User's
Manual

SL1000 High-Speed Data Acquisition Unit Communication Interface

YOKOGAWA
Yokogawa Test & Measurement Corporation

IM 720320-17E
5th Edition

Thank you for purchasing the SL1000 API Control. This Communication Interface User's Manual describes the functions and commands of the following communication interfaces.

- USB Interface
- Ethernet Interface (Optional)

To ensure correct use, please read this manual thoroughly before beginning operation. After reading the manual, keep it in a convenient location for quick reference whenever a question arises during operation.

List of Manuals

The following manuals are provided for the SL1000. Please read all of them.

Manual Title	Manual No.	Description
SL1000 High-Speed Data Acquisition Unit User's Manual	IM 720120-01E	Explains how to install the SL1000 and its input modules, and explains features related to the hardware, such as the display, and how to operate them.
SL1000 Acquisition Software User's Manual	IM 720120-61E	Explains all functions and procedures of the Acquisition Software used to configure and control the SL1000.
SL1000 Input Module User's Manual	IM 720120-51E	Explains the specifications of the input modules that can be installed in the SL1000.
701992 Xviewer User's Manual	IM 701992-01E	Explains all functions and procedures of the Xviewer software used to display the measured data as waveforms on a PC. This manual is not included with the /XV0 option.
SL1000 Control API User's Manual	IM 720320-01E	It explains the functions for controlling the SL1000 (the SL1000 control API).
SL1000 High-Speed Data Acquisition Unit Communication Interface User's Manual	IM 720320-17E	This manual. Explains the communication interface functions of the SL1000.

Notes

- **This manual, IM 720320-17E, applies to SL1000 High-Speed Data Acquisition Unit with firmware version 2.40 or later.**
  If the most recent firmware version is not running on your SL1000 not all of the features described in this manual can be used.
  You can check the firmware version of your SL1000 on the overview screen. For instructions on how to open the overview screen, see section 9.6 in the User's Manual IM 720120-61E. For instructions on how to update the firmware and for information about firmware versions, see the following Webpage.
  http://www.yokogawa.com/tm/
- The contents of this manual are subject to change without prior notice as a result of continuing improvements to the instrument's performance and functions.
- Every effort has been made in the preparation of this manual to ensure the accuracy of its contents. However, should you have any questions or find any errors, please contact your nearest YOKOGAWA dealer.
- Copying or reproducing all or any part of the contents of this manual without the permission of YOKOGAWA is strictly prohibited.

5th Edition: November 2017 (YMI)

Trademarks

- Microsoft, MS-DOS, Visual C++, Windows, Windows XP, and Windows NT are registered trademarks of Microsoft Corporation in the United States and/or other countries.
- Adobe and Acrobat are trademarks of Adobe Systems Incorporated.
- For purposes of this manual, the ® and TM symbols do not accompany their respective trademark names or registered trademark names.
- Other company and product names are trademarks or registered trademarks of their respective holders.

Revisions

1st Edition: December 2008
2nd Edition: December 2009
3rd Edition: September 2013
4th Edition: July 2017
5th Edition: November 2017

How to Use This Manual

Structure of This Manual

This user’s manual consists of the following sections.

Chapter 1 Connecting to the PC

Describes the procedure for connecting to the PC using the USB and Ethernet interfaces.

Chapter 2 Before Programming

Describes the syntax used to transmit commands.

Chapter 3 Command

Describes all the commands one by one.

Chapter 4 Status Reports

Describes the status byte, various registers, and queues.

Appendix

Describes reference material such as an ASCII character code table.

Symbols and Notations Used in This Manual

Safety Markings

The following markings are used in this manual.

CAUTION Calls attentions to actions or conditions that could cause light injury to the user or damage to the instrument or the user's data, and precautions that can be taken to prevent such occurrences.

Note Calls attention to information that is important for proper operation of the instrument.

Notation Used in the Procedural Explanations

On pages that describe the operating procedures in chapters 1 through 3, the following notations are used to distinguish the procedures from their explanations.

Procedure Carry out the procedure according to the step numbers. All procedures are written with inexperienced users in mind; experienced users may not need to carry out all the steps.

Explanation This section describes the setup items and the limitations regarding the procedures.

Notation of User Controls

Operation/Soft Key Names and Menu Items Set in Boldface

Boldface type indicates the names of user-controlled operation keys on the instrument panel, and soft key items and menu items displayed on screen.

SHIFT+Panel Key

The SHIFT+Panel key means you will press the SHIFT key to turn ON the indicator of SHIFT key and then press the panel key. The menu marked in purple above the pressed key appears on the screen.

