**

This command takes a screenshot of the Scope screen.

**Syntax**

```
[:SENSe]:EYEPulse:PRINt:COPY
[<file_name>,<directory>][,PNG|JPEG]
```

**Parameter**

Refer to :SYSTem:PRINt:COPY.

**Example of Use**

```
:MOD:ID 5
:EYEP:PRIN:COPY "screen_copy_eye","C:/screen_copy"
```

***Note:***

To take a screenshot of the entire screen, use :SYST:PRIN:COPY.

To query the screenshot image file data, use :SYST:DISP:DATA?.

## [:SENSe]:HISTogram:CENTer

**Function**

This command moves the marker position of the histogram measurement of Scope to the center of the screen.

**Syntax**

```
:HISTogram:CENTer
```

**Example of Use**

```
:HIST:CENT
```

## [:SENSe]:HISTogram:X1|X2

**Function**

This command sets and queries the position of the marker X1 or X2 for setting the histogram measurement area of Scope.

**Syntax**

```
[:SENSe]:HISTogram:X1|X2 <numeric>
[:SENSe]:HISTogram:X1|X2?
```

**Parameter**

X1|X2 Marker

<numeric>

Marker position (Time)

| | | |
|---|---|---|
| Unit | UI or ps | |
| Range | ps: | 0 to 4294967295 |
| | UI: | 0 to 4294967 |

**Response Data**

<numeric>

**Example of Use**

```
:HIST:X1?
>10050
```

## [:SENSe]:HISTogram:Y1|Y2

**Function**

This command sets and queries the position of the marker Y1 or Y2 for setting the histogram measurement area of Scope.

**Syntax**

```
[:SENSe]:HISTogram:Y1|Y2 <numeric>
[:SENSe]:HISTogram:Y1|Y2?
```

**Parameter**

Y1|Y2 Marker

<numeric>

Marker position (Amplitude)

Unit mV (Electrical input)

μW (Optical input)

Range mV: Between minimum and maximum values for display area

μW: Between minimum and maximum values for display area

Minimum value: Offset–Scale*5,

Maximum value: Offset+Scale*5

**Response Data**

<numeric>

**Example of Use**

```
:HIST:Y2?
>-60.6
```

## [:SENSe]:INPut:ATTenuation:CHA|CHB

**Function**

This command sets and queries the amount of attenuation for adjusting the Scope amplitude scal.

**Syntax**

```
[:SENSe]:INPut:ATTenuation:CHA|CHB <numeric>
[:SENSe]:INPut:ATTenuation:CHA|CHB?
```

**Parameter**

CHA|CHB

CHA Channel A

CHB Channel B

<numeric>

Attenuation

Range 0.00 to 30.00, 0.01 dB step

**Response Data**

<numeric>

**Example of Use**

```
:INP:ATT:CHA 20.00
```

## [:SENSe]:INPut:CHA|CHB

**Function**

This command sets and queries the On/Off state of the Ch A/B waveform display of Scope.

**Syntax**

```
[:SENSe]:INPut:CHA|CHB ON|OFF
[:SENSe]:INPut:CHA|CHB?
```

**Parameter**

CHA|CHB

| | |
|---|---|
| CHA | Channel A |
| CHB | Channel B |

ON|OFF

**Response Data**

ON|OFF

**Example of Use**

```
:INP:CHA ON
:INP:CHA?
>ON
```

## [:SENSe]:INPut:CLKRecovery

**Function**

This command sets and queries the clock recovery output mode of Scope.

**Syntax**

```
[:SENSe]:INPut:CLKRecovery OFF|LESS27|85
[:SENSe]:INPut:CLKRecovery?
```

**Parameter**

| | |
|---|---|
| OFF | Sets the clock recovery output to Off |
| LESS27 | Sets the clock recovery output to On and frequency to 0.1 to 2.7 GHz |

85 Sets the clock recovery output to On and frequency to 8.5 to 12.5 GHz

**Response Data**

OFF|LESS27|85

**Example of Use**

```
:INP:CLKR 85
```

## [:SENSe]:OPTion:MAX:SAMPles:NUMber

**Function**

This command sets and queries the number of data acquired in the data collection at one time per channel of Scope.

**Syntax**

```
[:SENSe]:OPTion:MAX:SAMPles:NUMber <integer>
[:SENSe]:OPTion:MAX:SAMPles:NUMber?
```

**Parameter**

<integer>

When Sampling mode is set to [EYE]:

509|1021|1350|2039|4093|8191|16381

When Sampling mode is set to [Coherent Eye] or [Pulse]:

512|1024|2048|4096|8192|16384

**Response Data**

<integer>

**Example of Use**

```
:SENS:OPT:MAX:SAMP:NUM?
>8191
```

## [:SENSe]:PRINt:GRATicule

**Function**

This command sets and queries the On/Off state for taking a screenshot of only the waveform area of Scope.

**Syntax**

```
[:SENSe]:PRINt:GRATicule <enable>
[:SENSe]:PRINt:GRATicule?
```

**Parameter**

0|OFF

1|ON

**Response Data**

0|1

**Example of Use**

```
:MOD:ID 5
:PRIN:GRAT ON
:PRIN:GRAT?
>1
```

## [:SENSe]:SAMPles:JUDGe:TYPE

**Function**

*Added in Version 3.03/4.01:* This command sets and queries the Margin Type for Mask Margin measurement.

**Syntax**

```
[:SENSe]:SAMPles:JUDGe:TYPE {COUNt|RATE}
[:SENSe]:SAMPles:JUDGe:TYPE?
```

**Parameter**

`COUNt` Hit Count
`RATE` Hit Ratio

**Response Data**

`COUNt|RATE`

**Example of Use**

To set the threshold for Mask Margin measurement to Hit Count 10:

```
:SAMP:JUDG:TYPE COUNt
:SAMP:JUDG 10
:SAMP:JUDG?
>10
```

To set the threshold for Mask Margin measurement to Hit Ratio 1E–5:

```
:SAMP:JUDG:TYPE RATE
:SAMP:JUDG:RATE E_5,1
:SAMP:JUDG:RATE?
>E_5,1
```

## [:SENSe]:SAMPles:JUDGe

**Function**

This command sets and queries the Hit Count for Mask Margin measurement.

**Syntax**

```
[:SENSe]:SAMPles:JUDGe <numeric>
[:SENSe]:SAMPles:JUDGe?
```

**Parameter**

<numeric>
Sample point counts in the mask area

**Response Data**

<integer>

**Example of Use**

See “Example of Use” of `[:SENSe]:SAMPles:JUDGe:TYPE`



## [:SENSe]:SAMPles:JUDGe:RATE

**Function**

*Added in Version 3.03/4.01:* This command sets and queries the Hit Ratio for Mask Margin measurement.

**Syntax**

```
[:SENSe]:SAMPles:JUDGe:RATE E_<integer>[,<integer>]
[:SENSe]:SAMPles:JUDGe:RATE?
```

**Parameter**

<integer>,<integer>
Specify the arguments for the parameter in the order below.

| Description | Range |
|---|---|
| Exponent of Hit Ratio | 1 to 12 |
| Mantissa of Hit Ratio (1 if omitted) | 1 to 9 |

To set the Hit Ratio to $2 \times 10^{-8}$, specify the parameter as “E_8,2”.

**Response Data**

```
E_<integer>,<integer>
```

**Example of Use**

See “Example of Use” of `[:SENSe]:SAMPles:JUDGe:TYPE`.

## [:SENSe]:PRINt:INVerse

**Function**

This command sets and queries the On/Off state of background color inversion for a screenshot of Scope.

**Syntax**

```
[:SENSe]:PRINt:INVerse <enable>
[:SENSe]:PRINt:INVerse?
```

**Parameter**

0|OFF
1|ON

**Response Data**

0|1

**Example of Use**

```
:MOD:ID 5
:PRIN:INV OFF
:PRIN:INV?
>0
```

## [:SENSe]:SAMPling:STATus

**Function**

This command runs and holds the sampling processing of Scope, and queries the state of the sampling processing.

**Syntax**

```
[:SENSe]:SAMPling:STATus RUN|HOLD
[:SENSe]:SAMPling:STATus?
```

**Parameter**

| | |
|---|---|
| RUN | Runs the sampling processing. |
| HOLD | Holds the sampling processing. |

**Response Data**

RUN|HOLD

**Example of Use**

To run the sampling processing:

```
:SAMP:STAT RUN
```

To query if the Sampling processing has been held (HOLD status):

```
:SAMP:STAT?
>HOLD
```

## [:SENSe]:TIME:ACQClock?

### Function

This command measures the frequency of the clock input to the **Trigger Clk In** connector of Scope and sets the measured result as the Clock Rate parameter, and queries the measured frequency.

### Syntax

```
[:SENSe]:TIME:ACQClock?
```

### Response Data

<integer>
Unit Hz

### Example of Use

```
:TIME:ACQC?
```

## [:SENSe]:TIME:AUTodetect

### Function

This command sets and queries the On/Off state of auto detection of the clock divide ratio (1/n) input to the **Trigger Clk In** connector of Scope.

### Syntax

```
[:SENSe]:TIME:AUTodetect ON|OFF
[:SENSe]:TIME:AUTodetect?
```

### Parameter

ON|OFF

### Response Data

ON|OFF

### Example of Use

```
:TIME:AUT ON
:TIME:AUT?
>ON
```

## [:SENSe]:TIME:CLKRate

### Function

This command sets and queries the clock rate of Scope.
When changing the clock rate, the bit rate is changed to the value multiplexed clock rate by divide ratio.

**Syntax**

```
[:SENSe]:TIME:CLKRate <numeric> [GHZ|MHZ|KHZ]
[:SENSe]:TIME:CLKRate?
```

**Parameter**

<numeric>
Clock rate
The following unit can be used.
GHZ: GHz
KHZ: kHz
MHz: MHz
When omitting the unit, the unit is fixed to MHz.

**Response Data**

<numeric> MHz

**Example of Use**

To set the clock rate of the Scope to 10312.5 MHz

```
:TIME:CLKR 10312.5
:TIME:CLKR?
>10312.50 MHz
```

## [:SENSe]:TIME:DATRate

**Function**

This command sets and queries the bit rate of Scope.
When changing the bit rate, the clock rate is changed to the value multiplexed bit rate by divide Ratio.

**Syntax**

```
[:SENSe]:TIME:DATRate <numeric> [Gbps|kbps|Mbps]
[:SENSe]:TIME:DATRate?
```

**Parameter**

<numeric>
Bit rate
The following unit can be used.
Gbps: Gbit/s
kbps: kbit/s
Mbps: Mbit/s
When omitting the unit, the unit is fixed to Mbps.

**Response Data**

<numeric>

Bit rate
Unit Mbit/s

**Example of Use**

To set the bit rate of the Scope to 155220 kbit/s

```
:TIME:DATR 155220 kbps
:TIME:DATR?
>155.220 Mbps
```

## [:SENSe]:TIME:DIVRatio

**Function**

This command sets and queries the clock divide ratio of Scope.
When changing the clock divide ratio, either of the bit rate or clock frequency is changed.

**Syntax**

```
[:SENSe]:TIME:DIVRatio <integer>,{CLKR|DATA}
[:SENSe]:TIME:DIVRatio?
```

**Parameter**

<integer>

| | |
|---|---|
| Divide ratio | 1 to 64 |
| DATR | The bit rate is re-calculated from the divide ratio and clock frequency. |
| CLKR | The clock frequency is re-calculated from the divide ratio and bit rate. |

**Response Data**

<integer>

Divide ratio 1 to 64

**Example of Use**

To set the 1/16 value of the clock frequency to the bit rate:

```
:TIME:DIVR 16,CLKR
```

## [:SENSe]:TIME:PATLength

**Function**

This command sets and queries the data pattern length used in the pulse pattern mode of the Scope.

**Syntax**

```
[:SENSe]:TIME:PATLength <numeric>
[:SENSe]:TIME:PATLength?
```

#### Parameter

<numeric>

Range 1 to 16777216

#### Response Data

<integer>

1 to 16777216

#### Example of Use

```
:TIME:PATL 8388607
```

### [:SENSe]:TMEMory:CHANnel

#### Function

This command sets and queries the channel saved as a reference trace of Scope.

When saving the reference trace, use [:SENSe]:TMEMory:REFerence:SET.

#### Syntax

```
[:SENSe]:TMEMory:CHANnel BOTH|CHA|CHB
[:SENSe]:TMEMory:CHANnel?
```

#### Parameter

| | |
|---|---|
| BOTH | Channel A and Channel B |
| CHA | Channel A |
| CHB | Channel B |

#### Response Data

Both|CHA|CHB

#### Example of Use

```
:TMEM:CHAN Both
:TMEM:CHAN?
>Both
```

### [:SENSe]:TMEMory:REFerence:CLEar

#### Function

This command deletes the reference trace of Scope.

#### Syntax

```
[:SENSe]:TMEMory:REFerence:CLEar
```

**Example of Use**

`:TMEM:REF:CLE`

## [:SENSe]:TMEMory:REFerence:SET

**Function**

This command saves the trace displayed on Scope as the reference trace.

**Syntax**

`[:SENSe]:TMEMory:REFerence:SET`

**Example of Use**

`:TMEM:REF:SET`

## 3.9.4 CONFigure subsystem

### :CONFigure:CLKRecovery

**Function**

This command sets and queries the bandwidth of the clock recovery unit for Scope.

**Syntax**

```
:CONFigure:CLKRecovery 1|2|4|8 [MHz]
:CONFigure:CLRecovery?
```

**Parameter**

1|2|4|8 [MHz]

**Response Data**

1|2|4|8 MHz

**Example of Use**

To set the bandwidth of clock recovery unit to 4 MHz

```
:CONF:CLKR 4 MHz
:CONF:CLKR?
>4 MHz
```

### :CONFigure:HISTogram:AXIS

**Function**

This command sets and queries the axis for histogram measurement of Scope.

**Syntax**

```
:CONFigure:HISTogram:AXIS TIME|AMPLitude
:CONFigure:HISTogram:AXIS?
```

**Parameter**

| | |
|---|---|
| TIME | Time direction histogram |
| AMPLitude | Amplitude direction histogram |

**Response Data**

TIME|AMPLitude

**Example of Use**

```
:CONF:HIST:AXIS AMPLitude
:CONF:HIST:AXIS?
>AMPLitude
```

## :CONFigure:MASK:ALGorithm

### Function

This command sets and queries the mask alignment method of Scope.

### Syntax

```
:CONFigure:MASK:ALGorithm 0|2
:CONFigure:MASK:ALGorithm?
```

### Parameter

0 Mask alignment by detecting the intersection of Zero Level and One level

2 Mask alignment by user operations

### Response Data

0|2

### Example of Use

```
:CONF:MASK:ALG 2
:CONF:MASK:ALG?
>2
```

## :CONFigure:MASK:AREa:RESTriction

### Function

This command sets and queries the mask area restriction of Scope.

### Syntax

```
:CONFigure:MASK:AREa:RESTriction <enable>
:CONFigure:MASK:AREa:RESTriction?
```

### Parameter

0|OFF

1|ON

### Response Data

0|1

### Example of Use

```
:CONF:MASK:ARE:REST ON
:CONF:MASK:ARE:REST?
>1
```

## :CONFigure:MASK:AREa:RESTriction:ANGLe

### Function

This command sets and queries the angle restricting the mask area of Scope.

### Syntax

```
:CONFigure:MASK:AREa:RESTriction:ANGLe <integer>
:CONFigure:MASK:AREa:RESTriction:ANGLe?
```

### Parameter

<integer>

Angle restriction mask area

Range –90 to 90 degrees

### Response Data

<integer>

### Example of Use

```
:CONF:MASK:ARE:REST:ANGL -30
:CONF:MASK:ARE:REST:ANGL?
>-30
```

## :CONFigure:MASK:AREa:RESTriction:WIDTh

### Function

This command sets and queries the width restricting the mask area of Scope.

### Syntax

```
:CONFigure:MASK:AREa:RESTriction:WIDTh <numeric>
:CONFigure:MASK:AREa:RESTriction:WIDTh?
```

### Parameter

<numeric>

Width restricting mask area

Range 0.01 to 1.00 UI

### Response Data

<numeric>

### Example of Use

```
:CONF:MASK:ARE:REST:WIDT 0.15
:CONF:MASK:ARE:REST:WIDT?
>0.15
```

## :CONFigure:MASK:MARGin

### Function

This command sets and queries the mask margin for the test mask of the Scope.

### Syntax

```
:CONFigure:MASK:MARGin <integer>[%]
:CONFigure:MASK:MARGin?
```

### Parameter

<integer>[%]
Mask Margin
Range –100 to 100%

### Response Data

<integer>%

### Example of Use

```
:CONF:MASK:MARG 10%
:CONF:MASK:MARG?
>10%
```

## :CONFigure:MASK:MARGin:CONTupdate

### Function

This command sets and queries the mask margin updating method for the mask test of Scope.

### Syntax

```
:CONFigure:MASK:MARGin:CONTupdate 0|1
:CONFigure:MASK:MARGin:CONTupdate?
```

### Parameter

0 Updates the mask margin only once.
1 Updates the mask margin whenever measuring

### Response Data

0|1

### Example of Use

```
:CONF:MASK:MARG:CONT?
>0
```

## :CONFigure:MASK:TYPe

### Function

This command sets and queries the type of the mask used for Scope.

### Syntax

```
:CONFigure:MASK:TYPe <integer>[,<file_name>]
:CONFigure:MASK:TYPe?
```

### Parameter

<integer>

Mask

| | |
|---|---|
| –1 | User Defined |
| 0 | 1GFC |
| 1 | 2GFC |
| 2 | 4GFC |
| 3 | 8GFC |
| 4 | 8GFC_Elect_Rx |
| 5 | 8GFC_Elect_Tx |
| 6 | 10GFC |
| 7 | 10GbE FEC |
| 8 | 1GbE |
| 9 | 2GbE |
| 10 | 10GbE WAN |
| 11 | 10GbE LAN/PHY |
| 12 | 10GFC FEC |
| 13 | OC48/STM16 |
| 14 | OTU-1 |
| 15 | OC192/STM64 |
| 16 | OC192/STM64 FEC(G.975) |
| 17 | OTU-2 1310nm |
| 18 | OTU-2 1550nm |
| 19 | OTU-2 1550nm Expand |
| 20 | OTU-2 Amplified |

If –1 is set to <integer>, specify the file name in <file_name>.

### Response Data

<integer>,<file_name>

### Example of Use

To select Mask other than User Mask:

```
:CONF:MASK:TYP?
>8,""
```

To select "test.txt" as a user mask file:

```
:CONF:MASK:TYP -1,"test.txt"
:CONF:MASK:TYP?
```

```
>-1,"test.txt"
```

## :CONFigure:MASK:UPDate

**Function**

This command updates the position of the mask of Scope.

**Syntax**

```
:CONFigure:MASK:UPDate
```

**Example of Use**

```
:CONF:MASK:UPD
```

## :CONFigure:MASK:USER:LOCation:X1|XDELta

**Function**

This command sets and queries the position for the horizontal direction of the mask of Scope.

**Syntax**

```
:CONFigure:MASK:USER:LOCation:X1|XDELta <numeric>
:CONFigure:MASK:USER:LOCation:X1|XDELta?
```

**Parameter**

X1|XDELta

| | |
|---|---|
| X1 | Position of user adjustment marker X1 |
| XDELta | Interval of user adjustment marker X1 and X2 |

<numeric>

Position in horizontal direction of mask (UI)

When XDELta is selected, the positive value is set.

**Response Data**

<numeric>

**Example of Use**

To set the user adjustment marker X1 to 0.25UI and the user adjustment marker X2 to 1 .25UI, respectively

```
:CONF:MASK:USER:LOC:X1 0.25
:CONF:MASK:USER:LOC:XDEL 1
```

## :CONFigure:MASK:USER:LOCation:Y1|YDELta

**Function**

This command sets and queries the position for the vertical direction of the mask of Scope.