Unit

k: Denotes "1000."	Example: 100 kS/s (sample rate)
K: Denotes "1024."	Example: 720 KB (file data size)

Symbols Used in the Syntax

The following table indicates symbols that are used in the syntax mainly in chapters 2 and 3. These symbols are referred to as BNF (Backus-Naur Form) symbols.

Symbol	Meaning	Example	Example of Input
< >	Defined value	CHANnel<x> <x> = 1 to 4	CHANNEL2
{ }	Select from values given in { }	COUPling {AC|DC|DC50|GND}	COUPLING AC
|	Exclusive OR		
[ ]	Can be omitted	TRIGger [:SIMPle]:SLOPe	TRIGger:SLOPe

Contents

Chapter 1 Connecting to a PC

Chapter 2 Before Programming

Chapter 3 Commands

Chapter 4 Status Reports

Appendix

1.1 Connecting via USB

Procedure

Use the following procedure to connect the SL1000 High-Speed Data Acquisition Unit (hereinafter, the SL1000 unit) to the PC.

Installing the Acquisition Software

Install the acquisition software that came with the SL1000 unit on the PC. For instructions, see section 2.2, “Installing or Uninstalling the Acquisition Software” in the SL1000 Acquisition Software User’s Manual (IM 720120-61E).

Connecting via USB Cable

Connect the SL1000 unit to the PC by following the procedure in section 4.1, “Connecting to a PC” in the SL1000 High-Speed Data Acquisition Unit User’s Manual (IM 720120-01E).

Installing the USB Driver

Install the USB driver on the PC by following the procedure in section 2.3, “Installing the USB Driver” in the SL1000 Acquisition Software Unit User’s Manual (IM 720120-61E). This is only necessary the first time the SL1000 unit is connected to the PC.

Note

The USB and Ethernet interfaces cannot be used at the same.

1.2 Connecting via Ethernet (Optional)

Procedure

Use the following procedure to connect the SL1000 unit to the PC.

Note

Because communications settings are also entered on the SL1000 unit when connecting via Ethernet, at first it is necessary to connect via USB.

Installing the Acquisition Software (First Time Only)

Install the acquisition software that came with the SL1000 unit on the PC. For instructions, see section 2.2, “Installing or Uninstalling the Acquisition Software” in the SL1000 Acquisition Software User’s Manual (IM 720120-61E).

Connecting via USB Cable

Connect the SL1000 unit to the PC by following the procedure in section 4.1, “Connecting to a PC” in the SL1000 High-Speed Data Acquisition Unit User’s Manual (IM 720120-01E).

Installing the USB Driver (First Time Only)

Install the USB driver on the PC by following the procedure in section 2.3, “Installing the USB Driver” in the SL1000 Acquisition Software Unit User’s Manual (IM 720120-61E). This is only necessary the first time the SL1000 unit is connected to the PC.

Entering Communication Settings (TCP/IP Settings)

Start the acquisition software, then enter communication settings for the SL1000 unit. For instructions, see section 3.2, “Specifying Communication Settings (When Using the Optional Ethernet Interface)” in the SL1000 Acquisition Software User’s Manual (IM 720120-61E).

Connecting via Ethernet Cable

Close the acquisition software, then turn OFF the power to the SL1000 unit. Connect the SL1000 unit to the PC via Ethernet by following the procedure in section 4.1, “Connecting to a PC” in the SL1000 High-Speed Data Acquisition Unit User’s Manual (IM 720120-01E).

2.1 Messages

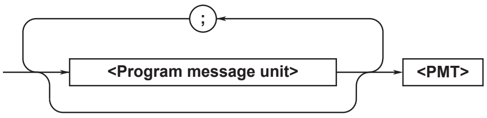
Messages

Messages are used to exchange information between the controller and the instrument. Messages that are sent from the controller to the instrument are called program messages and messages that are sent back from the instrument to the controller are called response messages.

If a program message contains a message unit that requests a response (a query), the instrument returns a response message upon receiving the program message. A single response message is always returned in response to a single program message.

Program Messages

The program message format is shown below



<Program Message Unit>

A program message consists of zero or more program message units; each unit corresponds to one command. The instrument executes the received commands in order.

Each program message unit is separated by a semicolon (`;`).

For details regarding the format of the program message unit, see the next section.

Example

$\underbrace{\texttt{:ACQuire:MODE NORMal}}_{\text{Unit}}\texttt{;}\underbrace{\texttt{MEASure:MODE ON}}_{\text{Unit}}\texttt{<PMT>}$

<PMT>

PMT is a program message terminator. The following three types are available.

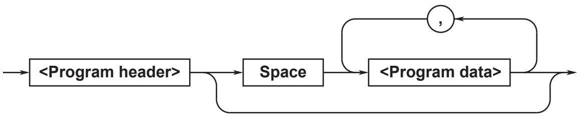
`NL` (New Line): Same as LF (Line Feed). ASCII code “0AH”

`^EOM`: The END message as defined by USBTMC (The data byte that is sent simultaneously with the END message is the last data of the program message.)

`NL^EOM`: NL with an END message added (NL is not included in the program message.)

Program Message Unit Format

The program message unit format is shown below



<Program Header>

The program header indicates the command type. For details, see page 4-3.

<Program Data>

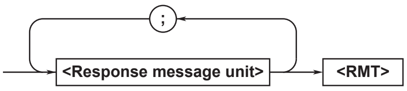
If certain conditions are required in executing a command, program data is added. A space (ASCII code “20H”) separates the program data from the header. If there are multiple sets of program data, they are separated by commas (`,`).

For details, see page 4-5.

Example $\underbrace{\texttt{:ACQuire:MODE}}_{\text{Header}}\ \underbrace{\texttt{NORMal}}_{\text{Data}}\texttt{<PMT>}$

Response Messages

The response message format is shown belo.



<Response Message Unit>

A response message consists of one or more response message units; each response message unit corresponds to one response.

Response message units are separated by a semicolon (`;`).

For details regarding the format of the response message unit, see the next section.

Example

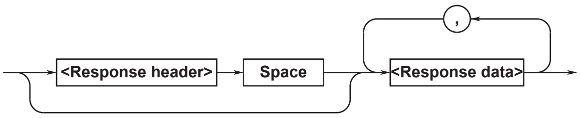
$\underbrace{\texttt{:ACQuire:MODE NORMal}}_{\text{Unit}}\texttt{;}\underbrace{\texttt{MEASure:MODE ON}}_{\text{Unit}}\texttt{<PMT>}$

<RMT>

A response message terminator. It is `NL^EOM`.

Response Message Unit Format

The response message unit format is shown belw.



<Response Header>

A response header sometimes precedes the response data. A space separates the data from the header. For details, see page 2-4.

<Response Data>

Response data contains the content of the response. If there are multiple sets of response data, they are separated by commas (,). For details, see page 4-5.

Exaple

$\underbrace{\texttt{1.25E-02<RMT>}}_{\text{Data}}\quad \underbrace{\texttt{:ACQUIRE:MODE}}_{\text{Header}}\ \underbrace{\texttt{NORMAL}}_{\text{Data}}\texttt{<RMT>}$

If there are multiple queries in a program message, responses are made in the same order as the queries. In most cases, a single query returns a single response message unit, but there are a few queries that return multiple units. The first response message unit always corresponds to the first query, but the nth response unit may not necessarily correspond to the nth query. Therefore, if you want to make sure that every response is retrieved, divide the program messages into individual messages.

Precautions to Be Taken when Transferring Messages

- If a program message that does not contain a query is sent, the next program message can be sent at any time.
- If a program message that contains a query is sent, a response message must be received before the next program message can be sent. If the next program message is sent before the response message is received in its entirety, an error occurs. The response message that was not received is discarded.
- If the controller tries to receive a response message when there is none, an error occurs. If the controller tries to receive a response message before the transmission of the program message is complete, an error occurs.
- If a program message containing multiple message units is sent, and the message contains incomplete units, the instrument attempts to execute the ones that are believed to be complete. However, these attempts may not always be successful. In addition, if the message contains queries, the responses may not be returned.

Deadlock

The instrument can store in its buffer program and response messages of length 1024 bytes or more (The number of available bytes varies depending on the operating conditions). When both the transmit and receive buffers become full at the same time, the instrument can no longer continue to operate. This state is called a deadlock. In this case, operation can be resumed by discarding the program message. Deadlock will not occur if the program message (including the <PMT>) is kept below 1024 bytes. Furthermore, deadlock never occurs if a program message does not contain a query.

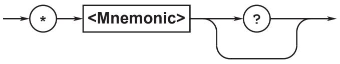
2.2 Commands

Commands

There are three types of commands (program headers) that are sent from the controller to the instrument. They differ in their program header formats.

Common Command Header

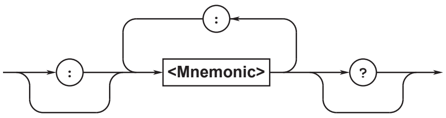
Commands that are defined in the USBTMC-USB488 are called common commands. The header format of a common command is shown below. An asterisk (*) is always placed in the beginning of a cmmand.



Common command example: `*CLS`

Compound Header

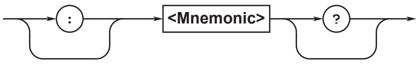
Dedicated commands used by the instrument are classified and arranged in a hierarchy according to their functions. The format of a compound header is shown below. A colon (:) must be used to specify a lower hiearchy.