**Syntax**

```
:CONFigure:MASK:USER:LOCation:Y1|YDELta <numeric>
:CONFigure:MASK:USER:LOCation:Y1|YDELta?
```

**Parameter**

Y1|YDELta

| | |
|---|---|
| Y1 | Position of user adjustment marker Y1 |
| YDELta | Interval of user adjustment marker Y1 and Y2 |

<numeric>

Position in vertical direction of mask (mV)

When YDELta is selected, the positive value is set.

**Response Data**

<numeric>

**Example of Use**

To set the user adjustment marker Y1 to 10 mV and the user adjustment marker Y2 to –10 mV, respectively

```
:CONF:MASK:USER:LOC:Y1 10
:CONF:MASK:USER:LOC:YDEL 20
```

## :CONFigure:MASK:USER:MARKer

**Function**

This command sets and queries the On/Off state of the user alignment marker display of the mask of Scope.

**Syntax**

```
:CONFigure:MASK:USER:MARKer 0|1
:CONFigure:MASK:USER:MARKer?
```

**Parameter**

| | |
|---|---|
| 0 | Display Off |
| 1 | Display On |

**Response Data**

0|1

**Example of Use**

```
:CONF:MASK:USER:MARK 1
:CONF:MASK:USER:MARK?
>1
```

## :CONFigure:MEASure:AMPTIME{1|2|3|4}

### Function

This command sets and queries the amplitude/time measurement items to be displayed in the Scope screen.

### Syntax

```
:CONFigure:MEASure:AMPTIME{1|2|3|4} {CHA|CHB},<integer>
:CONFigure:MEASure:AMPTIME{1|2|3|4}?
```

### Parameter

1|2|3|4

CHA: Channel A

CHB: Channel B

<integer> Measurement item

| | |
|---|---|
| 0 | One Level |
| 1 | Zero Level |
| 2 | Eye Amplitude |
| 3 | Eye Height |
| 4 | Crossing |
| 5 | SNR |
| 6 | Average Power (dBm) |
| 7 | Average Power (mW) |
| 8 | Extinction Ratio |
| 9 | Jitter p-p |
| 10 | Jitter RMS |
| 11 | Rise Time |
| 12 | Fall Time |
| 13 | Eye Width |
| 14 | DCD |
| 15 | OMA(mW) |
| 16 | OMA(dBm) |

### Response Data

{CHA|CHB|N/A},{<integer>|N/A}

N/A Blank

### Example of Use

To display the following measurement results on the screen:

- Channel A jitter (p-p)
- Channel A jitter (RMS)
- Channel A Crossing
- Channel A eye amplitude

```
:CONF:MEAS:AMPTIME1 CHA,9
:CONF:MEAS:AMPTIME2 CHA,10
:CONF:MEAS:AMPTIME3 CHA,4
:CONF:MEAS:AMPTIME4 CHA,2
```

To query the measurement result displayed on the screen:

```
:CONF:MEAS:AMPTIME1?
>CHA,9
:CONF:MEAS:AMPTIME2?
>CHA,10
:CONF:MEAS:AMPTIME3?
>CHA,4
:CONF:MEAS:AMPTIME4?
>CHA,2
```

## :CONFigure:MEASure:AREa:DISPlay

### Function

This command sets and queries the Amplitude/Time measurement area display of Scope.

### Syntax

```
:CONFigure:MEASure:AREa:DISPlay <enable>
:CONFigure:MEASure:AREa:DISPlay?
```

### Parameter

0|OFF
1|ON

### Response Data

0|1

### Example of Use

```
:CONF:MEAS:ARE:DISP ON
:CONF:MEAS:ARE:DISP?
>1
```

## :CONFigure:MEASure:AREa:ITEM

### Function

This command sets and queries the measurement item number displayed in the Amplitude/Time measurement area of Scope.

### Syntax

```
:CONFigure:MEASure:AREa:ITEM 1|2|3|4
:CONFigure:MEASure:AREa:ITEM?
```

**Parameter**

| | |
|---|---|
| 1 | measurement item 1 |
| 2 | measurement item 2 |
| 3 | measurement item 3 |
| 4 | measurement item 4 |

**Response Data**

1|2|3|4

**Example of Use**

```
:CONF:MEAS:ARE:ITEM 4
:CONF:MEAS:ARE:ITEM?
>4
```

## :CONFigure:MEASure:CHANnel

**Function**

This command sets and queries the active channel for waveform measurement of Scope.

**Syntax**

```
:CONFigure:MEASure:CHANnel A|B
:CONFigure:MEASure:CHANnel?
```

**Parameter**

| | |
|---|---|
| A: | Channel A |
| B: | Channel B |

**Response Data**

A|B

**Example of Use**

```
:CONF:MEAS:CHAN A
:CONF:MEAS:CHAN?
>A
```

## :CONFigure:MEASure:DEFine

**Function**

This command sets and queries the level measuring the Rise/Fall time on Scope.

**Syntax**

```
:CONFigure:MEASure:DEFine 0|1
```

```
:CONFigure:MEASure:DEFine?
```

**Parameter**

| | |
|---|---|
| 0 | 20/80% |
| 1 | 10/90% |

**Response Data**

0|1

**Example of Use**

```
:CONF:MEAS:DEF 1
:CONF:MEAS:DEF?
>1
```

## :CONFigure:MEASure:EYEBoundary:OFFSet

**Function**

This command sets and queries the horizontal position for 1 and 0 level measurement performed by Scope.

**Syntax**

```
:CONFigure:MEASure:EYEBoundary:OFFSet <numeric>
:CONFigure:MEASure:EYEBoundary:OFFSet?
```

**Parameter**

<numeric>

Position measuring level (UI)

Range 0.00 to 1.00

**Response Data**

<numeric>

**Example of Use**

```
:CONF:MEAS:EYEB:OFFS 0.3
:CONF:MEAS:EYEB:OFFS?
>0.3
```

## :CONFigure:MEASure:EYEBoundary:WIDTh

**Function**

This command sets and queries the horizontal width measuring 1 level and 0 level on Scope.

**Syntax**

```
:CONFigure:MEASure:EYEBoundary:WIDTh <numeric>
```

```
:CONFigure:MEASure:EYEBoundary:WIDTh?
```

**Parameter**

<numeric>

Width measuring level (UI)

Range 0.00 to 1.00

**Response Data**

<numeric>

**Example of Use**

```
:CONF:MEAS:EYEB:WIDT 0.20
:CONF:MEAS:EYEB:WIDT?
>0.20
```

3

Message List

## :CONFigure:MEASure:TRANsition:CORRect:FACTor

**Function**

This command sets and queries the correction factor for rise/fall time of Scope.

**Syntax**

```
:CONFigure:MEASure:TRANsiton:CORRect:FACTor <numeric>
:CONFigure:MEASure:TRANsiton:CORRect:FACTor?
```

**Parameter**

<numeric>

Correction factor

Range 0.0 to 9999.9

**Response Data**

<numeric>

**Example of Use**

```
:CONF:MEAS:TRAN:CORR:FACT 0.0
:CONF:MEAS:TRAN:CORR:FACT?
>0.0
```

## :CONFigure:MEASure:TRANsition:CORRection

**Function**

This command sets and queries the On/Off state of use of the correction factor for rise/fall time of Scope.

**Syntax**

```
:CONFigure:MEASure:TRANsiton:CORRection <enable>
:CONFigure:MEASure:TRANsiton:CORRection?
```

**Parameter**

0|OFF
1|ON

**Response Data**

0|1

**Example of Use**

To use the correction factor:

```
:CONF:MEAS:TRAN:CORR ON
:CONF:MEAS:TRAN:CORR?
>1
```

## :CONFigure:MEASure:TYPe

**Function**

This command sets the measurement items on Scope and measures the set item.
The query returns the current measurement item.

**Syntax**

```
:CONFigure:MEASure:TYPe
AMPHistogram|AMPMAsk|AMPTIME|HISTogram|MASK|OFF
:CONFigure:MEASure:TYPe?
```

**Parameter**

| | |
|---|---|
| AMPHistogram | Amplitude/Time & Histogram |
| AMPMask | Amplitude/Time & Mask Test |
| AMPTIME | Amplitude/Time |
| HISTogram | Histogram |
| MASK | Mask Test |
| OFF | Off |

**Response Data**

AMPHistogram|AMPMAsk|AMPTIME|HISTogram|MASK|OFF

**Example of Use**

```
:CONF:MEAS:TYP AMPM
:CONF:MEAS:TYP?
>AMPMask
```

***Note:***

As for the following subsystem query message, set the measurement items using the :CONFigure:MEASure:TYPe before the measurement.

```
:FETCH:AMPLitude
:FETCH:HISTogram
:FETCH:MASK
:FETCH:TIME
```

As for the query message acquiring the following measurement result, do not set the measurement items using the :CONFigure:MEASure:TYPe before the measurement.

```
:MEASure:AMPLitude
:MEASure:HISTogram:AMPLitude
:MEASure:HISTogram:TIME
:MEASure:TIME
:MEASure:MASK
```





## :CONFigure:SKEW:CHA|CHB

### Function

This command sets and queries the skew on Scope.

### Syntax

```
:CONFigure:SKEW:CHA|CHB <numeric>
:CONFigure:SKEW:CHA|CHB?
```

### Parameter

CHA|CHB

| | |
|---|---|
| CHA | Channel A |
| CHB | Channel B |

<numeric>

| | |
|---|---|
| Skew | –999.9 to 999.9 ps |

### Response Data

<numeric>

### Example of Use

```
:CONF:SKEW:CHA 6.4
:CONF:SKEW:CHA?
>6.4
```

## :CONFigure:TRACking:DRATe

**Function**

This command sets and queries the On/Off state of tracking of bit rate and clock rate of Scope.

**Syntax**

```
:CONFigure:TRACking:DRATe <enable>
:CONFigure:TRACking:DRATe?
```

**Parameter**

0|OFF
1|ON

**Response Data**

0|1

**Example of Use**

```
:CONF:TRAC:DRAT ON
:CONF:TRAC:DRAT?
>1
```

## :CONFigure:TRACking:DRATe:MASTer

**Function**

This command sets and queries the synchronization source for tracking of bit rate and clock rate of Scope.

**Syntax**

```
:CONFigure:TRACking:DRATe:MASTer 0|1|2|3
:CONFigure:TRACking:DRATe:MASTer?
```

**Parameter**

| | |
|---|---|
| 0 | PPG1 |
| 1 | ED1 |
| 2 | PPG2 |
| 3 | ED2 |

**Response Data**

0|1|2|3

**Example of Use**

```
:CONF:TRAC:DRAT:MAST 2
:CONF:TRAC:DRAT:MAST?
>2
```

## :CONFigure:TRACking:PATLength

**Function**

This command sets and queries the On/Off state of pattern length tracking of Scope.

**Syntax**

```
:CONFigure:TRACking:PATLength <enable>
:CONFigure:TRACking:PATLength?
```

**Parameter**

0|OFF
1|ON

**Response Data**

0|1

**Example of Use**

```
:CONF:TRAC:PATL OFF
:CONF:TRAC:PATL?
>0
```

## :CONFigure:TRACking:PATLength:MASTer

**Function**

This command sets and queries the synchronization source for pattern length tracking of Scope.

**Syntax**

```
:CONFigure:TRACking:PATLength:MASTer 0|1|2|3|4|5|6|7
:CONFigure:TRACking:PATLength:MASTer?
```

**Parameter**

| | |
|---|---|
| 0 | PPG1 |
| 1 | ED1 |
| 2 | PPG2 |
| 3 | ED2 |
| 4 | PPG3 |
| 5 | ED3 |
| 6 | PPG4 |
| 7 | ED4 |

**Response Data**

0|1|2|3|4|5|6|7

**Example of Use**

```
:CONF:TRAC:PATL:MAST 1
:CONF:TRAC:PATL:MAST?
>1
```

## 3.9.5 MEASure subsystem

### :MEASure:AMPLitude?

#### Function

This command starts measurement with the sampling mode and measurement function of Scope, respectively set to [Eye] and amplitude.

#### Syntax

```
:MEASure:AMPLitude?
```

#### Response Data

<numeric>,<numeric>,<numeric>,<numeric>,<numeric>,<numeric>,{<numeric>|"N/A"},{<numeric>|"N/A"},{<numeric>|"N/A"},{<numeric>|"N/A"},{<numeric>|"N/A"}

The response is output in the following order.

| Item | Unit | Details |
|---|---|---|
| One Level | mV/μW | 1 level |
| Zero Level | mV/μW | 0 level |
| Eye amplitude | mV/μW | Eye width |
| Eye height | mV/μW | Eye height |
| Crossing | % | Cross ratio |
| SNR | - | Signal-to-noise ratio |
| Average Power | μW | Averaging power |
| Average Power | dBm | Averaging power |
| Extinction Ratio | dB | Extinction ratio |
| OMA(mW) | mW | Optical modulation amplitude |
| OMA(dBm) | dBm | Optical modulation amplitude |

#### Example of Use

```
:MEAS:AMPL?
>35.2,-15.1,50.3,46.2,46,10.5,"N/A","N/A","N/A","N/A","N/A"
```

***Note:***

Before using this command, sets the Accumulation Type of the Setup dialog box to [Limited] and the measurement time to [Limit Type].

After passing the set measurement time period, this command queries the measurement results.

The measurement results divided in 11 items are returned: One Level, Zero Level, Eye amplitude, Eye height, Crossing, SNR, Average power (μW), Average power (dBm), Extinction ratio,OMA(mW), and OMA(dBm).

The following items are available for the optical channel. When the optical channel is not set to active channel, "N/A" is returned.

- Average power (μW)
- Average power (dBm)
- Extinction ratio
- OMA (mW)
- OMA (dBm)

To query the current measurement items,
use :CONFigure:MEASure:TYPe?.

The response does not return during the set measurement time. To prevent occurrence of timeout errors, set the sufficiently-long timeout for remote interface, with respect to the measurement time.

## :MEASure:TIME?

### Function

This command sets the Sampling mode of Scope to [Eye] and the measurement function to the time and starts the measurement.

### Syntax

```
:MEASure:TIME?
```

### Response Data

<numeric>,<numeric>,<numeric>,<numeric>,<numeric>,<numeric>
The response is output in the following order.

| Item | Unit | Details |
|---|---|---|
| Jitter p-p | ps | Jitter (Peak to peak) |
| Jitter RMS | ps | Jitter (RMS) |
| Rise Time | ps | Rise-time |
| Fall Time | ps | Fall-time |
| Eye Width | ps | Eye width |
| DCD | % | Duty Cycle Distortion |

### Example of Use

```
:MEAS:TIME?
>66.2,15.3,128.2,133.6,208.6,47.2
```

***Note:***

Before using this command, sets the Accumulation Type of the Setup dialog box to [Limited] and the measurement time to [Limit Type].

After passing the set measurement time period, this command queries the measurement results.

To query the measurement results, use :FETCh:TIME:MEASurement?.

To read the histogram measurement results of the time individually, use the following command;

:FETCh:HISTogram:TIME:MEASurement?

:FETCh:HISTogram:TIME:MEAN?

:FETCh:HISTogram:AMPLitude:PPeak?

:FETCh:HISTogram:AMPLitude:STDDeviation?

:FETCh:HISTogram:TIME:HITS?

The response does not return during the set measurement time. To prevent occurrence of timeout errors, set the sufficiently-long timeout for remote interface, with respect to the measurement time.



## :MEASure:HISTogram:AMPLitude?

### Function

This commad sets the Sampling mode of Scope to [Eye] and the measurement function to the histogram measurement function at the amplitude axis and starts the measurement.

### Syntax

```
:MEASure:HISTogram:AMPLitude?
```

### Response Data

<numeric>,<numeric>,<numeric>,<integer>

The response is output in the following order.

| Item | Unit | Content |
|---|---|---|
| Mean | mV/μW | Mean value |
| StdDiv | mV/μW | Standard deviation |
| Peak to Peak | mV/μW | Amplitude |
| Hits | | Number of sample in area |

### Example of Use

```
:MEAS:HIST:AMPL?
>32.1,4.53,28.1,89632
```

***Note:***

Before using this command, sets the Accumulation Type of the Setup dialog box to [Limited] and the measurement time to [Limit Type].

After passing the set measurement time period, this command queries the measurement results.

To read the histogram measurement results of the amplitude individually, use the following commands:

:FETCh:HISTogram:AMPLitude:MEASurement?

:FETCh:HISTogram:AMPLitude:MEAN?

:FETCh:HISTogram:AMPLitude:STDDeviation?

:FETCh:HISTogram:AMPLitude:PPeak?

:FETCh:HISTogram:AMPLitude:HITS?

To confirm whether to set the histogram measurement axis to the amplitude, use :CONFigure:HISTogram:AXIS?.

The response does not return during the set measurement time. To prevent occurrence of timeout errors, set the sufficiently-long timeout for remote interface, with respect to the measurement time.

## :MEASure:HISTogram:TIME?

### Function

This command starts measurement with the sampling mode and measurement function of Scope, respectively set to [Eye] and time-axis histogram measurement.

### Syntax

```
:MEASure:HIStogram:TIME?
```

### Response Data

<numeric>,<numeric>,<numeric>,<integer>

The response data is output in the following order.

| Item | Unit | Content |
|---|---|---|
| Mean | ps | Mean value |
| StdDiv | ps | Standard deviation |
| Peak to Peak | ps | Amplitude |
| Hits | | Number of sample in area |

### Example of Use

```
:MEAS:HIST:TIME?
>1.53,0.022,0.081,6831
```

***Note:***

Before using this command, sets the Accumulation Type of the Setup dialog box to [Limited] and the measurement time to [Limit Type].

After passing the set measurement time period, this command queries the measurement results.

To read the histogram measurement results of the time individually, use the following command;

:FETCh:HISTogram:TIME:MEASurement?

:FETCh:HISTogram:TIME:MEAN?

:FETCh:HISTogram:TIME:STDDeviation?

:FETCh:HISTogram:TIME:PPeak?

:FETCh:HISTogram:TIME:HITS?

To confirm whether to set the histogram measurement axis to the time, use the following command; :CONFigure:HISTogram:AXIS?.

As for the current measurement,
use :CONFigure:MEASurement:TYPe?.

The response does not return during the set measurement time. To prevent occurrence of timeout errors, set the sufficiently-long timeout for remote interface, with respect to the measurement time.

## :MEASure:MASK?

### Function

This command starts measurement with the sampling mode and measurement function of Scope, respectively set to [Eye] and mask test.

### Syntax

```
:MEASure:MASK?
```

### Response Data

<integer>,<integer>
The response is output in the following order.

| Item | Unit | Details |
|---|---|---|
| Total Samples | – | Total sample count |
| Failed | – | Sample count in the mask area |

**Example of Use**

```
:MEAS:MASK?
>16831,0
```

***Note:***

Before using this command, sets the Accumulation Type of the Setup dialog box to [Limited] and the measurement time to [Limit Type].

After passing the set measurement time period, this command queries the measurement results.

The measurement result can be queried using :FETCh:MASK:MEASurement?. To query each of measurement results separately, use the following commands:

:FETCh:MASK:SAMPles:TOTal?

:FETCh:MASK:SAMPles:FAILed[:{BOTTom|CENTer|TOP}]?

Each query does not return a response until the specified period of measurement time has elapsed. To prevent occurrence of timeout errors, set the sufficiently-long timeout for remote interface, with respect to the measurement time.

## :MEASure:MASK:MARGin?

**Function**

This command queries the measured mask margin if the measurement function of Scope is mask test.

**Syntax**

```
:MEASure:MASK:MARGin?
```

**Response Data**

<integer>

Range –100 to 100

Unit %

**Example of Use**

```
:MEAS:MASK:MARG?
>12
```

## 3.9.6 CALCulate subsystem

### :CALCulate:CHANnel:MATH

**Function**

This command sets and queries the calculation between channels of Scope.