Compound header example: `:ACQuire:MODE`

Simple Header

These commands are functionally independent and do not have a hierarchy. The format of a simple header is shownbelow.



Simple header example: `:STARt`

Note

A <mnemonic> is a character string made up of alphanumeric characters.

When Concatenating Commands

- **Command Group**

  A command group is a group of commands that have common compound headers arranged in a hierarchy. A command group may contain sub-groups.

  Example Group of commands related to acquisition

  ```
  :ACQuire:ECLock?
  :ACQuire:ECLock:PCOunt
  :ACQuire:ECLock:COUNt
  :ACQuire:MMODe
  :ACQuire:MODE
  :ACQuire:TIME
  ```

- **When Concatenating Commands of the Same Group**

  The instrument stores the hierarchical level of the command that is currently being executed, and performs analysis on the assumption that the next command sent will also belong to the same level. Therefore, common header sections can be omitted for commands belonging to the same group.

  Example
  ```
  ACQuire:MODE NORMal;
  TIME 0,0,0,0,500,0<PMT>
  ```

- **When Concatenating Commands of Different Groups**

  If the following command does not belong to the same group, a colon (:) is placed in front of the header (cannot be omitted).

  Example
  ```
  :ACQuire:MODE
  NORMal;:CHANNEL1:ACCL:COUPLING
  GND<PMT>
  ```

- **When Concatenating Simple Headers**

  If a simple header follows another command, a colon (:) is placed in front of the simple header (cannot be omitted).

  Example
  ```
  :ACQuire:MODE
  NORMal;:STARt<PMT>
  ```

- **When Concatenating Common Commands**

  Common commands that are defined in the USBTMC-USB488 are independent of hierarchy. Colons (:) are not needed before a common command.

  Example
  ```
  :ACQuire:MODE NORMal;*CLS;
  TIME 0,0,0,0,500,0<PMT>
  ```

- **When Separating Commands with <PMT>**

  If a terminator is used to separate two commands, each command is a separate message. Therefore, the common header must be specified for each command even when commands belonging to the same command group are being concatenated.

  Example
  ```
  :ACQuire:MODE
  NORMal<PMT>:ACQuire:TIME
  0,0,0,0,500,0<PMT>
  ```

Upper-Level Query

An upper-level query is a query in which a question mark (`?`) is appended to the highest level command of a group. Execution of an upper-level query allows all settings that can be specified in the group to be received at once. Some query groups which are comprised of more than three hierarchical levels can output all the lower level settings.

The response to an upper-level query can be transmitted as a program message back to the instrument. In this way, the settings that existed when the upper-level query was made can be restored. However, some upper-level queries do not return setup information that is not currently in use. It is important to remember that not all the group's information is necessarily returned as part of a response.

Header Interpretation Rules

The instrument interprets the header that is received according to the rules below.

- Mnemonics are not case sensitive.

  Example "`CONTrol`" can also be written as "`control`" or "`Control`."

- The lower-case section of the header can be omitted.

  Example "`CONTrol`" can also be written as "`CONTR`" or "`CONT`."

- The question mark (`?`) at the end of a header indicates that it is a query. The question mark (`?`) cannot be omitted.

  Example The shortest abbreviation for `CONTrol?` is `CONT?`.

- If the <x> (value) at the end of a mnemonic is omitted, it is interpreted as a 1.

  Example If "`CHANnel<x>`" is written as "`CHAN`," it means "`CHANnel1`."

- **The section enclosed by braces (`[]`) can be omitted.**

  Example `CHANnel1[:VOLTage]:COUPling` can also be written as `CHAN1:COUP`.
  However, the last section enclosed by braces (`[]`) cannot be omitted in an upper-level query.

  Example "`CHANnel1?`" and "`CHANnel1:COUPling?`" are different queries.

2.3 Response

Response

When the controller sends a message unit that has a question mark (?) in its program header (query), the instrument returns a response message to the query. A response message is returned in one of the following two forms.

- **Response Consisting of a Header and Data**
  If the response can be used as a program message without any change, it is returned with a command header attached.

  Example
  ```
  :ACQUire:MODE?<PMT>
  -> :ACQUire:MODE NORMAL<RMT>
  ```

- **Response Consisting of Data Only**
  If the response cannot be used as a program message unless changes are made to it (query-only command), only the data section is returned. However, there are query-only commands that return responses with the header attached.

When You Wish to Return a Response without a Header

Responses that return both header and data can be set so that only the data section is returned. The "COMMunicate:HEADer" command is used to do this.

Abbreviated Form

Normally, the lower-case section is removed from a response header before the response is returned to the controller. Naturally, the full form of the header can also be used. For this, the "COMMunicate:VERBose" command is used. The sections enclosed by braces ([]) are also omitted in the abbreviated form.

2.4 Data

Data

A data section comes after the header. A space must be included between the header and the data. The data contains conditions and values. Data is classified as below.

Data	Meaning
<Decimal>	A value expressed as a decimal number (Example: Probe attenuation of CH1 `-> CHANnel1:PROBe 100`)
<Voltage><Time> <Frequency> <Current>	A physical value (Example: Time axis range `-> TIMebase:TDIV 1US`)
<Register>	Register value expressed as binary, octal, decimal or hexadecimal. (Example: Extended event register value `-> STATUS:EESE #HFE`)
<Character Data>	Predefined character string (mnemonic). Can be selected from { }. (Example: Select the input coupling of CH1 `-> CHANnel1:COUPling {AC|DC|DC50|GND}`)
<Boolean>	Indicates ON and OFF. Set using ON, OFF or a value (Example: Turn ON the CH1 display `-> CHANnel1:DISPlay ON`)
<String data>	An arbitrary character string (Example: Comment to a screen data output `-> MATH1:UNIT:USERdefine "VOLT"`)
<Filename>	Indicates a file name. (Example: Save file name `-> FILE:SAVE:WAVeform:NAME "CASE1"`)
<Block data>	Arbitrary 8-bit data (Example: Response to acquired waveform data `-> #800000010ABCDEFGHIJ`)

<Decimal>

<Decimal> indicates a value expressed as a decimal number, as shown in the table below. Decimal values are given in the NR form as specified in the ANSI X3.42-1975.

Symbol	Meaning	Example		
<NR1>	Integer	125-1	+1000	
<NR2>	Fixed-point number	125.0	-.90	+001.
<NR3>	Floating-point number	125.0E+0	-9E-1	+.1E4
<NRf>	Any of the forms <NR1> to <NR3> is allowed.			

- The instrument can receive decimal values that are sent from the controller in any of the forms, <NR1> to <NR3>. This is represented by <NRf>.
- For response messages that the instrument returns to the controller, the form (<NR1> to <NR3> to be used) is determined by the query. The same form is used regardless of the size of the value.
- For the <NR3> format, the "+" sign after the "E" can be omitted. However, the "-" sign cannot be omitted.
- If a value outside the setting range is entered, the value is normalized so that it is just inside the range.
- If a value has more significant digits than the available resolution, the value is rounded.

<Voltage>, <Time>, <Frequency>, and <Current>

<Voltage>, <Time>, <Frequency>, and <Current> indicate decimal values that have physical significance. <Multiplier> or <Unit> can be attached to the <NRf> form that was described earlier. It is expressed in one of the following forms.

Form	Example
<NRf><Multiplier><Unit>	5MV
<NRf><Unit>	5E-3V
<NRf><Multiplier>	5M
<NRf>	5E-3

<Multiplier>

<Multipliers> which can be used are indicated below.

Symbol	Word	Multiplier
EX	Exa	10^{18}
PE	Peta	10^{15}
T	Tera	10^{12}
G	Giga	10^{9}
MA	Mega	10^{6}
K	Kilo	10^{3}
M	Milli	10^{-3}
U	Micro	10^{-6}
N	Nano	10^{-9}
P	Pico	10^{-12}
F	Femto	10^{-15}
A	Ato	10^{-18}

<Unit>

<Units> that can be used are indicated below.

Symbol	Word	Meaning
V	Volt	Voltage
S	Second	Time
HZ	Hertz	Frequency
MHZ	Megahertz	Frequency
A	Ampere	Current

- <Multiplier> and <Unit> are not case sensitive.
- "U" is used to indicate micro "μ".
- "MA" is used for Mega to distinguish it from Milli. The only exception is Megahertz which is expressed as "MHZ." Therefore, the "M (Milli)" multiplier cannot be used for frequencies.
- If both <Multiplier> and <Unit> are omitted, the default unit is used.
- Response messages are always expressed in the <NR3> form. Response messages are returned using the default unit without the <Multiplier> or <Unit>.

<Register>

<Register> indicates an integer, and can be expressed in hexadecimal, octal, or binary as well as a decimal number. It is used when each bit of the value has a particular meaning. It is expressed in one of the following forms.

Form	Example
<NRf>	1
#H<Hexadecimal value made up of the digits 0 to 9 and A to F>	#H0F
#Q<Octal value made up of the digits 0 to 7>	#Q777
#B<Binary value made up of the digits 0 and 1>	#B001100

- <Register> is not case sensitive.
- Response messages are always expressed as <NR1>.