**Syntax**

```
:CALCulate:CHANnel:MATH <enable>
:CALCulate:CHANnel:MATH?
```

**Parameter**

0|OFF
1|ON

**Response Data**

1|0

**Example of Use**

```
:CALC:CHAN:MATH OFF
:CALC:CHAN:MATH?
>0
```

### :CALCulate:CHANnel:MATH:DEFine

**Function**

This command sets and queries the calculation formula between channels of Scope.

**Syntax**

```
:CALCulate:CHANnel:MATH:DEFine 0|1|2
:CALCulate:CHANnel:MATH:DEFine?
```

**Parameter**

0: Channel A + Channel B
1: Channel A – Channel B
2: Channel B– Channel A

**Response Data**

0|1|2

**Example of Use**

```
:CALC:CHAN:MATH:DEF?
>1
```

## :CALCulate:MARKer:AOFF

**Function**

This command deletes the marker displayed in the Scope screen.

**Syntax**

```
:CALCulate:MARKer:AOFF
```

**Example of Use**

```
:CALC:MARK:AOFF
```

## :CALCulate:MARKer:CENTer

**Function**

This command displays all markers of Scope at the center of the screen.

**Syntax**

```
:CALCulate:MARKer:CENTer
```

**Example of Use**

```
:CALC:MARK:CENT
```

## :CALCulate:MARKer:Y1|Y2

**Function**

This command sets and queries the On/Off state of the amplitude marker Y1/Y2 display of Scope.

**Syntax**

```
:CALCulate:MARKer:Y1|Y2 <enable>
:CALCulate:MARKer:Y1|Y2?
```

**Parameter**

| | |
|---|---|
| Y1\|Y2 | Marker |
| 0\|OFF | |
| 1\|ON | |

**Response Data**

0|1

**Example of Use**

```
:CALC:MARK:Y2 ON
:CALC:MARK:Y2?
>1
```

## :CALCulate:MARKer:LOCation:CHA|CHB:Y1|Y2

**Function**

This command sets and queries the position of the amplitude marker Y1/Y2 of Scope.

If the specified marker is not displayed, execution of this command displays the marker at the specified position.

**Syntax**

```
:CALCulate:MARKer:LOCation:CHA|CHB:Y1|Y2 <numeric>
:CALCulate:MARKer:LOCation:CHA|CHB:Y1|Y2?
```

**Parameter**

CHA|CHB

| | |
|---|---|
| CHA | Channel A |
| CHB | Channel B |
| Y1|Y2 | Marker |

<numeric>

Marker position (Amplitude)

Unit mV (Electrical input)
μW (Optical input)

Range
mV: –1000 to 1000
μW: Value obtained by dividing –1000 to 1000 by O/E conversion gain

**Response Data**

<numeric>|"Marker Off"

The response data when the marker is not displayed is "N/A".

**Example of Use**

To query the value of the marker Y2 for the Channel A waveform:

```
:CALC:MARK:LOC:CHA:Y2?
>55.35
```

## :CALCulate:MARKer:LOCation:CHA|CHB:YDELta?

**Function**

This command queries the marker difference between Y1 and Y2.

**Syntax**

```
:CALCulate:MARKer:LOCation:CHA|CHB:YDELta?
```

**Parameter**

CHA|CHB

| | |
|---|---|
| CHA | Channel A |

CHB Channel B

**Response Data**

<numeric>|"Marker Off"

<numeric>

Marker position (Amplitude)

Unit mV (Electrical input)

μW (Optical input)

"Marker Off" is returned if one or both of markers are not displayed.

**Example of Use**

To query the marker difference between Y1 and Y2 for Channel B

```
:CALC:MARK:LOC:CHB:YDEL?
>15.3
```

## :CALCulate:MARKer:X1|X2

**Function**

This command sets and queries the On/Off state of the time marker X1/X2 display of Scope.

**Syntax**

```
:CALCulate:MARKer:X1|X2 <enable>
:CALCulate:MARKer:X1|X2?
```

**Parameter**

X1|X2 Marker

0|OFF

1|ON

**Response Data**

0|1

**Example of Use**

```
:CALC:MARK:X1 ON
:CALC:MARK:X1?
>1
```

## :CALCulate:MARKer:LOCation:X1|X2

**Function**

This command sets and queries the position of the time marker X1/X2 of Scope.

If the specified marker is not displayed, execution of this command displays the marker at the specified position.

**Syntax**

```
:CALCulate:MARKer:LOCation:X1|X2 <numeric>
:CALCulate:MARKer:LOCation:X1|X2?
```

**Parameter**

X1|X2 Marker

<numeric>

Time displaying marker

Unit UI or ps

The range is as follows.

| Unit | Range |
|---|---|
| ps | 0 to 4294967295 |
| UI | 0 to 4294967 |

**Response Data**

<numeric>|"Marker Off"

The response data when the marker is not displayed is "Marker Off".

**Example of Use**

To display the marker X2 to the 1.5 UI position

```
:CALC:MARK:LOC:X2 1.5
```

## :CALCulate:MARKer:LOCation:XDELta?

**Function**

This command queries the time marker difference between X1 and X2.

**Syntax**

```
:CALCulate:MARKer:LOCation:XDELta?
```

**Response Data**

<numeric>|"Marker Off"

<numeric>

Time displaying marker

Unit: UI or ps

The response data when both or either of markers are not displayed is "Marker Off".

**Example of Use**

```
:CALC:MARK:LOC:XDEL?
>152.33
```

3

Message List

## 3.9.7 FETCh subsystem

### :FETCh:AMPLitude:MEASurement?

**Function**

This command queries the results of all measurement items for the amplitude measurement function of Scope.

**Syntax**

```
:FETCh:AMPLitude:MEASurement?
```

**Response Data**

<numeric>,<numeric>,<numeric>,<numeric>,<numeric>,<numeric>,{<numeric>|N/A},{<numeric>|N/A},{<numeric>|N/A},{<numeric>|N/A},{<numeric>|N/A}……|N/A

The measurement items are 11 items; 1 level, 0 level, eye amplitude, eye height, crossing ratio, signal to noise ratio, average power (μW), average power (dBm), extinct ratio, OMA (mW), and OMA (dBm).

The current value, average, standard deviation, and minimum/maximum values for this 11 measurement items can be read out.

The response consists of 55 items and is output in the following order.

**Table 3.9.7-1 Data Order**

| Item | Unit | Current value | Average | Standard deviation | Minimum value | Maximum value |
|---|---|---|---|---|---|---|
| One Level | mV/μW | 1 | 12 | 23 | 34 | 45 |
| Zero Level | mV/μW | 2 | 13 | 24 | 35 | 46 |
| Eye amplitude | mV/μW | 3 | 14 | 25 | 36 | 47 |
| Eye height | mV/μW | 4 | 15 | 26 | 37 | 48 |
| Crossing | % | 5 | 16 | 27 | 38 | 49 |
| SNR | | 6 | 17 | 28 | 39 | 50 |
| Average Power | dBm | 7 | 18 | 29 | 40 | 51 |
| Average Power | μW | 8 | 19 | 30 | 41 | 52 |
| Extinction Ratio | | 9 | 20 | 31 | 42 | 53 |
| OMA (mW) | mW | 10 | 21 | 32 | 43 | 54 |
| OMA (dBm) | dBm | 11 | 22 | 33 | 44 | 55 |

**Example of Use**

```
:FETC:AMPL:MEAS?
>35.12,-15.01,50.53,46.29,46,10.5,N/A,N/A,N/A,N/A,N/A,
36.98,-14.85,52.66,40.44,43.09,10.22,N/A,N/A,N/A,N/A,N/A,
0.23,0.15,1.46,1.25,0.69,0.38,N/A,N/A,N/A,N/A,N/A,
36.29,-15.3,48.28,36.39,41.02,9.08,N/A,N/A,N/A,N/A,N/A,
37.67,-14.4,57.04,44.19,45.16,11.36,N/A,N/A,N/A,N/A,N/A
```

***Note:***

When an optical channel is not set to an active channel, the following data becomes N/A.

- Average power (μW)
- Average power (dBm)
- Extinct rate
- OMA (mW)
- OMA (dBm)

The response data is N/A when the measurement item is set to the following:

- Histogram
- Mask
- OFF



## :FETCh:AMPLitude:LEVel:ONE?

**Function**

This command queries One Level for the amplitude measurement function of the Scope.

**Syntax**

```
:FETCh:AMPLitude:LEVel:ONE?
```

**Response Data**

<numeric>,<numeric>,<numeric>,<numeric>,<numeric>|N/A
This indicates the 1 level, its average, standard deviation, and minimum/maximum values.
The electrical channel is set in mV, and the optical channel is set in the μW unit.

**Example of Use**

```
:FETC:AMPL:LEV:ONE?
>35.16,35.11,0.11,34.78,35.44
```

***Note:***

he response data in the following cases is N/A.

- The measurement item is set to Histogram, Mask, or Off.

## :FETCh:AMPLitude:LEVel:ZERO?

**Function**

This command queries Zero Level for the amplitude measurement function of the Scope.

**Syntax**

```
:FETCh:AMPLitude:LEVel:ZERO?
```

**Response Data**

<numeric>,<numeric>,<numeric>,<numeric>,<numeric> | N/A

The response consists of the following Zero Level values, in order: current value, average, standard deviation, minimum value, and maximum value.

Unit mV (Electrical channel)
μW (Optical channel)

**Example of Use**

```
:FETC:AMPL:LEV:ZERO?
>-15.12,-15.20,0.05,-15.35,-15.05
```

***Note:***

The response data in the following cases is N/A.

- The measurement item is set to Histogram, Mask, or Off.

## :FETCh:AMPLitude:EYEAmplitude?

**Function**

This command queries the eye amplitude for the amplitude measurement function of the Scope.

**Syntax**

```
:FETCh:AMPLitude:EYEAmplitude?
```

**Response Data**

<numeric>,<numeric>,<numeric>,<numeric>,<numeric> | N/A

The response consists of the following eye amplitude values, in order: current value, average, standard deviation, minimum value, and maximum value.

The electrical channel is set in mV, and the optical channel is set in the μW unit.

**Example of Use**

```
:FETC:AMPL:EYEA?
>55.22,54.89,0.12,54.53,55.25
```

***Note:***

The response data in the following cases is N/A.

- The measurement item is set to Histogram, Mask, or Off.

## :FETCh:AMPLitude:EYEHeight?

### Function

This command queries the eye height for the amplitude measurement function of the Scope.

### Syntax

```
:FETCh:AMPLitude:EYEHeight?
```

### Response Data

<numeric>,<numeric>,<numeric>,<numeric>,<numeric> | N/A

The response consists of the following eye height values, in order: current value, average, standard deviation, minimum value, and maximum value. The electrical channel is set in mV, and the optical channel is set in the μW unit.

### Example of Use

```
:FETC:AMPL:EYEH?
>45.81,45.77,0.08,45.53,46.01
```

***Note:***

The response data in the following cases is N/A.

- The measurement item is set to Histogram, Mask, or Off.

## :FETCh:AMPLitude:CROSsing?

### Function

This command queries the eye cross ratio for the amplitude measurement function of the Scope.

### Syntax

```
:FETCh:AMPLitude:CROSsing?
```

### Response Data

<numeric>,<numeric>,<numeric>,<numeric>,<numeric> | N/A

The response consists of the following eye cross ratio values, in order: current value, average, standard deviation, minimum value, and maximum value.

Unit %
Range 0.00 to 100.00

**Example of Use**

```
:FETC:AMPL:CROS?
>46.01,45.80,0.19,45.27,46.41
```

***Note:***

The response data in the following cases is N/A.

- The measurement item is set to Histogram, Mask, or Off.

## :FETCh:AMPLitude:SNR?

**Function**

This command queries the signal-to-noise ratio (SNR) for the amplitude measurement function of the Scope.

**Syntax**

```
:FETCh:AMPLitude:SNR?
```

**Response Data**

<numeric>,<numeric>,<numeric>,<numeric>,<numeric> | N/A
The response consists of the following SNR values, in order: current value, average, standard deviation, minimum value, and maximum value.
There is no unit.

**Example of Use**

```
:FETC:AMPL:SNR?
>10.08,10.11,0.19,9.55,10.70
```

***Note:***

The response data in the following cases is N/A.

- The measurement item is set to Histogram, Mask, or Off.

## :FETCh:AMPLitude:AVEPower?

**Function**

This command queries the optical average power input to the O/E converter.

**Syntax**

```
:FETCh:AMPLitude:AVEPower?
```

**Response Data**

<numeric>,<numeric>,<numeric>,<numeric>,<numeric>,<numeric>,<numeric>,<numeric>,<numeric>,<numeric> | N/A

This indicates the optical power, average, its standard deviation, and minimum/maximum values in μW and dBm.

The response data is output in the following order.

| Order | Item | Unit | Content |
|---|---|---|---|
| 1 | Average Power | μW | Average power |
| 2 | Average Power | dBm | Average power |
| 3 | Average | μW | Average of average power |
| 4 | Average | dBm | Average of average power |
| 5 | Std.Dev. | μW | Standard deviation of average power |
| 6 | Std.Dev. | dBm | Standard deviation of average power |
| 7 | Min | μW | Minimum value of average power |
| 8 | Min | dBm | Minimum value of average power |
| 9 | Max | μW | Maximum value of average power |
| 10 | Max | dBm | Maximum value of average power |

**Example of Use**

```
:FETC:AMPL:AVEP?
>25.00,-16.02,25.50,-15.93,0.02,0.05,24.86,-16.05,26.12,
-15.83
```

***Notes:***

- A valid value is returned only when an optical interface is specified as the active channel.
- The response data in the following cases is N/A.
  - The measurement item is set to Histogram, Mask, or Off.
  - The O/E converter doesn't exist.
  - An active channel of the waveform measurement is set to A.

## :FETCh:AMPLitude:EXTRatio?

**Function**

This command queries the extinction ratio for the amplitude measurement function of the Scope.

**Syntax**

```
:FETCh:AMPLitude:EXTRatio?
```

**Response Data**

<numeric>,<numeric>,<numeric>,<numeric>,<numeric> | N/A

The response consists of the following extinction ratio values, in order: current value, average, standard deviation, minimum value, and maximum value.
There is no unit.

**Example of Use**

```
:FETC:AMPL:EXTR?
>6.82,6.77,0.13,6.38,7.16
```

***Notes:***

- This message can query a valid value only when an optical interface is specified as the active channel.

- The response data in the following cases is N/A.

  - The measurement item is set to Histogram, Mask, or Off.
  - The O/E converter doesn't exist.
  - An active channel of the waveform measurement is set to A.

## :FETCh:AMPLitude:OMA:MW?

**Function**

This command queries the optical modulation amplitude (OMA) of the Scope amplitude measurement function, in the mW unit.

**Syntax**

```
:FETCh:AMPLitude:OMA:MW?
```

**Response Data**

<numeric>,<numeric>,<numeric>,<numeric>,<numeric>|N/A
The response consists of the following OMA values, in order: current value, average, standard deviation, minimum value, and maximum value.

**Example of Use**

```
:FETC:AMPL:OMA:MW?
>0.15,0.16,0.03,0.06,0.25
```

***Note:***

The response data in the following cases is N/A.

- The measurement item is set to Histogram, Mask, or Off.
- The O/E converter doesn't exist.
- An active channel of the waveform measurement is set to A.

## :FETCh:AMPLitude:OMA:DBM?

**Function**

This command queries the OMA of the Scope amplitude measurement function, in the dBm unit.

**Syntax**

```
:FETCh:AMPLitude:OMA:DBM?
```

**Response Data**

<numeric>,<numeric>,<numeric>,<numeric>,<numeric> | N/A

The response consists of the following OMA values, in order: current value, average, standard deviation, minimum value, and maximum value.

**Example of Use**

```
:FETC:AMPL:OMA:DBM?
>-8.22,-8.24,0.21,-8.85,-7.59
```

***Note:***

The response data in the following cases is N/A.

- The measurement item is set to Histogram, Mask, or Off.
- The O/E converter doesn't exist.
- An active channel of the waveform measurement is set to A.

## :FETCh:AMPTime:QUEStionableeye?

**Function**

This command queries if "EYE?" is displayed on the Scope amplitude measurement screen.

**Syntax**

```
:FETCh:AMPLitude:QUEStionableeye?
```

**Response Data**

0 | 1

0: Not display

1: Display

**Example of Use**

```
:FETC:AMPT:QUES?
>1
```

## :FETCh:HISTogram:AMPLitude:MEASurement?

### Function

This command queries four results of the histogram measurement of the amplitude axis in the Scope histogram measurement function.

### Syntax

```
:FETCh:HISTogram:AMPLitude:MEASurement?
```

### Response Data

<numeric>,<numeric>,<numeric>,<integer> | N/A

The response data is output in the following order.

| Item | Unit | Content |
|---|---|---|
| Mean | mV/μW | Mean value |
| StdDiv | mV/μW | Standard deviation |
| Peak to Peak | mV/μW | Amplitude |
| Hits | | Number of samples in area |

### Example of Use

```
:FETC:HIST:AMPL:MEAS?
>32.1,4.53,28.1,89632
```

***Notes:***

- This message queries data that the measurement ends.
- To execute the histogram measurement of the amplitude axis and to update the result, use the following command; MEASure:HISTogram:AMPLitude?.
- When the measurement item is not set to the following or histogram at the time axis, the response data is N/A.
  - Amplitude/Time
  - Amplitude/Time&Mask
  - Mask
  - OFF

## :FETCh:HISTogram:AMPLitude:MEAN?

## :FETCh:HISTogram:AMPLitude:STDDeviation?

## :FETCh:HISTogram:AMPLitude:PPeak?

## :FETCh:HISTogram:AMPLitude:HITS?



### Function

:FETCh:HISTogram:AMPLitude:MEAN? queries the mean amplitude of the histogram measurement on the amplitude axis in the Scope histogram measurement function.
:FETCh:HISTogram:AMPLitude:STDDeviation? queries the standard deviation of the histogram measurement result on the amplitude axis in the Scope histogram measurement function.
:FETCh:HISTogram:AMPLitude:PPeak? queries the amplitude (Peak to Peak) of the histogram measurement on the amplitude axis in the Scope histogram measurement function.
:FETCh:HISTogram:AMPLitude:HITS? queries the number of samples (this number of samples is called the number of hits) in the area of the histogram measurement on the amplitude axis in the function of the measurement of the histogram of Scope.

### Syntax

```
:FETCh:HISTogram:AMPLitude:MEAN?
:FETCh:HISTogram:AMPLitude:STDDeviation?
:FETCh:HISTogram:AMPLitude:PPeak?
:FETCh:HISTogram:AMPLitude:HITS?
```

### Response Data

<numeric>|N/A
<numeric>|N/A
<numeric>|N/A
<integer>|N/A

The units, mV and µW, are used for the electrical and optical interfaces, respectively.

### Example of Use

```
:FETC:HIST:AMPL:MEAN?
>32.1
:FETC:HIST:AMPL:STDD?
>4.53
:FETC:HIST:AMPL:PP?
>28.1
:FETC:HIST:AMPL:HITS?
```

```
>89632
```

***Notes:***

- This message queries data that the measurement ends.
- To execute the histogram measurement of the amplitude axis and to update the result, use the following command: MEASure:HISTogram:AMPLitude?.
- When the measurement item is not set to the following or histogram on the time axis, the response data is N/A.
  - Amplitude/Time
  - Amplitude/Time&Mask
  - Mask
  - OFF

## :FETCh:HISTogram:TIME:MEASurement?

**Function**

This command queries the results of four measurement items for time-axis histogram measurement of Scope.

**Syntax**

```
:FETCh:HISTogram:TIME:MEASurement?
```

**Response Data**

<numeric>,<numeric>,<numeric>,<integer>|N/A

The response data is output in the following order.

| Item （Unit） | Content |
|---|---|
| Mean (ps) | Mean value |
| StdDiv (ps) | Standard deviation |
| Peak to Peak (ps) | Amplitude |
| Hits | Number of samples in area |

**Example of Use**

```
:FETC:HIST:TIME:MEAS?
>1.53,0.022,0.081,6831
```

***Notes:***

- This message queries data that the measurement ends.
- To execute the histogram measurement of the time axis and to update the result, use the following command; :MEASure:HISTogram:TIME?.