<Character Data>

<Character Data> is a specified string of character data (a mnemonic). It is mainly used to indicate options and is chosen from the character strings given in { }. For interpretation rules, refer to “Header Interpretation Rules” on page 4-4.

Form	Example
{AC|DC|DC50|GND}	AC

- As with the header, the “COMMunicate:VERBose” command can be used to select whether to return the response in the full form or in the abbreviated form.
- The “COMMunicate:HEADer” setting does not affect the character data.

<Boolean>

<Boolean> is data that indicates ON or OFF. It is expressed in one of the following forms.

Form	Example
{ON|OFF|<NRf>}	ON OFF 1 0

- When <Boolean> is expressed in the <NRf> form, “OFF” is selected if the rounded integer value is 0, and ON for all other cases.
- A response message is always returned with a 1 if the value is ON and 0 if the value is OFF.

<String data>

<String data> is not a specified character string like <Character data>. It is an arbitrary character string. The character string must be enclosed in single quotation marks (') or double quotation marks (").

Form	Example
<String data>	'ABC' "IEEE488.2-1987"

- If a character string contains a double quotation mark ('), the double quotation mark is replaced by two double quotation marks (""). This rule also applies to a single quotation mark within a character string.
- A response message is always enclosed in double quotation marks (").
- <String data> is an arbitrary character string. Therefore the instrument assumes that the remaining program message units are part of the character string if no single (') or double quotation mark (") is encountered. As a result, no error is detected if a quotation mark is omitted.

<Filename>

<Filename> is data that indicates a file name. It is expressed in one of the following forms.

Form	Example
{<NRf>|<Character data>|<String data>}	1 CASE "CASE"

- <NRf> is rounded to an 8-digit integer and converted to ASCII code. The result is the file name (example: 1 becomes "00000001"). Negative values are not allowed.
- Response messages are always returned in the <String data> form.
- For <Character data>, the first 12 characters become the file name.
- For <String data>, the first 259 characters become the file name.
- For a description of the number of characters of the <String data> file name, see the *DL9500/DL9700 User’s Manual*.

<Block data>

<Block data> is arbitrary 8-bit data. It is only used in response messages on the DL9500/DL9700. Below is the syntax.

Form	Example
#N<N-digit decimal number> <Data byte sequence>	#800000010ABCDEFGHIJ

- #N
  Indicates that the data is <Block data>. “N” indicates the number of succeeding data bytes (digits) in ASCII code characters.
- <N-digit decimal number>
  Indicates the number of bytes of data (example: 00000010 = 10 bytes).
- <data byte sequence>
  Expresses the actual data (example: ABCDEFGHIJ).
- Data is comprised of 8-bit values (0 to 255). This means that the ASCII code “0AH,” which stands for “NL,” can also be a code used for data. Hence, care must be taken when programming the controller.

2.5 Synchronization with the Controller

Overlap Commands and Sequential Commands

There are two types of commands, overlap commands and sequential commands. In the case of overlap commands, the execution of the next command may start before the execution of the previous command is completed.
For example, if the next program message is transmitted when specifying the V/div value and querying the result, the response always returns the most recent setting (5 V in this case).

```
:CHANnel1:VDIV 5V;VDIV?<PMT>
```

This is because the next command is forced to wait until the processing of “CHANnel1:VDIV” itself is completed. This type of command is called a sequential command.
On the contrary, let us assume that you send the next program message when you wish to load a file and query the V/div value of the result.

```
:FILE:LOAD:SETup:EXECute "CASE1";:
CHANnel1:VDIV?
```

In this case, “CHANnel1:VDIV?” is executed before the loading of the file is completed, and the V/div value that is returned is the value before the file is loaded.
The act of executing the next command before the processing of itself is completed such as with “FILE:LOAD:SETup:EXECute "CASE1"” is called an overlap operation. A command that operates in this way is called an overlap command.
In such case, the overlap operation can be prevented by using the methods below.

Synchronizing with Overlap Commands

- **Using the *WAI Command**

  The *WAI command holds the subsequent commands until the overlap command is completed.

  Example
  ```
  :COMMunicate:OPSE #H0040;:
  FILE:LOAD:SETup:
  EXECute "CASE1";*WAI;:
  CHANnel1:VDIV?<PMT>
  ```

  “COMMunicate:OPSE” is a command used to select the “*WAI” target. Here, media access is specified. Because “*WAI” is executed immediately before “CHANnel1:VDIV?,” “CHANnel1:VDIV?” is not executed until the file loading is complete.

- **Using the COMMunicate:OVERlap command**

  The COMMunicate:OVERlap command enables (or disables) overlap operation.