- When the measurement item is not set to the following or histogram at the amplitude axis, the response data is N/A.
  - Amplitude/Time
  - Amplitude/Time&Mask
  - Mask
  - OFF



## :FETCh:HISTogram:TIME:MEAN?
## :FETCh:HISTogram:TIME:STDDeviation?
## :FETCh:HISTogram:TIME:PPeak?
## :FETCh:HISTogram:TIME:HITS?

**Function**

:FETCh:HISTogram:TIME:MEAN? queries the mean value of the histogram measurement on the time axis in the Scope histogram measurement function. The unit is ps.
:FETCh:HISTogram:TIME:STDDeviation? queries the standard deviation of the histogram measurement results on the time axis in the Scope histogram measurement function. The unit is ps or UI.
:FETCh:HISTogram:TIME:PPeak? queries the time difference (Peak to Peak) of the histogram measurement on the time axis in the Scope histogram measurement function. The unit is ps.
:FETCh:HISTogram:TIME:HITS? queries the sample count of the histogram measurement on the time axis in the Scope histogram measurement function.

**Syntax**

```
:FETCh:HISTogram:TIME:MEAN?
:FETCh:HISTogram:TIME:STDDeviation?
:FETCh:HISTogram:TIME:PPeak?
:FETCh:HISTogram:TIME:HITS?
```

**Response Data**

<numeric>|N/A
<numeric>|N/A
<numeric>|N/A
<integer>|N/A

**Example of Use**

```
:FETC:HIST:TIME:MEAN?
>1.53
:FETC:HIST:TIME:STDD?
>0.022
```

```
:FETC:HIST:TIME:PP?
>0.081
:FETC:HIST:TIME:HITS?
>6831
```

***Notes:***

- This message queries data that the measurement ends.
- To execute the histogram measurement of the time axis and to update the result, use the following command; :MEASure:HISTogram:TIME?.
- When the measurement item is not set to the following or histogram at the amplitude axis, the response data is N/A.
  - Amplitude/Time
  - Amplitude/Time&Mask
  - Mask
  - OFF

## :FETCh:MASK:MEASurement?

**Function**

This command queries the most recent results for the mask measurement of the Scope.

**Syntax**

```
:FETCh:MASK:MEASurement?
```

**Response Data**

<integer>,<integer> | N/A

The results of the following two measurement items are returned in order:

- Total Samples: Total number of samples
- Total Failed Samples: Total number of samples in mask area

**Example of Use**

```
:FETC:MASK:MEAS?
>16831,0
```

***Notes:***

- To update the result after executing the mask test, use:MEASure:MASK?.
- When the measurement item is not set to the following, the response data is N/A.

- Amplitude/Time
- Amplitude/Time&Histogram
- Histogram
- OFF

## :FETCh:MASK:SAMPles:TOTal?

**Function**

For the mask measurement of the Scope, this command queries total sample count.

**Syntax**

```
:FETCh:MASK:SAMPles:TOTal?
```

**Response Data**

<integer>|N/A

**Example of Use**

```
:FETC:MASK:SAMP:TOT?
>16831
```

***Notes:***

- To update the result after executing the mask test, use :MEASure:MASK?.

- When the measurement item is not set to the following, the response data is N/A.

  - Amplitude/Time
  - Amplitude/Time&Histogram
  - Histogram
  - OFF

## :FETCh:MASK:SAMPles:FAILed[:{BOTTom|CENTer|TOP}]?

**Function**

This command queries the number of samples in the mask area, in mask measurement by Scope.

**Syntax**

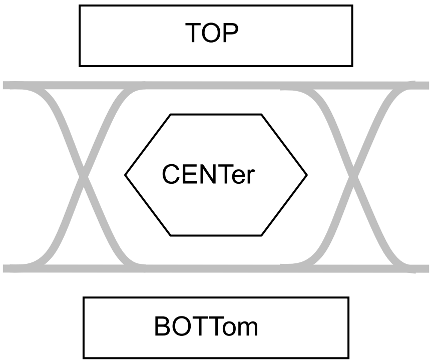
```
:FETCh:MASK:SAMPles:FAILed[:{BOTTom|CENTer|TOP}]?
```

**Parameter**

BOTTom|CENTer|TOP

| | |
|---|---|
| BOTTom | The number of samples in the bottom mask area |
| CENTer | The number of samples in the center mask area |

TOP The number of samples in the top mask area




If the parameter is omitted, the sum of the samples in the mask areas is returned.

**Response Data**

<integer> | N/A

**Example of Use**

```
:FETC:MASK:SAMP:FAIL?
>30
:FETC:MASK:SAMP:FAIL:BOTT?
>0
:FETC:MASK:SAMP:FAIL:CENT?
>20
:FETC:MASK:SAMP:FAIL:TOP?
>10
```

***Notes:***

- To update the result after executing the mask test, use :MEASure:MASK?.

- When the measurement item is not set to the following, the response data is N/A.

  - Amplitude/Time
  - Amplitude/Time&Histogram
  - Histogram
  - OFF

## :FETCh:TIME:MEASurement?

**Function**

This command queries the results obtained by the time measurement function of Scope.

**Syntax**

`:FETCh:TIME:MEASurement?`

**Response Data**

<numeric>,<numeric>,<numeric>,<numeric>,<numeric>,<numeric>......
|N/A

Data are returned as 6 items: jitter p-p, jitter RMS, rise time, fall time, eye width, and DCD.

This command can read 30 values in total from the following six items: current value, mean value, standard deviation, and minimum/maximum values.

The response is output in the following order.

| Item | Unit | Items |
|---|---|---|
| Jitter p-p | ps or UI | Jitter (Peak to peak) |
| Jitter RMS | ps or UI | Jitter (RMS) |
| Rise Time | ps or UI | Rise-time |
| Fall Time | ps or UI | Fall-time |
| Eye Width | ps or UI | Eye width |
| DCD | % | Duty Cycle Distortion |

**Table 3.9.7-2 Data Order**

| Item | Unit | Current value | Mean value | Standard deviation | Minimum value | maximum value |
|---|---|---|---|---|---|---|
| Jitter p-p | ps/UI | 1 | 7 | 13 | 19 | 25 |
| Jitter RMS | ps/UI | 2 | 8 | 14 | 20 | 26 |
| Rise Time | ps/UI | 3 | 9 | 15 | 21 | 27 |
| Fall Time | ps/UI | 4 | 10 | 16 | 22 | 28 |
| Eye Width | ps/UI | 5 | 11 | 17 | 23 | 29 |
| DCD | % | 6 | 12 | 18 | 24 | 30 |

**Example of Use**

```
:FETC:TIME:MEAS?
>66.29,15.03,128.26,133.69,208.61,47.22,68.03,15.33,125.
99,134.01,203.98,47.01,1.99,0.31,2.21,1.86,3.19,0.55,62.
06,14.40,119.36,128.43,194.41,45.36,74.00,16.26,132.62,1
39.59,213.55,48.66
```

***Note:***

The response data is N/A when the measurement item is set to Histogram, Mask, or Off.

## :FETCh:TIME:{JITTer:PPeak|JITTer:RMS|TRISe|FTIMe|EYEWidth|DCD}?

**Function**

This command queries each item for the time measurement of the Scope.

**Syntax**

```
:FETCh:TIME:{JITTer:PPeak|JITTer:RMS|TRISe|FTIMe|EYEWidt
h|DCD}?
```

**Parameter**

JITTer:PPeak | JITTer:RMS | TRISe | FTIMe | EYEWidth | DCD

Specify the measurement item you want to query the result.

| Parameter | Item | Unit |
|---|---|---|
| JITTer:PPeak | Jitter (Peak to Peak) | ps or UI |
| JITTer:RMS | Jitter (RMS) | ps or UI |
| TRISe | Rise Time (Rise time) | ps or UI |
| FTIMe | Fall Time (Fall-time) | ps or UI |
| EYEWidth | Eye Width (Eye width) | ps or UI |
| DCD | Duty Cycle Distortion | % |

**Response Data**

<numeric>,<numeric>,<numeric>,<numeric>,<numeric> | N/A

The measurement results of the specified measurement item are returned in the following order: current value, average, standard deviation, minimum value, and maximum value.

**Example of Use**

```
:FETC:TIME:JITT:PP?
>66.25,65.89,0.98,63.95,68.83
:FETC:TIME:JITT:RMS?
>15.31,15.52,0.26,14.74,16.30
:FETC:TIME:TRIS?
>128.22,130.11,1.52,125.55,134.67
:FETC:TIME:FTIM?
>133.66,129.96,2.59,122.19,137.75
:FETC:TIME:EYEW?
>208.60,206.15,3.32,216.11,196.19
:FETC:TIME:DCD?
>47.2,45.22,1.22,41.56,48.88
```

***Note:***

The response data is N/A when the measurement item is set to Histogram, Mask, or Off.

## 3.9.8 TRACe subsystem

### :TRACe[:DATA]:CHANnelA|CHANnelB|CHANnels?

**Function**

This command queries the trace data of Scope that is in eye pattern mode.

**Syntax**

```
:TRACe[:DATA]:{CHANnelA|CHANnelB|CHANnels}?
```

**Parameter**

:TRACe[:DATA]:CHANnelA

This command requests sending of trace data for Channel A.
Before sending this command, always set Channel A to On and Channel B to Off.

:TRACe[:DATA]:CHANnelB

This command requests sending of trace data for Channel B.
Before sending this command, always set Channel B to On and Channel A to Off.

:TRACe[:DATA]:CHANnels

This command requests sending of trace data for Channel A and Channel B. Set both Channel A and Channel B to On.
When the display of the channel querying the trace data is set to Off, the data cannot be returned.

**Response Data**

- When channel to read not displayed: "Channel Off"
- When reading one channel:

{CHA|CHB}-<integer>(<numeric>,<numeric>)[,(<numeric>,<numeric>)]
…

- When reading one channel:

CHA-<integer>(<numeric>,<numeric>)[,(<numeric>,<numeric>)]…,CHB-<integer>(<numeric>,<numeric>)[,(<numeric>,<numeric>)]…

CHA|CHB

| | |
|---|---|
| CHA | Channel A |
| CHB | Channel B |

<integer>

Trace data score

(<numeric>,<numeric>)

Each time and amplitude

**Example of Use**

To query the trace data for the Channel A:

```
:TRAC:CHANA?
>CHA-2039(86.0,39.97),(86.0,167.13)…
```

To query the trace data for the Channel A and B

```
:TRAC:CHAN?
>CHA-2039(86.0,39.97),(86.0,167.13)…,(285.9,-3.92),CHB-2
039(86.0,152.10)…
```

***Note:***

This message is invalid when Scope is in Pulse or Coherent Eye mode.

Before sending this message, send :TRACe[:DATA]:PREPare to stop updating the Scope screen and make the Scope get ready for receiving trace data query.

If this message is sent after :TRACe[:DATA]:PREPare is sent, the Scope starts data acquisition and creates trace data. Once trace data is created, this message can be used.

To release this status, send :TRACe[:DATA]:END after completing reading of the trace data.

The following changes are made for the MX210000A of Version 3.00 or later.

- This message can be used when Scope is in pulse mode. However, this message cannot be used when in coherent eye.
- Sending of the command :TRACe[:DATA]:PREPare is not required before sending the data.
- Sending of the command :TRACe[:DATA]:END is not required after sending the data.

## :TRACe[:DATA]:END

**Function**

This command terminates the status (readout mode ) set by the :TRACe[:DATA]:PREPare command.

**Syntax**

```
:TRACe[:DATA]:END
```

**Example of Use**

```
:TRAC:END
```

***Note:***

When executing this command, the normal screen-update mode is returned.

Send this command after reading all the trace data after sending the trace data query.

This command is used in combination with the :TRACe[:DATA]:PREPare command.

## :TRACe[:DATA]:PREPare

### Function

This command sets the instrument to the status (readout mode ) in which the Scope trace data can be read via the remote interface.

### Syntax

```
:TRACe[:DATA]:PREPare CHA|CHB|BOTH
```

### Parameter

CHA:

Sets status for reading only Channel A data
Before using this command, Channel A is set to On and Channel B is set to Off.
Send :TRACe[:DATA]:CHANnelA? after this parameter is set.

CHB:

Sets status for reading only Channel B data
Before using this command, Channel B is set to On and Channel A is set to Off.
Send : TRACe[:DATA]:CHANnelB? after this parameter is set.

BOTH:

Sets status for reading Channel A and B data
Before using this command, Both Channel A and B is set to On.
Send : TRACe[:DATA]:CHANnels? after this parameter is set.

### Example of Use

To set the Channel A waveform display to On.

```
:INPut:CHA ON
:INPut:CHB OFF
```

To activate the data reading status of Channel A.

```
:TRACe:PREPare CHA
```

To read out the data of Channel A.

```
:TRACe:CHANnelA?
```

To release the data reading status of Channel A.
`:TRACe:END`

***Note:***

When this command is sent, updating of the Scope waveform screen is stopped.

Send :TRACe[:DATA]:PREPare before sending :TRACe[:DATA]:CHANnelA|CHANnelB|CHANnels? to query the trace data.

The status set by this command is released by :TRACe[:DATA]:END.

# Appendix A Command Compatibility With Existing Products

# A.1 Command Compatibility Tables

✓: Compatible.

*: Partly compatible; occasional errors when sending commands for previous hardware to this instrument.

(Blank): Incompatible; sending commands for previous hardware to this instrument always causes errors.

**Table A.1-1 Command Compatibility (SCPI)**

| Command | MP1800A | MP1632C | MP1776A |
|---|---|---|---|
| :CALCulate:DATA:EALarm | * | | |
| :CALCulate:DATA:MONitor | ✓ | | |
| :CALCulate:OPTical:STATus | * | | |
| :CALibrate:AMPLitude | | | |
| :CALibrate:APPLication | | | |
| :CALibrate:CGain | | | |
| :CALibrate:OEPower | | | |
| :CALibrate:RESPonsivity | | | |
| :CALibrate:SYSTem:CGain | | | |
| :CONFigure:EXRCorrection | | | |
| :CONFigure:MASK:MARGin | | | |
| :CONFigure:MASK:TYPe | | | |
| :CONFigure:MEASure:AMPTIME{1\|2\|3\|4} | | | |
| :CONFigure:MEASure:CHANnel | | | |
| :CONFigure:MEASure:TYPe | | | |
| :DISPlay:RESult:EALarm:HRESet | ✓ | ✓ | ✓ |
| :DISPlay:RESult:EALarm:MODE | ✓ | | |
| :DISPlay:WINDow:GRAPhics:CLEar | | | |
| :DISPlay:WINDow[:SCALe]:AUTOscale | | | |
| :DISPlay:WINDow:X[:SCALe]:BITs | | | |

**Table A.1-1 Command Compatibility (SCPI) (Cont'd)**

| Command | MP1800A | MP1632C | MP1776A |
|---|---|---|---|
| :INPut:BITRate | | | |
| :INPut:BITRate:DIVRate | | | |
| :INPut:BITRate:STANdard | | | |
| :INPut:DATA:ATTFactor | | | |
| :INPut:DATA:INTerface | * | | |
| :INPut:DATA:THReshold | | | |
| :INSTrument:OE:CONDition | | | |
| :INSTrument:OE[:EVENt] | | | |
| :INSTrument:OE:NTRansition | | | |
| :INSTrument:OE:PTRansition | | | |
| :INSTrument:OE:RESet | | | |
| :INSTrument:PE1:CONDition | | | |
| :INSTrument:PE1[:EVENt] | | | |
| :INSTrument:PE1:NTRansition | | | |
| :INSTrument:PE1:PTRansition | | | |
| :INSTrument:PE1:RESet | | | |
| :INSTrument:PE2:CONDition | | | |
| :INSTrument:PE2[:EVENt] | | | |
| :INSTrument:PE2:NTRansition | | | |
| :INSTrument:PE2:PTRansition | | | |
| :INSTrument:PE2:RESet | | | |
| :INSTrument:WAV:CONDition | | | |

Table A.1-1 Command Compatibility (SCPI) (Cont'd)

| Command | MP1800A | MP1632C | MP1776A |
|---|---|---|---|
| :INSTrument:WAV[:EVENt] | | | |
| :INSTrument:WAV:NTRansition | | | |
| :INSTrument:WAV:PTRansition | | | |
| :INSTrument:WAV:RESet | | | |
| :INSTrument:XSFP:CONDition | ✓ | | |
| :INSTrument:XSFP[:EVENt] | ✓ | | |
| :INSTrument:XSFP:NTRansition | ✓ | | |
| :INSTrument:XSFP:PTRansition | ✓ | | |
| :INSTrument:XSFP:RESet | ✓ | | |
| :MEASure:AMPLitude | | | |
| :MEASure:MASK | | | |
| :MEASure:TIME | | | |
| :MODule:ID | ✓ | | |
| :OUTPut:BITRate | | | |
| :OUTPut:BITRate:DIVRate | | | |
| :OUTPut:BITRate:OFFSet | | | |
| :OUTPut:BITRate:STANdard | | | |
| :OUTPut:CLOCk:FREQuency | ✓ | ✓ | |
| :OUTPut:CLOCk:OFFset:PPM | ✓ | | |
| :OUTPut:CLOCk:OPERation | ✓ | | |
| :OUTPut:CMU:EXTClock | ✓ | | |
| :OUTPut:CMU:FREQuency | ✓ | | |
| :OUTPut:CMU:REFClock | ✓ | | |
| :OUTPut:CMU:RESolution | ✓ | | |
| :OUTPut:DATA:AMPLitude | ✓ | ✓ | |
| :OUTPut:DATA:ATTFactor | ✓ | | |
| :OUTPut:DATA:OUTPut | ✓ | | |
| :OUTPut:DATA:RELative | | | |
| :OUTPut:RCLock:SELect | * | * | |
| :OUTPut:SYNC:SOURce | | | |
| [:SENSe]:DISPlay:MODE | | | |
| [:SENSe]:INPut:FILter | | | |
| [:SENSe]:INPut:WAVLength | | | |
| :SENSe:MEASure:ASTate | ✓ | | |
| :SENSe:MEASure:ASTP | ✓ | | |
| :SENSe:MEASure:ASTRt | ✓ | | |

**Table A.1-1 Command Compatibility (SCPI) (Cont'd)**

| Command | MP1800A | MP1632C | MP1776A |
|---|---|---|---|
| :SENSe:MEASure:EALarm:ELAPsed | ✓ | ✓ | ✓ |
| :SENSe:MEASure:EALarm:MODE | ✓ | ✓ | ✓ |
| :SENSe:MEASure:EALarm:PERiod | ✓ | ✓ | ✓ |
| :SENSe:MEASure:EALarm:STARt | ✓ | ✓ | ✓ |
| :SENSe:MEASure:EALarm:STATe | ✓ | ✓ | ✓ |
| :SENSe:MEASure:EALarm:STOP | ✓ | ✓ | ✓ |
| :SENSe:MEASure:EALarm:TIMed | ✓ | ✓ | ✓ |
| :SENSe:MEASure:STARt | ✓ | ✓ | ✓ |
| :SENSe:MEASure:STOP | ✓ | ✓ | ✓ |
| :SENSe:MMEMory:PATTern:RECall | | | |
| :SENSe:PATTern:DATA:LENGth | ✓ | | |
| :SENSe:PATTern:LOGic | ✓ | | |
| :SENSe:PATTern:SYNC:ASYNc | ✓ | ✓ | ✓ |
| :SENSe:PATTern:SYNC:FPOSition | ✓ | | |
| :SENSe:PATTern:SYNC:PSMode | * | * | * |
| :SENSe:PATTern:SYNC:THReshold | ✓ | | ✓ |
| :SENSe:PATTern:TYPE | * | | |
| :SENSe:PARam:AEXECute | | | |
| :SENSe:PARam:TRACking | | | |
| :SENSe:SAMPles:JUDGs | | | |
| [:SENSe]:SAMPling:STATus | | | |
| [:SENSe]:TIME:ACQClock | | | |
| [:SENSe]:TIME:CRKRate | | | |
| [:SENSe]:TIME:DATRate | | | |
| [:SENSe]:TIME:DIVRatio | | | |
| [:SENSe]:TIME:PATLength | | | |
| :SOURce:MMEMory:PATTern:RECall | | | |
| :SOURce:OPTical:SIGNal:OUTPut | ✓ | | |
| :SOURce:OPTical:SIGNal:WLENgth | ✓ | | |
| :SOURce:OPTical:XFP:REFClock | | | |
| :SOURce:OUTPut:ASET | ✓ | | |
| :SOURce:PATTern:DATA:LENGth | ✓ | | |
| :SOURce:PATTern:EADDition:RATE | ✓ | * | |
| :SOURce:PATTern:EADDition:SET | ✓ | ✓ | |
| :SOURce:PATTern:EADDition:SINGle | ✓ | ✓ | |

**Table A.1-1 Command Compatibility (SCPI) (Cont'd)**

| Command | MP1800A | MP1632C | MP1776A |
|---|---|---|---|
| :SOURce:PATTern:EADDition:VARiation | ✓ | | |
| :SOURce:PATTern:LOGic | ✓ | | |
| :SOURce:PATTern:TYPE | * | | |
| :STATus:OPERation:CONDition | ✓ | ✓ | |
| :STATus:OPERation:ENABle | ✓ | ✓ | |
| :STATus:OPERation:NTRansition | ✓ | ✓ | |
| :STATus:OPERation:PTRansition | ✓ | ✓ | |
| :STATus:OPERation[:EVENt] | ✓ | ✓ | |
| :STATus:PRESet | ✓ | ✓ | |
| :SYSTem:BEEPer:SET | | | |
| :SYSTem:DATE | ✓ | | |
| :SYSTem:DISPlay:RESult | ✓ | | |
| :SYSTem:ERRor | ✓ | ✓ | ✓ |
| :SYSTem:ERRor:HCLear | | | |
| :SYSTem:ERRor:HISTory | | | |
| :SYSTem:INFormation:ERRor | ✓ | | |
| :SYSTem:MEMory:INITialize | ✓ | ✓ | ✓ |
| :SYSTem:MMEMory:RECall | | | |
| :SYSTem:MMEMory:STORe | | | |
| :SYSTem:ORGanization:HARDware | | | |
| :SYSTem:PRINt:COPY | ✓ | ✓ | ✓ |
| :SYSTem:TERMination | ✓ | ✓ | |
| :SYSTem:TIME | ✓ | | |
| :SYSTem:VERSion | ✓ | ✓ | ✓ |
| :TRACe[:DATA]:CHANnelA \| CHANnelB \| CHANnels | | | |
| :TRACe[:DATA]:END | | | |
| :TRACe[:DATA]:PREPare | | | |
| :TRACe:PREamble | | | |

Table A.1-2 Command Compatibility (Native)

| Command | MP1800A | MP1632C | MP1776A |
|---|---|---|---|
| TRM | ✓ | ✓ | |
| RTM | ✓ | | |
| INF | ✓ | | |
| INI | ✓ | ✓ | ✓ |
| HCP | ✓ | ✓ | ✓ |
| OON | ✓ | | |
| SAT | ✓ | | |
| SOT | ✓ | | |
| STT | ✓ | | |
| RFC | * | * | |
| CRE | * | * | |
| CEC | ✓ | | |
| SOP | | | |
| OPE | | | |
| CRF | | | |
| CRS | ✓ | | |
| COP | | | |
| PTS | * | | |
| LGC | ✓ | | |
| DLN | ✓ | | |
| DON | ✓ | | |
| DAP | ✓ | ✓ | |
| DAT | ✓ | | |
| PRO | | | |
| EAD | ✓ | ✓ | |
| EAV | ✓ | | |
| ESI | ✓ | ✓ | |
| ERT | ✓ | * | |
| DSD | * | | |
| DTH | | | |

Table A.1-2 Command Compatibility (Native) (Cont'd)

| Command | MP1800A | MP1632C | MP1776A |
|---|---|---|---|
| SYE | ✓ | | ✓ |
| SYM | * | * | * |
| SYN | ✓ | ✓ | ✓ |
| FPS | ✓ | | |
| HRE | ✓ | ✓ | ✓ |
| MTR | ✓ | | |
| ERS | ✓ | ✓ | |
| END | * | | |
| ER | ✓ | ✓ | |
| EC | ✓ | ✓ | |
| CC | ✓ | ✓ | |
| FRQ | ✓ | ✓ | |
| MOD | ✓ | ✓ | ✓ |
| PRD | ✓ | ✓ | ✓ |
| CUR | ✓ | | |
| STA | ✓ | ✓ | ✓ |
| STO | ✓ | ✓ | ✓ |
| MSR | ✓ | ✓ | ✓ |
| MSA | ✓ | ✓ | ✓ |
| MSO | ✓ | ✓ | ✓ |
| MLP | ✓ | ✓ | ✓ |
| ETI | ✓ | ✓ | ✓ |

## A.2 Command Compatibility Between MP2100A and MP2100B

If one of the following bit rate setting commands is used for PPG Ch3 or PPG Ch4 of MP2100B, an error occurs:

:OUTPut:BITRate:STANdard
:OUTPut:CLOCk:OPERation
:OUTPut:BITRate
:OUTPut:CLOCk:FREQuency
:OUTPut:CMU:FREQuency
:OUTPut:BITRate:OFFSet
:OUTPut:CLOCk:OFFset:PPM

If one of the following ED bit rate setting commands is used for MP2100B, an error occurs:

:INPut:BITRate:STANdard
:INPut:BITRate

The following changes are made to the Reference CLK setting commands for MP2100B:

Addition:
:OUTPut:RCLock
Changed into compatible commands:
:OUTPut:CMU:REFClock
:OUTPut:RCLock:SELect

For MP2100B, Ch Tracking and Ch2 Reference CLK are set to On and Ch1, by default. Therefore, if your BERTWave is MP2100A, execution of a sequence results in an error, in case where the following items for Ch1 and Ch2 are set to different values, respectively.

- Bitrate
- Test Pattern
- Gating Cycle/Period

# A.3 Introduction of Native Commands

### Native Command Usage Purpose

The purpose of the Native command is to allow use of remote interface software for previous Anritsu pulse pattern generator and error detector models with the BERTWave.

### Native Command Format

The Native command is configured by the string of ASCII code.
For the Native command format, refer to 2.5 "Message Format".
The word string that indicates the Native command end is LF (line feed) or CR (carriage return) +LF.

***Note:***

Native commands must end with LF or CR+LF; otherwise, a timeout error results because the communication is not able to end.

### Channel Setting Method

When using the pulse pattern generator/ error detector with two or more channels, the channel to be controlled is set using :MODule:ID command before executing the Native command.

### Example of Use

When controlling the PPG/ED_1ch:

```
:MOD:ID 1
```

When controlling the PPG/ED_2ch:

```
:MOD:ID 2
```

When querying the currently controlled channel:

```
:MOD:ID?
```

# A.4 Compatible Commands and Descriptions

## TRM

### Function

This command has the same function as :SYSTem:TERMination.

### Syntax

```
TRM 0|1
TRM?
```

### Parameter

| | |
|---|---|
| 0 | LF+EOI (default) |
| 1 | CR+LF+EOI |

### Response Data

TRM 0|1

### Example of Use

```
TRM 0
TRM?
>TRM 0
```

## RTM?

### Function

This command has the same function as :SYSTem:{DATE|TIME}?.

### Response Data

RTM <year>,<month>,<day>,<hour>,<minute>,<second>

Each of the parameters in the response data is a two-digit number. The lower two digits of the year is output as <year>.

### Example of Use

```
RTM?
>RTM 09,10,24,09,51,13
```

## INF?

### Function

This command has the same function as :SYSTem:INFormation:ERRor?.

### Response Data

INF <error_code>

The meanings of <error_code> are the same as those of :SYSTem:INFormation:ERRor?.

### Example of Use

```
INF?
>INF 1
```

## :SYSTem:MEMory:INITialize

### Function

This command has the same function as *RST.

### Example of Use

```
:SYST:MEM:INIT
```

## INI

### Function

This command has the same function as :SYSTem:MEMory:INITialize.

### Example of Use

```
INI
```

## HCP

### Function

This command has the same function as :SYSTem:PRINt:COPY.

### Example of Use

```
HCP
```

## OON

### Function

This command sets/queries the On/Off state of signal outputting from all PPG channels and optical outputting from XFP/SFP+.

### Syntax

```
OON 0|1
OON?
```

### Parameter

| | |
|---|---|
| 0 | PPG output and optical output OFF |
| 1 | PPG output and optical output ON |

### Response Data

OON 0|1

**Example of Use**

```
OON 1
OON?
>OON 1
```

## :SENSe:MEASure:ASTRt

**Function**

This command has the same function as*TRG.

**Example of Use**

```
:SENS:MEAS:ASTR
```

## SAT

**Function**

This command has the same function as :SENSe:MEASure:ASTRt.

**Example of Use**

```
SAT
```

## SOT

**Function**

This command has the same function as :SENSe:MEASure:ASTP.

**Example of Use**

```
SOT
```

## STT?

**Function**

This command has the same function as :SENSe:MEASure:ASTate?.

**Response Data**

STT 0|1

**Example of Use**

```
STT?
>STT 1
```

## :OUTPut:CMU:REFClock

**Function**

This command has the same function as :OUTPut:RCLock:SELect.

**Syntax**

```
:OUTPut:CMU:REFClock
INTernal|EXTernal|CH1External|CH2External|SYNChronize
:OUTPut:CMU:REFClock?
```

**Example of Use**

```
:OUTP:CMU:REFC INT
:OUTP:CMU:REFC?
>INT
```

## RFC

**Function**

This command has the same function as :OUTPut:RCLock:SELect.

**Syntax**

```
RFC 0|1|2|3|4
RFC?
```

**Parameter**

| | |
|---|---|
| 0 | INTernal |
| 1 | EXTernal |
| 2 | *Added in Version 4:* CH1External |
| 3 | *Added in Version 4:* CH2External |
| 4 | *Added in Version 4:* SYNChronize |

**Response Data**

RFC 0|1|2|3|4

**Example of Use**

```
RFC 0
RFC?
>RFC 0
```

## CRE

**Function**

This command has the same function as RFC.

**Syntax**

```
CRE 0|1|2|3|4
CRE?
```

**Example of Use**

```
CRE 0
CRE?
>CRE 0
```

## CEC

**Function**

This command has the same function as :OUTPut:CMU:EXTClock.

**Syntax**

```
CEC 0|1
CEC?
```

**Parameter**

0 Ext 10MHz In

1 Ext 1/16 In

**Response Data**

CEC 0|1

**Example of Use**

```
CEC 1
CEC?
>CEC 1
```

## SOP

**Function**

This command has the same function as :OUTPut:SYNC:SOURce.

**Syntax**

```
SOP <integer>
SOP?
```

**Parameter**

<integer>

| | |
|---|---|
| 0 | PPG1_1/1Clk |
| 1 | PPG1_1/2Clk |
| 2 | PPG1_1/4Clk |
| 3 | PPG1_1/8Clk |
| 4 | PPG1_1/16Clk |
| 5 | PPG1_1/64Clk |
| 6 | PPG2_1/1Clk |
| 7 | PPG2_1/2Clk |
| 8 | PPG2_1/4Clk |
| 9 | PPG2_1/8Clk |
| 10 | PPG2_1/16Clk |
| 11 | PPG2_1/64Clk |
| 13 | ED1_1/16Clk |
| 15 | ED2_1/16Clk |
| 16 | PPG1_Pattern Sync |
| 17 | PPG2_Pattern Sync |
| 19 | ED1_1/4Clk |
| 20 | ED1_1/8Clk |
| 22 | ED2_1/4Clk |
| 23 | ED2_1/8Clk |
| 24 | ED3_1/4Clk |
| 25 | ED3_1/8Clk |
| 26 | ED4_1/4Clk |
| 27 | ED4_1/8Clk |
| 28 | ED4_1/16Clk |
| 29 | PPG3_Pattern Sync |
| 30 | PPG4_Pattern Sync |

**Response Data**

SOP <integer>

**Example of Use**

To set the 1/16 divide clock synchronized with the data output of the PPG Channel 1 to the output signal to the Sync Output connector.

```
SOP 4
SOP?
>SOP 4
```

## :SENSe:PARam:AEXecute

### Function

This command is compatible with version 4 or later, and performs batch setting of PPG/ED of Ch1 and Ch2.

### Syntax

```
:SENSe:PARam:AEXecute
<ppg1>,<ppg2>,<ed1>,<ed2>,<bitrate_standard>,<bitrate>,<
ppm>,<pattern>,<amplitude>
```

### Parameter

<ppg1>,<ppg2>,<ed1>,<ed2>

Specify the object device(s) to be set by the command.

Specify 1 for the object device(s) to be set by the command; specify 0 otherwise.

<bitrate_standard> = <string>

Select a standard for the bit rate of PPG/ED. The parameter values are the same as for :{OUTPut|INPut}:BITRate:STANdard.

<bitrate> = <integer>

Specify/query the bit rate (kbit/s) if "Variable" is selected as the bit rate standard for PPG/ED. The parameter values are the same as for :{OUTPut|INPut}:BITRate.

<ppm> = <integer>

Specify/query the bit rate offset (ppm) for PPG. The parameter values are the same as for :OUTPut:BITRate:OFFSet.

<pattern> = PRBS{7|9|15|23|31}|USER

Specify the test pattern for PPG/ED. The parameter values are the same as for :{SOURce|SENSe}:PATTern:TYPE.

<amplitude> = <numeric>

Specify the signal amplitude (Vpp) of PPG. The parameter values are the same as for :OUTPut:DATA:AMPLitude.

### Example of Use

To set PPG/ED of Ch1/2 to the bit rate of 10.3125 Gbit/s, offset of 100 ppm, test pattern of PRBS 2^31–1, and amplitude of 0.8 Vpp:

```
:SENS:PAR:AEX 1,1,1,1,"VARIABLE",10312500,100,PRBS31,0.80
```

## :OUTPut:CLOCk:OPERation

### Function

This command has the same function as :OUTPut:BITRate:STANdard.

Appendix

Appendix A

**Syntax**

```
:OUTPut:CLOCk:OPERation <bitrate_standard>
:OUTPut:CLOCk:OPERation?
```

**Parameter**

<bitrate_standard> = <string>

The parameter values are the same as for :OUTPut:BITRate:STANdard.

**Example of Use**

```
:OUTP:CLOC:OPER "10G_LAN"
:OUTP:CLOC:OPER?
>"10G_LAN"
```

## OPE

**Function**

This command has the same function as :OUTPut:BITRate:STANdard.

**Syntax**

```
OPE <bitrate_standard>
OPE?
```

**Parameter**

<bitrate_standard> = <integer>

**Table A.4-1 Bit Rate Standards**

| <integer> | Standard | Bit Rate (bit/s) | Remarks |
|---|---|---|---|
| 0 | Variable-1/1 | 6.25G to 12.5G | |
| 1 | 1GFC | 1.0625G | *1 |
| 2 | 2GFC | 2.125G | *1 |
| 3 | 4GFC | 4.25G | |
| 4 | 8GFC | 8.5G | |
| 5 | 10GFC | 10.518G | |
| 6 | 10GFC FEC | 11.3168G | |
| 7 | 1GbE | 1.25G | *1 |
| 8 | 2GbE | 2.5G | *1 |
| 9 | Infiniband | 2.5G | *1 |
| 10 | 10GbE WAN | 9.95328G | |
| 11 | 10GbE LAN/PHY | 10.3125G | |
| 12 | 10GbE OTU1e | 11.049G | |
| 13 | 10GbE OTU2e | 11.095G | |

*1: If the Option 090 is not installed, the bit rate that is equal to or less than Variable-1/4 cannot be selected for the ED.

Table A.4-1 Bit Rate Standards (Cont'd)

| <integer> | Standard | Bit Rate | Remarks |
|---|---|---|---|
| 14 | OC-3/STM-1 | 155.22M | *1 |
| 15 | OC-12/STM-4 | 622.08M | *1 |
| 16 | OC-48/STM16 | 2.488G | *1 |
| 17 | OTU-1 | 2.666057G | *1 |
| 18 | OC-192/STM-64 | 9.95328G | |
| 19 | G.975 FEC | 10.664G | |
| 21 | Variable-1/2 | 6.25G to 3.125G | |
| 22 | Variable-1/4 | 3.125G to 1.5625G | *1 |
| 23 | Variable-1/8 | 1.5625G to 781.25M | *1 |
| 24 | Variable-1/16 | 781.25M to 390.625M | *1 |
| 25 | Variable-1/32 | 390.625M to 195.312M | *1 |
| 26 | Variable-1/64 | 195.312M to 125M | *1 |
| 27 | Infiniband x2 | 5G | |
| 28 | Infiniband x4 | 10G | |
| 29 | OC-24 | 1.244G | *1 |
| 30 | CPRI | 614.4M | *1 |
| 31 | CPRI x2 | 1.2288G | *1 |
| 32 | CPRI x4 | 2.4576G | *1 |
| 33 | CPRI x5 | 3.072G | *1, *2 |
| 34 | CPRI x10 | 6.144G | *2 |
| 35 | OBSAI RP3 | 768M | *1, *2 |
| 36 | OBSAI RP3 x2 | 1.536G | *1, *2 |
| 37 | OBSAI RP3 x4 | 3.072G | *1, *2 |
| 38 | OBSAI RP3 x8 | 6.144G | *2 |

*2: If the Option 090/092 is not installed, the bit rate cannot be selected because the bit rate range is limited.

#### Response Data

OPE <bitrate_standard> = <integer>

#### Example of Use

```
OPE 3
OPE?
>OPE 3
```

### :OUTPut:CLOCk:FREQuency

#### Function

This command has the same function as :OUTPut:BITRate.

#### Syntax

```
:OUTPut:CLOCk:FREQuency <bitrate>
:OUTPut:CLOCk:FREQuency?
```

#### Example of Use

```
:OUTP:CLOC:FREQ 8500000
:OUTP:CLOC:FREQ?
>8500000
```

### :OUTPut:CMU:FREQuency

#### Function

This command has the same function as :OUTPut:BITRate.

#### Syntax

```
:OUTPut:CMU:FREQuency <bitrate>
:OUTPut:CMU:FREQuency?
```

#### Example of Use

```
:OUTP:CMU:FREQ 8500000
:OUTP:CMU:FREQ?
>8500000
```

### CRF

#### Function

This command has the same function as :OUTPut:BITRate.

#### Syntax

```
CRF <bitrate>
CRF?
```

#### Response Data

CRF <bitrate>

#### Example of Use

```
CRF 8500000
CRF?
>CRF 8500000
```

### CRS

#### Function

This command has the same function as :OUTPut:CMU:RESolution.

**Syntax**

```
CRS 0|1
CRS?
```

**Parameter**

0 KHz

1 MHz

**Response Data**

CRS 0|1

**Example of Use**

```
CRS 0
CRS?
>CRS 0
```

## :OUTPut:CLOCk:OFFSet:PPM

**Function**

This command has the same function as :OUTPut:BITRate:OFFSet.

**Syntax**

```
:OUTPut:CLOCk:OFFSet:PPM <numeric>
:OUTPut:CLOCk:OFFSet:PPM?
```

**Example of Use**

```
:OUTP:CLOC:OFFS:PPM 100
:OUTP:CLOC:OFFS:PPM?
>100
```

## COP

**Function**

This command has the same function as :OUTPut:BITRate:OFFSet.

**Syntax**

```
COP <numeric>
COP?
```

**Response Data**

COP <integer>

**Example of Use**

```
COP 100
COP?
>COP 100
```

***Note:***

If Ch3 or Ch4 has been specified by MODule:ID, execution of the command results in an error (–220 Parameter error), and execution of the query returns the current setting (because the bit rate of Ch3/4 is dependent on Ch1/2).