  Example
  ```
  :COMMunicate:OVERlap #HFFBF;:
  FILE:LOAD:SETup:
  EXECute "CASE1";:CHANnel1:
  VDIV?<PMT>
  ```

  “COMMunicate:OVERlap #HFFBF” enables overlap operation on commands other than media access. Because the overlap operation of file loading is disabled, “FILE:LOAD:SETup:EXECute "CASE1"” operates in the same way as a sequential command. Therefore, CHANnel1:VDIV? is not executed until the file loading is complete.

- **Using the *OPC Command**

  The *OPC command sets the OPC bit, bit 0 of the standard event register (see page 4-4), to 1 when the overlap operation is completed.

  Example
  ```
  :COMMunicate:OPSE #H0040;
  *ESE 1;
  *ESR?;*SRE 32;:FILE:LOAD:SETup:
  EXECute "CASE1";*OPC<PMT>
  (Read the response to *ESR?)
  (Wait for a service request)
  :CHANnel1:VDIV?<PMT>
  ```

  “COMMunicate:OPSE” is a command used to select the “*OPC” target. Here, media access is specified.
  “*ESE 1” and “*SRE 32” indicate that a service request is generated only when the OPC bit is 1.
  “*ESR?” clears the standard event register.
  In the example above, “CHANnel1:VDIV?” is not executed until a service request is generated.

- **Using the *OPC? Query**

  The `*OPC?` query generates a response when an overlap operation is completed.

  Example
  ```
  :COMMunicate:OPSE #H0040;:FILE:
  LOAD:SETup:EXECute "CASE1";
  *OPC?<PMT>
  ```
  (Read the response to `*OPC?`)
  ```
  :CHANnel1:VDIV?<PMT>
  ```

  "`COMMunicate:OPSE`" is a command used to select the "`*OPC?`" target. Here, media access is specified. Because "`*OPC?`" does not generate a response until the overlap operation is completed, the loading of the file will have been completed by the time the response to "`*OPC?`" is read.

Note

Most commands are sequential commands. Overlap commands are indicated as overlap commands in chapter 3. All other commands are sequential commands.

Achieving Synchronization without Using Overlap Commands

Even for sequential commands, synchronization is sometimes required for non communication-related reasons such as a trigger occurrence.

For example, if the next program message is transmitted to make an inquiry about the waveform data which has been acquired with the trigger mode set to single, the `WAVeform:SEND?` command may be executed regardless of whether the acquisition has been completed or not and may result in command execution error.

```
:TRIGger:MODE SINGle;:STARt;:WAVeform:
SEND?<PMT>
```

In this case, the following method must be used to synchronize with the end of the acquisition.

- **Using the STATus:CONDition? Query**

  The "`STATus:CONDition?`" query is used to query the contents of the condition register (page 4-5). Whether waveforms are being retrieved can be determined by reading bit 0 of the condition register. If bit 0 of the condition register is "1," waveforms are being retrieved. Otherwise, it is stopped.

  Example
  ```
  :TRIGger:MODE SINGle;:STARt<PMT>
  :STATus:CONDition?<PMT>
  ```
  (Read the response. If bit 0 is 1, repeat this command until it becomes 1.)
  ```
  :WAVeform:SEND?<PMT>
  ```

  The `WAVeform:SEND?` command will not be executed until bit 0 of the condition register is set to "`0`."

- **Using the Extended Event Register**

  The changes in the condition register can be reflected in the extended event register (page 4-5).

  Example
  ```
  :STATus:FILTer1 FALL;:
  STATus:EESE 1;EESR?;*SRE 8;:
  TRIGger:MODE SINGle;:STARt<PMT>
  ```
  (Read the response to `STATus:EESR?`)
  (Wait for a service request)
  ```
  :WAVeform:SEND?<PMT>
  ```

  The "STATus:FILTer1 FALL" command sets the transition filter so that bit 0 (FILTer1) of the extended event register is set to 1 when bit 0 of the condition register changes from 1 to 0.

  The "`:STATus:EESE 1`" command is used to reflect only bit 0 of the extended event register to the status byte.
  The "`STATus:EESR?`" command is used to clear the extended event register.
  The "`*SRE 8`" command is used to generate a service request solely on the cause of the extended event register.
  The "`WAVeform:SEND?`" command is not executed until a service request is generated.

- **Using the COMMunicate:WAIT Command**

  The "`COMMunicate:WAIT`" command halts communications until a specific event is generated.

  Example
  ```
  :STATus:FILTer1 FALL;:
  STATus:EESR?;:TRIGger:
  MODE SINGle<PMT>
  ```
  (Read the response to `STATus:EESR?`)
  ```
  :COMMunicate:WAIT 1;:WAVeform:
  SEND?<PMT>
  ```

  For a description of "`STATus:FILTer1 FALL`" and "`STATus:EESR?`" see the previous section regarding the extended event register.
  The "`COMMunicate:WAIT 1`" command indicates that the program will wait for bit 0 of the extended event register to be set to "`1`."
  The `WAVeform:SEND?` command will not be executed until bit 0 of the extended event register is set to "`1`."