## PTS

**Function**

This command has the same function as :SOURce:PATTern:TYPE and :SENSe:PATTern:TYPE.

**Syntax**

```
PTS <ppg_ed>,<pattern>
PTS? <ppg_ed>
```

**Parameter**

<ppg_ed>

| | |
|---|---|
| 0 | PPG |
| 1 | ED |

<pattern>

| | |
|---|---|
| 0 | PRBS 2^7–1 |
| 1 | PRBS 2^9–1 |
| 2 | PRBS 2^15–1 |
| 3 | PRBS 2^23–1 |
| 4 | PRBS 2^31–1 |
| 5 | Programmable Pattern |

**Response Data**

PTS <pattern>

**Example of Use**

To set the ED test pattern to PRBS 2^15–1:

```
PTS 1,2
PTS? 1
>PTS 2
```

## LGC

### Function

This command has the same function as :SOURce:PATTern:LOGic and :SENSe:PATTern:LOGic.

### Syntax

```
LGC <ppg_ed>,<pos_neg>
LGC? <ppg_ed>
```

### Parameter

<ppg_ed>

0 PPG

1 ED

<pos_neg>

0 Positive

1 Negative

### Response Data

LGC <pos_neg>

### Example of Use

To set the test pattern logic of the PPG to the negative logic:

```
LGC 0,1
LGC? 0
>LGC 1
```

## DLN?

### Function

This command has the same function as :SOURce:PATTern:DATA:LENGth? and :SENSe:PATTern:DATA:LENGth?.

### Syntax

```
DLN? <ppg_ed>
```

### Parameter

<ppg_ed>

0 PPG

1 ED

### Response Data

DLN <integer>

#### Example of Use

```
DLN? 1
>16384
```

## DON

#### Function

This command has the same function as :OUTPut:DATA:OUTPut.

#### Syntax

```
DON 0|1
DON?
```

#### Parameter

| | |
|---|---|
| 0 | Off |
| 1 | On |

#### Response Data

DON 0|1

#### Example of Use

```
DON 1
DON?
>DON 1
```

## DAP

#### Function

This command has the same function as :OUTPut:DATA:AMPLitude.

#### Syntax

```
DAP <numeric>
DAP?
```

#### Parameter

The parameter values are the same as for :OUTPut:DATA:AMPLitude.

#### Response Data

DAP <numeric>

#### Example of Use

```
DAP 0.5
DAP?
>DAP 0.5
```

## DAT

### Function

This command has the same function as :OUTPut:DATA:ATTFactor.

### Syntax

```
DAT <numeric>
DAT?
```

### Parameter

The parameter values are the same as for :OUTPut:DATA:ATTFactor.

### Response Data

DAT <integer>

### Example of Use

```
DAT 20
DAT?
>DAT 20
```

## PRO?

### Function

This command has the same function as :OUTPut:DATA:RELative?.

### Response Data

PRO <numeric>

### Example of Use

```
PRO?
>PRO 0.4
```

## EAD

### Function

This command has the same function as :SOURce:PATTern:EADDition:SET.

### Syntax

```
EAD 0|1|7
EAD?
```

#### Parameter

0 Does not generate error
1 Generates error
7 Sets to generate single error

#### Response Data

EAD 0|1

#### Example of Use

```
EAD 1
EAD?
>EAD 1
```

### EAV

#### Function

This command has the same function as :SOURce:PATTern:EADDition:VARiation.

#### Syntax

```
EAV 0|1
EAV?
```

#### Parameter

0 Repeat
1 Single

#### Response Data

EAV 0|1

#### Example of Use

```
EAV 0
EAV?
>EAV 0
```

### ESI

#### Function

This command has the same function as :SOURce:PATTern:EADDition:SINGle.

#### Example of Use

```
ESI
```

## ERT

### Function

This command has the same function
as :SOURce:PATTern:EADDition:RATE.

### Syntax

```
ERT 1,<integer>
ERT?
```

### Parameter

1: Indicates a fixed-point part of the error addition rate is 1.
<integer>
Exponent part of error addition rate
Range 2 to 12

### Response Data

ERT 1,<integer>
<integer>
2 to 12

### Example of Use

To set the error addition rate to 1E–9:

```
ERT 1,9
ERT?
>ERT 1,9
```

## DSD

### Function

This command has the same function as :INPut:DATA:INTerface.

### Syntax

```
DSD 0|1|2|3
DSD?
```

### Parameter

0 Electrical Single-Ended Data
1 Electrical Single-Ended XData
2 Differential 50 Ohm
3 Optical

### Response Data

DSD 0|1|2|3

**Example of Use**

```
DSD 0
DSD?
>DSD 0
```

## DTH

**Function**

This command has the same function as :INPut:DATA:THReshold.

**Syntax**

```
DTH <numeric>
DTH?
```

**Parameter**

The parameter values are the same as for :INPut:DATA:THReshold.

**Response Data**

DTH <integer>

**Example of Use**

```
DTH -85
DTH?
>DTH -85
```

## SYN

**Function**

This command has the same function as :SENSe:PATTern:SYNC:ASYNc.

**Syntax**

```
SYN 0|1
SYN?
```

**Response Data**

SYN 0|1

**Example of Use**

```
SYN 1
SYN?
>SYN 1
```

## SYE

### Function

This command has the same function as :SENSe:PATTern:SYNC:THReshold.

### Syntax

```
SYE <integer>
SYE?
```

### Parameter

<integer>

| | |
|---|---|
| 0 | 1E–2 |
| 1 | 1E–3 |
| 2 | 1E–4 |
| 3 | 1E–5 |
| 4 | 1E–6 |
| 5 | 1E–7 |
| 6 | 1E–8 |
| 8 | Internal |

### Response Data

SYE 0|1|2|3|4|5|6|8

### Example of Use

```
SYE 0
SYE?
>SYE 0
```

## SYM

### Function

This command has the same function as :SENSe:PATTern:SYNC:PSMode.

### Syntax

```
SYM 0|1
SYM?
```

### Parameter

| | |
|---|---|
| 0 | SYNC Control Off |
| 1 | SYNC Control On |

### Response Data

SYM 0|1

**Example of Use**

```
SYM 0
SYM?
>SYM 0
```

## FPS

**Function**

This command has the same function as :SENSe:PATTern:SYNC:FPOSition.

**Syntax**

```
FPS <numeric>
FPS?
```

**Parameter**

<numeric>

**Response Data**

<integer>

**Example of Use**

```
FPS 1
FPS?
>FPS 1
```

## HRE

**Function**

This command has the same function as :DISPlay:RESult:EALarm:HRESet.

**Example of Use**

```
HRE
```

## MTR?

**Function**

This command has the same function as :CALCulate:DATA:MONitor?.

**Syntax**

```
MTR? 0|3|4
```

**Parameter**

| | |
|---|---|
| 0 | Bit Error |
| 3 | CR Unlock |
| 4 | SYNC Loss |

**Response Data**

MTR 0|1

| | |
|---|---|
| 0 | When an alarm occurs |
| 1 | When no alarm occurs |

**Example of Use**

```
MTR? 0
>MTR 0
```

## ERS?

**Function**

This command is compatible with existing products and queries if a bit error has occurred at ED.

**Response Data**

ERS 0|1

| | |
|---|---|
| 0 | Indicates a bit error has not occurred. |
| 1 | Indicates a bit error has occurred. |

**Example of Use**

```
ERS?
>ERS 0
```

## :SENSe:PARam:AEXecute?

**Function**

This command is compatible with version 4 or later, and performs BER measurement of 10 ms duration at Ch1 and Ch2.

**Syntax**

```
:SENSe:PARam:AEXecute?
```

**Response Data**

<ed1_er>,<ed1_ec>,<ed1_unlock>,<ed1_sync>,<ed2_er>,<ed2_ec>,<ed2_unlock>,<ed2_sync>

<ed1_er>/<ed2_er>

Total bit error rate

Range: 0.0001E–18 to 1.0000E–00

(Form2)

<ed1_ec>/<ed2_ec>

Total bit error count

Range: 0 to 9999999 or 1.0000E+07 to 9.9999E+17

(Form1)

<ed1_unlock>/<ed2_unlock>

CR Unlock status

Range: "Not Occur" or "Occur"

<ed1_sync>/<ed2_sync>

SYNC Loss status

Range: "Not Occur" or "Occur"

**Example of Use**

To stop the re-plotting processing, and query the measurement results with a gating time of 10 ms:

```
:SYST:DISP:RES OFF
:SENS:PAR:AEX?
>"0.0000E-07","0","Not Occur","Not
Occur","1.0000E-02","850007","Not Occur","Not Occur"
```

***Note:***

If the measurement is performed with a gating time of 10 ms under the following conditions, an error occurs:

- When the re-plotting processing for the Scope screen is on
  Turn off the re-plotting processing for the Scope screen, and then send this command.
- When BER measurement is in progress
  Before executing this command, make sure that the measurement has completed, by using :SENSe:MEASure:EALarm:STATe? for example.
- When the user pattern is being recalled
  If one of the following pattern file recall commands is executed, read the status register to make sure the recall processing is completed.

:SOURce:MMEMory:PATTern:RECall
:SENSe:MMEMory:PATTern:RECall

## END?

### Function

This command has the same function as :CALCulate:DATA:EALarm?.

### Syntax

```
END? {0|1|2},{1|2|3|4}
```

### Parameter

Specify the measurement data item to query, combining the first and second parameters.
Though CURRent or LAST can be specified for :CALCulate:DATA:EALarm?, this command queries only the results of the last measurement performed as specified by Gating Time.

0,1 Measurement Start Time
0,2 Measurement End Time
0,3 Measurement elapsed time
0,4 Measurement remaining time
1,3 SYNC Loss Seconds
1,4 CR Unlock Seconds
2,1 Bit Error Rate Total
2,2 Bit Error Count Total
2,3 Clock Count Total
2,6 Bit Error Rate Insertion
2,7 Bit Error Rate Omission
2,8 Bit Error Count Insertion
2,9 Bit Error Count Omission

### Response Data

END <result>
Count, Seconds: 0 to 9999999 or 1.0000E+07 to 9.9999E+17 (Form1)
Rate: 0.0000E–18 to 1.0000E–00 (Form2)
"-------" if there is no data to return.

Start/end time:XX-XX-XX XX:XX:XX format
Elapsed/remaining time: X XX:XX:XX format
"ERR" if there is no data to return.

#### Example of Use

To query the measurement elapsed time:

```
END? 0,3
>END 0 00:00:25
```

To query the error rate:

```
END? 2,1
>END 1.0253E-06
```

## ER?

#### Function

This command is partially compatible with :CALCulate:DATA:EALarm? and queries the bit error rate measured by the ED.

#### Response Data

ER <form2>|-------

<form2>: 0.0000E–18 to 1.0000E–00

-------: If there is no data.

#### Example of Use

```
ER?
>ER 3.8938E-02
```

## EC?

#### Function

This command is partially compatible with :CALCulate:DATA:EALarm? and queries the bit error count measured by the ED.

#### Response Data

EC <form1>|-------

<form1>: 0 to 9999999 or 1.0000E+07 to 9.9999E+17

-------: If there is no data.

#### Example of Use

```
EC?
>EC 1.9469E+08
```

## CC?

#### Function

This command is partially compatible with :CALCulate:DATA:EALarm? and queries the clock count measured by the ED.

#### Response Data

CC <form1>|-------

<form1>: 0 to 9999999 or 1.0000E+07 to 9.9999E+17

-------: If there is no data.

#### Example of Use

```
CC?
>CC 1.0000E-09
```

### FRQ?

#### Function

This command is partially compatible with :CALCulate:DATA:EALarm? and queries the clock frequency measured by the ED.

#### Response Data

FRQ <integer>

#### Example of Use

```
FRQ?
>FRQ 8500000
```

### MOD

#### Function

This command has the same function as :SENSe:MEASure:EALarm:MODE.

#### Syntax

```
MOD 0|1|2
MOD?
```

#### Parameter

| | |
|---|---|
| 0 | Repeat |
| 1 | Single |
| 2 | Untimed |

#### Response Data

MOD 0|1|2

#### Example of Use

To set the gating cycle of ED to Repeat:

```
MOD 0
MOD?
>MOD 0
```

## PRD

### Function

This command has the same function
as :SENSe:MEASure:EALarm:PERiod.

### Syntax

```
PRD <days>,<hours>,<minutes>,<seconds>
PRD?
```

### Parameter

<days>,<hours>,<minutes>,<seconds>

### Response Data

PRD <days>,<hours>,<minutes>,<seconds>
Each of the parameters in the response data is a two-digit number.

### Example of Use

To set the gating time to 1 hour:

```
PRD 0,1,0,0
PRD?
>PRD 00,01,00,00
```

## CUR

### Function

This command has the same function
as :DISPlay:RESult:EALarm:MODE.

### Syntax

```
CUR 0|1
CUR?
```

### Parameter

0 Off
1 On

### Response Data

CUR 0|1

### Example of Use

```
CUR 1
CUR?
>CUR 1
```

## STA

### Function
This command has the same function as :SENSe:MEASure:STARt.

## STO

### Function
This command has the same function as :SENSe:MEASure:STOP.

## MSR?

### Function
This command has the same function as :SENSe:MEASure:EALarm:STATe?.

### Response Data
MSR 0|1

0 Measurement stops

1 During measurement

### Example of Use
```
MSR?
>MSR 1
```

## MSA?

### Function
This command has the same function as :SENSe:MEASure:EALarm:STARt?.

### Response Data
MSA <year>,<month>,<day>,<hour>,<minute>,<second>

0000,00,00,00,00,00 is returned if there is no time data to return.

### Example of Use
```
MSA?
>MSA 2009,10,05,16,25,40
```

## MSO?

### Function

This command has the same function as :SENSe:MEASure:EALarm:STOP?.

### Response Data

MSO <year>,<month>,<day>,<hour>,<minute>,<second>
0000,00,00,00,00,00 is returned if there is no time data to return.

### Example of Use

```
MSO?
>MSO 2009,10,05,16,25,40
MSO?
>MSO 0000,00,00,00,00,00
```

## MLP?

### Function

This command has the same function as :SENSe:MEASure:EALarm:ELAPsed?.

### Response Data

MLP <days>,<hours>,<minutes>,<seconds>

### Example of Use

```
MLP?
>MLP 00,00,02,10
```

## ETI?

### Function

This command has the same function as :SENSe:MEASure:EALarm:TIMed?.

### Response Data

ETI <days>,<hours>,<minutes>,<seconds>

### Example of Use

```
ETI?
>ETI 00,00,02,10
```

# Appendix B Error Codes

This appendix explains the code and message responses to the `SYSTem:ERRor?` query command.

- Command error
- Execution error
- Device unique error

When these errors occur, the standard event status register bit becomes 1. A service request can be generated when an error occurs depending on the setting of the standard event status enable register bit.

When an error occurs, the standard event status register bit that becomes 1 is listed in the table below.





**Table -1 Relationship between Error Number and Standard Event Register**

| Error Code | Message | Error Name | Standard Event Register Bit |
|---|---|---|---|
| –113 | Undefined header | Command error | 5 |
| –220 | Parameter error | Execution error | 4 |
| –310 | System error | Device Dependent error | 3 |

Command error

The corresponding bit is set to 1 if the received program message is undefined, does not conform to the syntax or has a misspelling. Bit 5 of the standard event status register is set when the following errors occur. The errors are generated when the following events occur.

- When sending message not in conformance with syntax described in section 2.5 "Message Format"
  Example: At typographical error in header
  Header includes 2-byte character

- When sending message not in conformance with Common Commands or Device Unique Commands described in Chapter 3 "Message Details" and Appendix A "Command Compatibility With Existing Products"

- When sending a module command, without specifying the module with :Module:ID

- When sending a message not described in Chapter 3
  Examples:
  - The message contains a typing error(s).
  - The message contains a double-byte character(s).
  - The space that should separate the message and parameter is missing.
  - The semicolon that should separate messages is missing.
  - The upper-case (mandatory) character(s) in the message has been omitted.

- When the number of parameters in the message is incorrect
  Examples:
  - Two parameters have been added to the message requiring one parameter when sending the message.
  - The comma(s) that should separate parameters is missing.

- When the parameter format is incorrect
  Examples:
  - Character-string data has been specified in the message requiring a numeric value as a parameter, when sending the message.
  - Character-string data is not enclosed by quotation marks.

### Execution error

Bit 4 of the standard event status register is set when the following errors occur. The errors are generated when the following events occur.

- When header continuation parameter value out of setting range
  Example: When 850000 set when bit rate setting range is 8500000 to 11320000

- When message cannot be executed in current equipment status
  Example: When sending message for setting Scope to instrument without Scope function

### Device Dependent error

Bit 3 of the standard event status register is set when an error occurs internally.

# Appendix C Sample Program

This chapter explains examples of sample programs and how to execute them.

# C.1 Executing Sample Programs

## C.1.1 Setting Sample Program Operating Environment

The sample program operating environment is as follows.

PC

OS: Windows XP Professional Service Pack 2

VISA: NI-VISA Version 4.6

Program tool: Microsoft Visual Visual C# 2008

MP2100A BERTWave

GPIB Address: 1

IP Address: 198.168.12.10

Subnet Mask: 255.255.255.0

Port Number: 5001

Installing NI-VISA

To use VISA at Visual C# 2008, add the following function at installation.

- Development Support .NET Framework 3.5 Language Support
- NI Measurement & Automation Explore –.NET Framework 3.5 Language Support

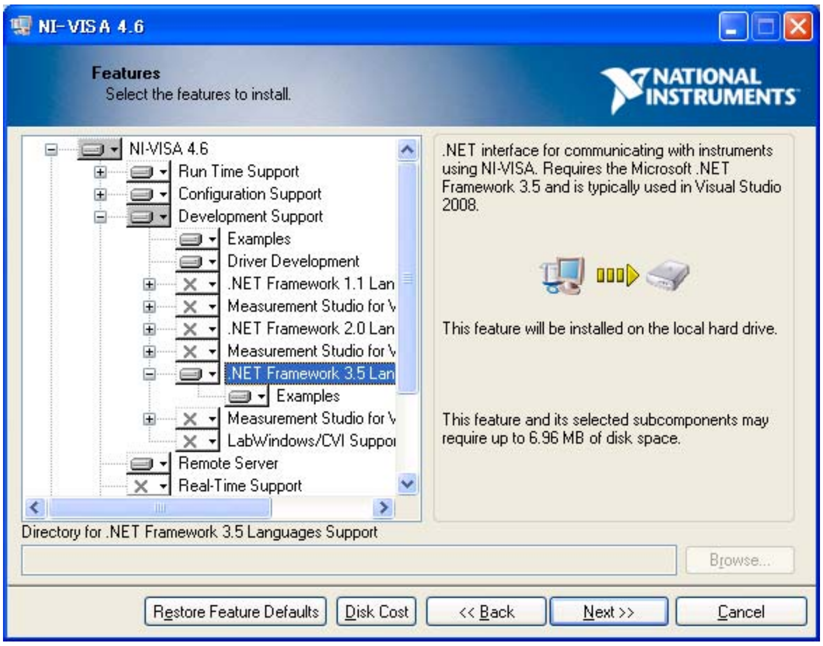



**Figure C.1.1-1 Function Selection Screen at VISA Install**

Setting Visual C# 2008

To use VISA at Visual C# 2008, operate as follows.

1. Click [Add Reference] at the Project menu.
2. Click the .NET tab in the Add Reference dialog box.
3. Select National Instruments Common and National Instruments VisaNS, and click OK.

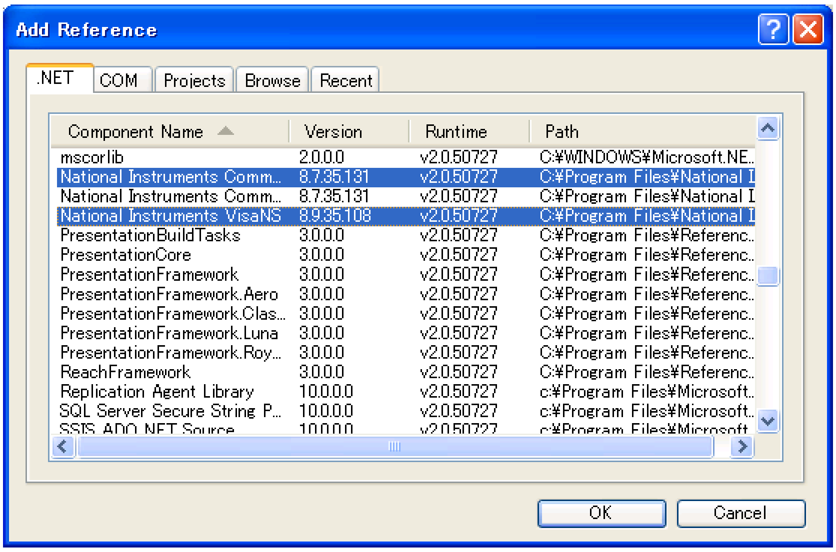
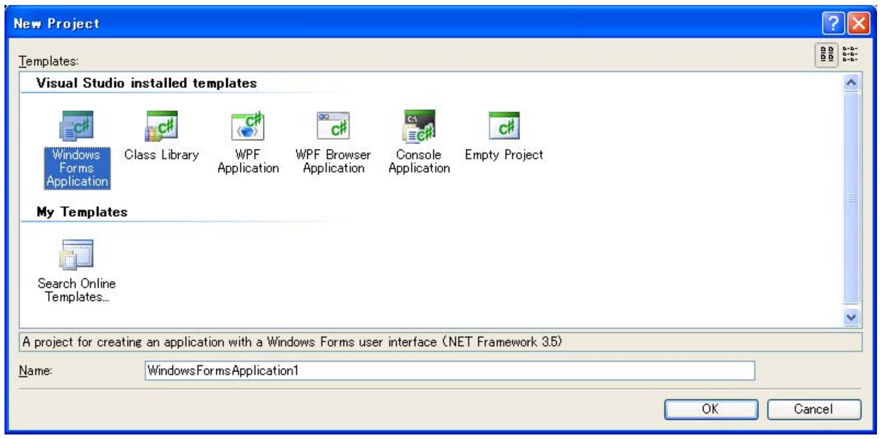



**Figure C.1.1-2　Add Reference Dialog Box**

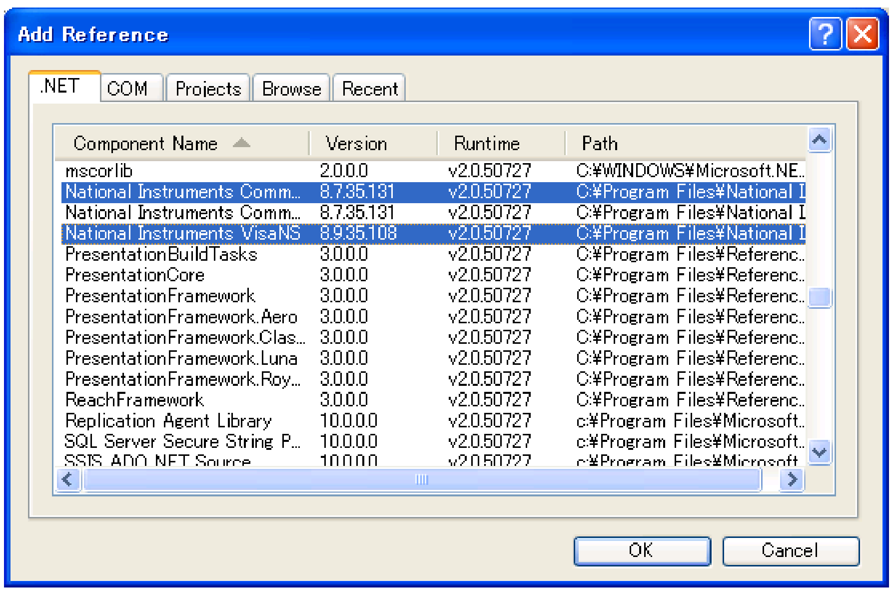
## C.1.2 Executing Sample Program

The executing procedure for the sample program is as follows.

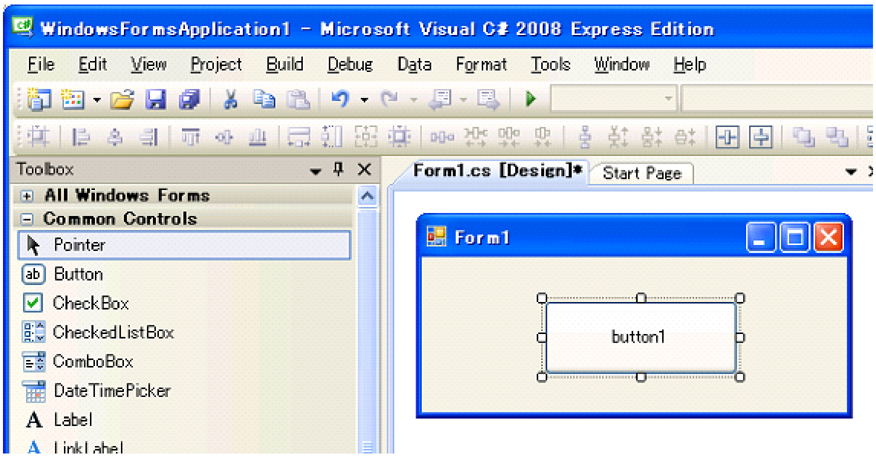
1. Start the Visual C# 2008.
2. Click [New Project] from the File menu.
3. Select the Visual C# Windows Forms Application and click [OK].




4. Start the screen editor and click [Add Reference] at the Project menu.
5. Click the .NET tab in the Add Reference dialog box.
6. Select National Instruments Common and National Instruments VisaNS and click [OK].




7. Referring to the sample program screen design figure, arrange control of the buttons in Form1.cs [Design].




8. Double-click the arranged button to open the screen for inputting the source code.

```
private void button1_Click(object sender, System.EventArgs
   e)
{
}
```

9. Copy the sample program in this document and paste it into the Form1.cs screen.

```
private void button1_Click(object sender, System.EventArgs
     e)
{
                //Paste it into this part.
      }
```

10. Change the IP address and GPIB address.

    For a LAN connection, the part "192.168.12.10" in the program must be changed to the IP address set at the BERTWave.

    For a GPIB connection, number of the part "GPIB::1::INSTR" in the program must be changed to GPIB address of BERTWave.

11. Click [Open Debug] from the Debug menu.

# C.2 Example 1: Controlling Pulse Pattern Generator

This sample program controls the instrument via the Ethernet interface.

Add the following declaration before starting the program.

```
using System.Net;
using System.Net.Sockets;
```

### Processing Flow

1. Define the TCP/IP client class for the IP address 192.168.12.10 and port number 5001.
2. Send :MODULE:ID 1 to set the control target to PPG/CH_Ch1.
3. Set the reference clock to the internal clock.
4. Set the bit rate specifications to 1GbE.
5. Set the pattern to PRBS2^23–1.
6. Set the amplitude to 0.5 V.
7. Set the error insertion to Off.
8. Output the signal of the PPG/CH_Ch1.
9. Set all module output to On.
10. Query errors.

```
//  set IP address of BERTWave
string IPadr="192.168.12.10";
//  set  port number of BERTWave
Int32 port = 5001;
TcpClient client = new TcpClient(IPadr, port);
NetworkStream stream = client.GetStream();

//  send messages
string message = ":MOD:ID 1\n";
Byte[] data = System.Text.Encoding.ASCII.GetBytes(message);
stream.Write(data, 0, data.Length);
Console.WriteLine("Sent: {0}", message);

message = ":OUTPUT:CMU:REFCLOCK INTERNAL\n";
data = System.Text.Encoding.ASCII.GetBytes(message);
stream.Write(data, 0, data.Length);
Console.WriteLine("Sent: {0}", message);

message = ":OUTPUT:BITRATE:STANDARD '1GBE'\n";
data = System.Text.Encoding.ASCII.GetBytes(message);
stream.Write(data, 0, data.Length);
Console.WriteLine("Sent: {0}", message);

message = ":SOURCE:PATTERN:TYPE PRBS23\n";
data = System.Text.Encoding.ASCII.GetBytes(message);
stream.Write(data, 0, data.Length);
Console.WriteLine("Sent: {0}", message);

message = ":OUTPUT:DATA:AMPLITUDE DATA,0.5\n";
data = System.Text.Encoding.ASCII.GetBytes(message);
stream.Write(data, 0, data.Length);
Console.WriteLine("Sent: {0}", message);

message = ":SOURCE:PATTERN:EADDITION:SET OFF\n";
data = System.Text.Encoding.ASCII.GetBytes(message);
stream.Write(data, 0, data.Length);
Console.WriteLine("Sent: {0}", message);

message = ":OUTPUT:DATA:OUTPUT ON\n";
data = System.Text.Encoding.ASCII.GetBytes(message);
stream.Write(data, 0, data.Length);
Console.WriteLine("Sent: {0}", message);

message = ":SYSTEM:ERROR?\n";
```

```
data = System.Text.Encoding.ASCII.GetBytes(message);
stream.Write(data, 0, data.Length);
Console.WriteLine("Sent: {0}", message);

//  read response
data = new Byte[256];
String responseData = String.Empty;
Int32 bytes = stream.Read(data, 0, data.Length);
responseData = System.Text.Encoding.ASCII.GetString(data, 0, bytes);
Console.WriteLine("Received: {0}", responseData);

// Close everything.
stream.Close();
client.Close();
```

## C.3 Example 2: Controlling Error Detector

This sample program controls the instrument via the GPIB interface.

### Processing Flow

1. Define the class of the GPIB address 1.
2. Send :MODULE:ID 1 to set the control target to PPG/CH_Ch1.
3. Set the bit rate specifications to 10G FC.
4. Set the pattern to PRBS2^23–1.
5. Set the input connector to Data only.
6. Set the threshold value to 0 V.
7. Set the auto-pattern sync to On.
8. Set the threshold value of the auto-pattern sync threshold value to 1E−5.
9. Set the error measurement method to Single.
10. Set the measurement time to 20 seconds.
11. Start the measurement.
12. Query errors.

```
 // Open session
 NationalInstruments.VisaNS.MessageBasedSession mbs =
(NationalInstruments.VisaNS.MessageBasedSession)
NationalInstruments.VisaNS.ResourceManager.GetLocalManager().
Open("GPIB::1::INSTR");

 mbs.Timeout = 30000; // Timeout 30sec

 //  Initialize BERTWave.
 mbs.Write("*RST\n");
 //  Select module as PPG/ED_Ch1.
 mbs.Write(":MODULE:ID 1\n");
 //  Set bitrate standard of ED as 1 Giga bit Ethernet.
 mbs.Write( ":INPUT:BITRATE:STANDARD '10G_FC'\n");
 //   Set test pattern of ED as PRBS2^23-1.
 mbs.Write(":SENSE:PATTERN:TYPE PRBS23\n");
 //   Set input connector as Data only.
 mbs.Write( ":INPUT:DATA:INTERFACE DATA\n");
 //   Set threshold voltage as 0V.
 mbs.Write( ":INPUT:DATA:THRESHOLD 0\n");
 //   Set automatic pattern synchronization as On.
 mbs.Write(":SENSE:PATTERN:SYNC:ASYNC ON\n");
 //   Set threshold level of automatic synchronization as 10^-5.
 mbs.Write(":SENSE:PATTERN:SYNC:THRESHOLD E_5\n");
 //   Set measure mode as Single.
 mbs.Write(":SENSE:MEASURE:EALARM:MODE SINGLE\n");
 //   Set measuring time as 20 seconds.
 mbs.Write( ":SENSE:MEASURE:EALARM:PERIOD 0,0,0,20\n");
 //   Start measurement.
 mbs.Write(":SENSE:MEASURE:START\n");
 string ret = mbs.Query(":SYSTEM:ERROR?\n");
 Console.WriteLine(ret);
```

# C.4 Example 3: Controlling Optical Transceiver

This sample program controls the instrument via the Ethernet interface.

Add the following declaration before starting the program.

```
using System.Net;
using System.Net.Sockets;
```

### Processing Flow

1. Define the TCP/IP client class for the IP address 192.168.12.10 and port number 5001.
2. Send :MODULE:ID 3 to set the control target to XFP/SFP+.
3. Query whether the optical transceiver is installed or not.
4. Query the wavelength of the optical transceiver.
5. Set the optical transceiver output to ON.
6. Query errors.

```
// set IP address of BERTWave
string IPadr = "192.168.12.10";
// set port number of BERTWave
Int32 port = 5001;
TcpClient client = new TcpClient(IPadr, port);
NetworkStream stream = client.GetStream();

// send messages
string message = ":MOD:ID 3\n";
Byte[] data = System.Text.Encoding.ASCII.GetBytes(message);
stream.Write(data, 0, data.Length);
Console.WriteLine("Sent: {0}", message);

message = ":CALCULATE:OPTICAL:STATUS? 'READY'\n";
data = System.Text.Encoding.ASCII.GetBytes(message);
stream.Write(data, 0, data.Length);
Console.WriteLine("Sent: {0}", message);

// read response
data = new Byte[256];
String responseData = String.Empty;
Int32 bytes = stream.Read(data, 0, data.Length);
responseData = System.Text.Encoding.ASCII.GetString(data, 0, bytes);
Console.WriteLine("Received: {0}", responseData);

if (responseData == "N")
{
// Result is "None"
Console.WriteLine("Optical Transceiver does not exist.");
return;
}

message = ":SOURCE:OPTICAL:SIGNAL:WLENGTH?\n";
data = System.Text.Encoding.ASCII.GetBytes(message);
stream.Write(data, 0, data.Length);
Console.WriteLine("Sent: {0}", message);

// read response
data = new Byte[256];
responseData = String.Empty;
bytes = stream.Read(data, 0, data.Length);
responseData = System.Text.Encoding.ASCII.GetString(data, 0, bytes);
Console.WriteLine("Received: {0}", responseData);
```

```
message = ":SOURCE:OPTICAL:SIGNAL:OUTPUT 1\n";
data = System.Text.Encoding.ASCII.GetBytes(message);
stream.Write(data, 0, data.Length);
Console.WriteLine("Sent: {0}", message);

message = ":SYSTEM:ERROR?\n";
data = System.Text.Encoding.ASCII.GetBytes(message);
stream.Write(data, 0, data.Length);
Console.WriteLine("Sent: {0}", message);

//  read response
data = new Byte[256];
responseData = String.Empty;
bytes = stream.Read(data, 0, data.Length);
responseData = System.Text.Encoding.ASCII.GetString(data, 0, bytes);
Console.WriteLine("Received: {0}", responseData);

// Close everything.
stream.Close();
client.Close();
```

# C.5 Example 4: Controlling Scope

This sample program controls the instrument via the GPIB interface.

Processing Flow

1. Define the class of the GPIB address 1.
2. Send :MODULE:ID 5 to set the control target to Scope.
3. Set the display mode to the pulse mode.
4. Start the measurement.
5. Query the measurement status.
6. Query errors.

```
// Open session
 NationalInstruments.VisaNS.MessageBasedSession mbs =
(NationalInstruments.VisaNS.MessageBasedSession)
NationalInstruments.VisaNS.ResourceManager.GetLocalManager().
Open("GPIB::1::INSTR");

 mbs.Timeout = 30000; // Timeout 30sec

 mbs.Write(":MODULE:ID 5\n");
 mbs.Write(":DISPLAY:WINDOW:GRAPHICS:CLEAR\n");
 //   Set measuring mode as EYE mode.
 mbs.Write(":SENSE:DISPLAY:MODE PULSE\n");
 //   Start measurement.
 mbs.Write(":SENSE:SAMPLING:STATUS RUN\n");
 string ret = mbs.Query(":SENSE:SAMPLING:STATUS?\n");
 Console.WriteLine(ret);
 ret = mbs.Query(":SYSTEM:ERROR?\n");
 Console.WriteLine(ret);
```

# Appendix D BASIC Sample Program

This appendix describes the sample program in Appendix D using the BASIC language.

# D.1 Sample Program Operating Environment

## D.1.1 Setting Sample Program Operating Environment

The sample program operating environment is as follows.

PC

OS: Windows XP Professional Service Pack 2

VISA: NI-VISA Version 4.6

Program tool: Microsoft Visual BASIC 2008 Express Edition

MP2100A BERTWave

GPIB Address: 1

IP Address: 198.168.12.10

Subnet Mask: 255.255.255.0

Port Number: 5001

Installing NI-VISA
To use VISA at Visual BASIC 2008, add the following function at installation.

- Development Support .NET Framework 3.5 Language Support
- NI Measurement & Automation Explore –.NET Framework 3.5 Language Support

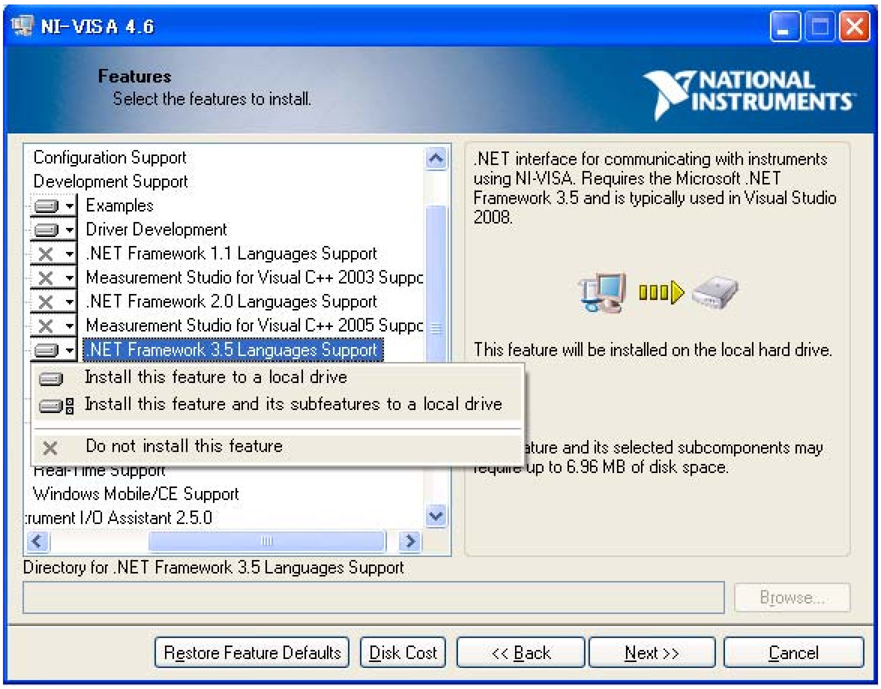
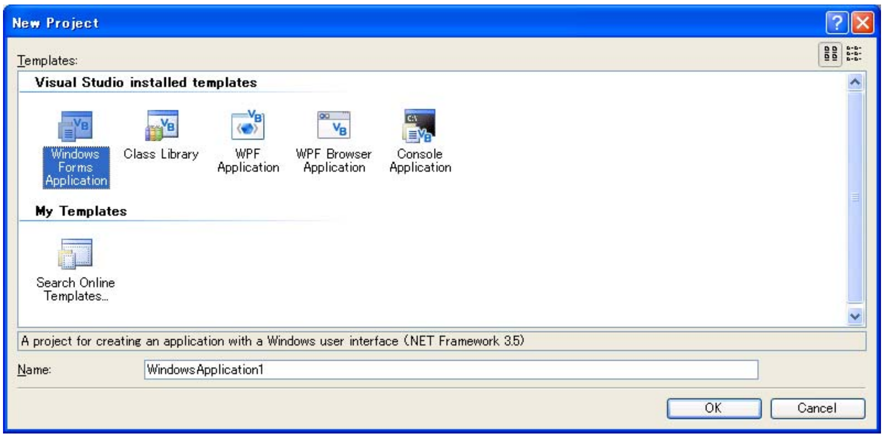



**Figure D.1.1-1 Function Selection Screen at VISA Install**

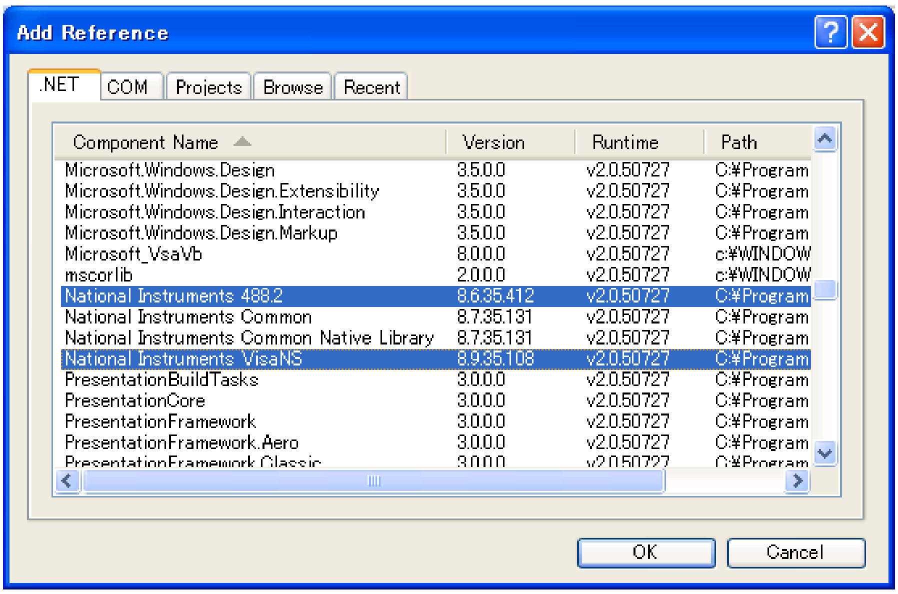
## D.1.2 Executing Sample Program

To execute the sample program, follow the below procedures.

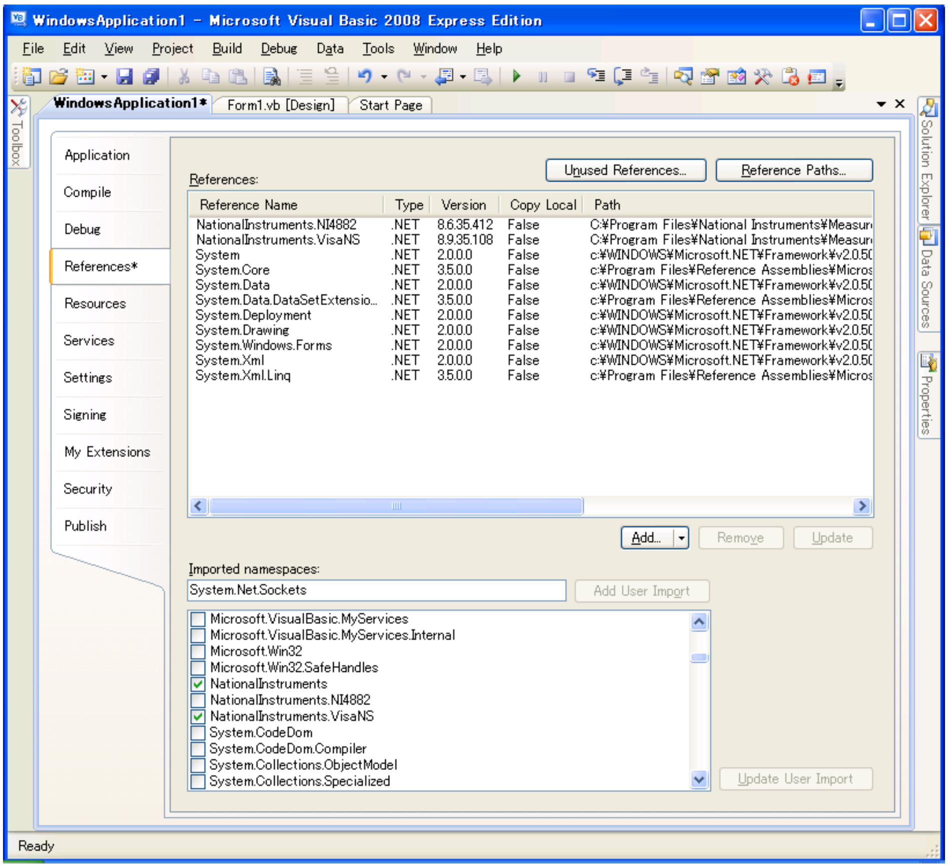
1. Start the Visual BASIC 2008.
2. Click [File]–[New Project].
3. Select the Visual Basic of the Windows application, and click [OK].



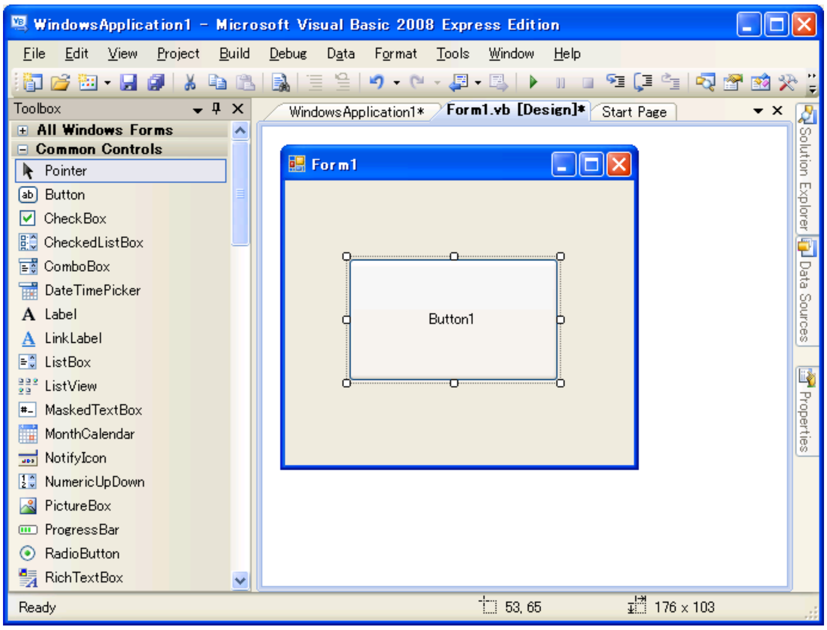

4. The editor to edit the screen is started. Click [Project] –[Add Reference] from the menu bar.
5. In the Add Reference dialog box, click [.NET].
6. Select [National Instruments Common] and [. National Instruments VisaNS] and click [OK].




7. Click [View] — [Solution Explore] from the menu bar.
8. Double-click [My Project] at the Solution Explore.
9. Check National Instruments Common and National Instruments VisaNS, System.Net.Socket in the imported name spaces.



10. Click [Add].
11. In the Add Reference dialog box, click [OK].
12. Allocate the button control in the Form1.vb [Design].



13 Double-click the arranged button to open the screen for inputting the source code. The following code is added in the automatic-generated Form1.Desiner.cs.

```
Private Sub Button1_Click(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles Button1.Click

End Sub
```

14. Copy the sample program in this document and paste it into the Form1.Desiner.cs.

```
Private Sub Button1_Click(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles Button1.Click

     ' Paste it into this part.

End Sub
```

15. Change the IP address and GPIB address.

    For an Ethernet connection, the part "192.168.12.10" described above must be changed to the IP address set at the BERTWave.

    For a GPIB connection, the part " GPIB::1::INSTR " described above must be changed to the GPIB address of the BERTWave.

16. Set the Ethernet IP address using at the control PC.
    The IP address must be set the same as the BERTWave IP address.

17. Click [Open Debug] from the Debug menu.

# D.2 Example 1: Controlling Pulse Pattern Generator

This sample program control the instrument via the Ethernet interface.

Processing Flow

1. Define the TCP/IP client class for the IP address 192.168.12.10 and port number 5001.
2. Send :MODULE:ID 1 to set the control target to PPG/ED_Ch1.
3. Set the reference clock to the internal clock.
4. Set the bit rate specifications to 1GbE.
5. Set the pattern to PRBS2^23-1.
6. Set the amplitude to 0.5 V.
7. Set the error insertion to Off.
8. Output the signal of the PPG Ch1.
9. Query errors.

```
' Create a TcpClient.
Dim LF As String = Chr(10)
Dim server As String = "192.168.12.10"
Dim Message As String
Dim port As Int32 = 5001
Dim client As New TcpClient(server, port)

' Get a client stream for reading and writing.
Dim stream As NetworkStream = client.GetStream()

 ' Translate the passed message into ASCII and store it as a Byte array.
 Message = ":MODULE:ID 1"
 Dim data As [Byte]() = System.Text.Encoding.ASCII.GetBytes(Message + LF)
 ' Send the message to the connected TcpServer.
 stream.Write(data, 0, data.Length)
 Console.WriteLine("Sent: " + Message)

 Message = ":OUTPUT:CMU:REFCLOCK INTERNAL"
 data = System.Text.Encoding.ASCII.GetBytes(Message + LF)
 stream.Write(data, 0, data.Length)
 Console.WriteLine("Sent: " + Message)

 Message = ":OUTPUT:BITRATE:STANDARD '1GBE'"
 data = System.Text.Encoding.ASCII.GetBytes(Message + LF)
 stream.Write(data, 0, data.Length)
 Console.WriteLine("Sent: " + Message)

 Message = ":SYSTEM:ERROR?"
 data = System.Text.Encoding.ASCII.GetBytes(Message + LF)
 stream.Write(data, 0, data.Length)
 Console.WriteLine("Sent: " + Message)

 Message = ":SOURCE:PATTERN:TYPE PRBS23"
 data = System.Text.Encoding.ASCII.GetBytes(Message + LF)
 stream.Write(data, 0, data.Length)
 Console.WriteLine("Sent: " + Message)

 Message = ":SOURCE:PATTERN:EADDITION:SET OFF"
 data = System.Text.Encoding.ASCII.GetBytes(Message + LF)
 stream.Write(data, 0, data.Length)
 Console.WriteLine("Sent: " + Message)

 Message = ":OUTPUT:DATA:OUTPUT 1"
 data = System.Text.Encoding.ASCII.GetBytes(Message + LF)
```

```
 stream.Write(data, 0, data.Length)
 Console.WriteLine("Sent: " + Message)

 ' String to store the response ASCII representation.
 Dim responseData As [String] = [String].Empty

 ' Read the first batch of the TcpServer response bytes.
 Dim bytes As Int32 = stream.Read(data, 0, data.Length)
 responseData = System.Text.Encoding.ASCII.GetString(data, 0, bytes)
 Console.WriteLine("Received: " + responseData)

' Close everything.
stream.Close()
client.Close()
```

# D.3 Example 2: Controlling Error Detector

This sample program controls the instrument via the GPIB interface.

### Processing Flow

1. Define the class of the GPIB address 1.
2. Send :MODULE:ID 1 to set the control target to PPG/CH_Ch1.
3. Set the tracking of the PPG to Off.
4. Set the bit rate specifications to 10G FC.
5. Set the pattern to PRBS2^23-1.
6. Set the input connector to Single Ended Data.
7. Set the threshold value to 0 V.
8. Set the auto-pattern sync to On.
9. Set the threshold value of the auto-pattern sync to 1E-5.
10. Set the error measurement method to Single.
11. Set the measurement time to 20 seconds.
12. Start the measurement.
13. Query errors.

```
Dim gbs As GpibSession
Dim message, ret As String
gbs    =    CType(ResourceManager.GetLocalManager().Open("GPIB::1::INSTR"),
GpibSession)
gbs.Timeout = 30000

 '  Select module as PPG/ED_Ch1.
 message = ":MODULE:ID 1"
 gbs.Write(message)
 Console.WriteLine("Sent: " + message)
 '  Set tracking to PPG off.
 message = ":SENSE:PARAM:TRACKING 0"
 gbs.Write(message)
 Console.WriteLine("Sent: " + message)
 '  Set bitrate standard of ED as 10 Giga bit Fiber Channel.
 message = ":INPUT:BITRATE:STANDARD '10G_FC'"
 gbs.Write(message)
 Console.WriteLine("Sent: " + message)
 '   Set test pattern of ED as PRBS2^23-1.
 message = ":SENSE:PATTERN:TYPE PRBS23"
 gbs.Write(message)
 Console.WriteLine("Sent: " + message)

 '   Set input connector as Data only.
 message = ":INPUT:DATA:INTERFACE DATA"
 gbs.Write(message)
 Console.WriteLine("Sent: " + message)

 '   Set threshold voltage as 0V.
 message = ":INPUT:DATA:THRESHOLD 0"
 gbs.Write(message)
 Console.WriteLine("Sent: " + message)
 '   Set automatic pattern synchronization as On.
 message = ":SENSE:PATTERN:SYNC:ASYNC ON"
 gbs.Write(message)
 Console.WriteLine("Sent: " + message)
 '   Set threshold level of automatic synchronization as 10^-5.
 message = ":SENSE:PATTERN:SYNC:THRESHOLD E_5"
 gbs.Write(message)
 Console.WriteLine("Sent: " + message)
 '   Set gating mode as Single.
 message = ":SENSE:MEASURE:EALARM:MODE SINGLE"
 gbs.Write(message)
```

```
Console.WriteLine("Sent: " + message)
'   Set gating time as 20 seconds.
message = ":SENSE:MEASURE:EALARM:PERIOD 0,0,0,20"
gbs.Write(message)
Console.WriteLine("Sent: " + message)

'   Start measurement.
message = ":SENSE:MEASURE:START"
gbs.Write(message)
Console.WriteLine("Sent: " + message)

ret = gbs.Query(":SYSTEM:ERROR?")
Console.WriteLine("Received: " + ret)
```

# D.4 Example 3: Controlling Optical Transceiver

This sample program controls the instrument via the Ethernet cable.

### Processing Flow

1. Define the TCP/IP client class for the IP address 192.168.12.10 and port number 5001.
2. Send :MODULE:ID 3 to set the target control to XFP/SFP+.
3. Query whether the optical transceiver is installed or not.
   When the optical transceiver is installed, perform the processing at Step 4 and 5.
4. Query the wavelength of the optical transceiver.
5. Set the output of the optical transceiver to On.
6. Query errors.

```
' Create a TcpClient.
Dim LF As String = Chr(10)
Dim server As String = "192.168.12.10"
Dim Message As String
Dim port As Int32 = 5001
Dim client As New TcpClient(server, port)

' Get a client stream for reading and writing.
Dim stream As NetworkStream = client.GetStream()

 ' Translate the passed message into ASCII and store it as a Byte array.
 Message = ":MODULE:ID 3"
 Dim data As [Byte]() = System.Text.Encoding.ASCII.GetBytes(Message + LF)
 ' Send the message to the connected TcpServer.
 stream.Write(data, 0, data.Length)
 Console.WriteLine("Sent: " + Message)

 Message = ":CALCULATE:OPTICAL:STATUS? 'READY'"
 data = System.Text.Encoding.ASCII.GetBytes(Message + LF)
 stream.Write(data, 0, data.Length)
 Console.WriteLine("Sent: " + Message)

 ' String to store the response ASCII representation.
 Dim responseData As [String] = [String].Empty

 ' Read the first batch of the TcpServer response bytes.
 Dim bytes As Int32 = stream.Read(data, 0, data.Length)
 responseData = System.Text.Encoding.ASCII.GetString(data, 0, bytes)
 Console.WriteLine("Received: " + responseData)

 If (responseData(1) = "N") Then

     '  Result is "None"
     Console.WriteLine("Optical Transceiver does not exist.")

 Else

     Message = ":SOURCE:OPTICAL:SIGNAL:WLENGTH?"
     data = System.Text.Encoding.ASCII.GetBytes(Message + LF)
     stream.Write(data, 0, data.Length)
     Console.WriteLine("Sent: " + Message)

     responseData = [String].Empty
     bytes = stream.Read(data, 0, data.Length)
```

```
        responseData = System.Text.Encoding.ASCII.GetString(data, 0, bytes)
        Console.WriteLine("Received: " + responseData)

        Message = ":SOURCE:OPTICAL:SIGNAL:OUTPUT 1"
        data = System.Text.Encoding.ASCII.GetBytes(Message + LF)
        stream.Write(data, 0, data.Length)
        Console.WriteLine("Sent: " + Message)

    End If

     Message = ":SYSTEM:ERROR?"
     data = System.Text.Encoding.ASCII.GetBytes(Message + LF)
     stream.Write(data, 0, data.Length)
     Console.WriteLine("Sent: " + Message)

     responseData = [String].Empty
     bytes = stream.Read(data, 0, data.Length)
     responseData = System.Text.Encoding.ASCII.GetString(data, 0, bytes)
     Console.WriteLine("Received: " + responseData)

    ' Close everything.
    stream.Close()
    client.Close()
```

# D.5 Example 4: Controlling Scope

This sample program controls the instrument via the GPIB interface.

Processing Flow

1. Define the class of the GPIB address 1.
2. Send :MODULE:ID 5 to set the control target to Scope.
3. Clear the screen.
4. Set the display mode to the pulse mode.
5. Query the measurement execution status.
6. Start the measurement when Sampling HOLD is set.
7. Query errors.

```
Dim gbs As GpibSession
Dim message, ret As String
gbs    =    CType(ResourceManager.GetLocalManager().Open("GPIB::1::INSTR"),
GpibSession)
gbs.Timeout = 30000

 '   Select module as EYE/Pulse Scope.
 message = ":MODULE:ID 5"
 gbs.Write(message)
 Console.WriteLine("Sent: " + message)
 '   Clear Display.
 message = ":DISPLAY:WINDOW:GRAPHICS:CLEAR"
 gbs.Write(message)
 Console.WriteLine("Sent: " + message)
 '   Set measuring mode as Pulse mode.
 message = ":SENSE:DISPLAY:MODE PULSE"
 gbs.Write(message)
 Console.WriteLine("Sent: " + message)
 '   Query Status.
 ret = gbs.Query(":SENSE:SAMPLING:STATUS?")
 Console.WriteLine("Received: " + ret)

 If ret(0) = "H" Then
     '   Start measurement.
     message = ":SENSE:SAMPLING:STATUS RUN"
     gbs.Write(message)
     Console.WriteLine("Sent: " + message)
 End If

 ret = gbs.Query(":SYSTEM:ERROR?")
 Console.WriteLine("Received: " + ret)
```

Appendix

Appendix D

# *Appendix E Bibliography*


(1) IEEE488.1-1987 *IEEE Standard Digital Interface for Programmable Instrumentation -Description*

(2) IEEE488.2-1992 *IEEE Standard Codes, Formats, Protocols, and Common Commands for Use With IEEE Std 488.1-1987, IEEE Standard Digital Interface for Programmable Instrumentation -Description*

(3) IVI Foundation *SCPI 1999*

(4) IEEE802.3-2005 *IEEE Standard for Information technology.*
*Telecommunications and information exchange between systems.*
*Local and metropolitan area networks.*
*Specific requirements*
*Part 3: Carrier sense multiple access with collision detection (CSMA/CD) access method and physical layer specifications.*

(5) Microsoft Corporation Microsoft *Visual Studio 2008 Document*

(6) National Instruments Corporation *NI-VISA .NET Framework 3.5 Help*

(7) Anritsu Corporation *MX180000A Signal Quality Analyzer Control Software Remote Control Operation Manual*
http://dl.cdn-anritsu.com/en-au/test-measurement/files/Manuals/Operation-Manual/mx180000a_remote_opm_e_21_0.pdf

(8) Anritsu Corporation *Ideal Remote-Control Sequences*
http://dl.cdn-anritsu.com/en-au/test-measurement/files/Application-Notes/Application-Note/MP2100A_EF5100.pdf