3.1 Notes on Use of Communication Commands

When using communication commands, please note the following.

- The following functions of the SL1000 High-Speed Data Acquisition Unit are not available when using communication commands.
    - Synchronized operation of multiple units (the API can be used for synchronized operation)
    - Automatic recording to PC
- Reading of data per WAVeform group (only possible while measurement is stopped).
- At least one module must be set to measurement group 1 (using the: TIMebase:MODUle<x>:GROUp command).
- The initial value of the: CHANnel<x>:DISPlay command that turns measurement ON/OFF on individual channels is OFF. Turn this setting ON when using the command for the first time.
- When using the MRECord group commands for auto recording, the initial value for the recording destination is “PC.” When auto-recording, you must change the recording destination to HDD.

3.2 List of Commands

Command	Function	Page

ACQuire Group

Command	Function	Page
:ACQuire?	Queries all settings related to waveform acquisition.	3-12
:ACQuire:CLOCk	Sets or queries the time base (internal/external clock).	3-12
:ACQuire:COUNt	Sets or queries the waveform acquisition count for normal mode.	3-12
:ACQuire:ECLock? (Ext CLock)	Sets or queries all settings related to the external sample clock.	3-12
:ACQuire:ECLock:PCOunt (Ext CLock Pretrigger COunt)	Sets or queries the pre-trigger count when using the external sample clock.	3-12
:ACQuire:ECLock:COUNt (Ext CLock COUNt)	Sets or queries the sample count when using the external sample clock.	3-12
:ACQuire:MMODe (Motion MODe)	Sets or queries the waveform acquisition operation mode.	3-12
:ACQuire:MODE	Sets or queries the waveform acquisition mode.	3-12
:ACQuire:TIME	Sets or queries the measurement time.	3-12
:ACQuire:PROTate	Sets or queries the Pulse/Rotate setting when inputting the external clock.	3-12
:ACQuire:MHNum?	Queries the maximum history number for trigger measurement mode.	3-13

ALARm Group

Command	Function	Page
:ALARm?	Queries all settings related to alarms.	3-14
:ALARm:ACK:EXECute	Clears alarm output.	3-14
:ALARm:ACOunt?	Queries the number of acquisitions, which is counted from the start of measurement, when an alarm occurs.	3-14
:ALARm:COMBination	Sets or queries the AND/OR state of the alarms of each channel.	3-14
:ALARm:CONDition?	Queries the alarm output terminal condition.	3-14
:ALARm:CHANnel<x>?	Queries all settings related to channel alarms.	3-14
:ALARm:CHANnel<x>:CONDition?	Queries the alarm condition of the specified channel.	3-14
:ALARm:CHANnel<x1>:HYSTeresis<x2>	Sets or queries the alarm hysteresis of a channel.	3-14
:ALARm:CHANnel<x1>:LEVel<x2>	Sets or queries the alarm level of a channel (when the input of the specified channel is not logic).	3-14
:ALARm:CHANnel<x>:TYPE	Sets or queries the alarm type of a channel.	3-14
:ALARm:CHANnel<x>:AVALue? (Alarm VALue)	Queries the measured value at the alarm occurrence on the specified channels as an ASCII string.	3-15
:ALARm:CTIMe?	Queries the time of the most recent channel alarm condition change.	3-15
:ALARm:HOLD	Sets or queries the alarm hold.	3-15
:ALARm:MODE	Sets or queries the alarm operation mode.	3-15
:ALARm:CMODe	Sets or queries the channel alarm operation mode.	3-15
:ALARm:SMODe	Sets or queries the system alarm operation mode.	3-15
:ALARm:OTERminal	Sets or queries the alarm output terminal on/off state.	3-15
:ALARm:SOURce	Sets or queries the alarm detection source.	3-15
:ALARm:STATus?	Queries the channel alarm status.	3-15
:ALARm:SSTatus?	Queries the system alarm status value.	3-16
:ALARm:SYSTem?	Queries all settings related to the system alarm.	3-16
:ALARm:SYSTem:SOURce?	Queries all settings related to system alarm detection.	3-16
:ALARm:SYSTem:SOURce:BOVerrun	Sets or queries system alarm buffer overrun detection.	3-16
:ALARm:SYSTem:SOURce:FSTop	Sets or queries system alarm fan stop detection.	3-16
:ALARm:SYSTem:SOURce:DFULl	Sets or queries system alarm HDD full detection.	3-16
:ALARm:SYSTem:SOURce:MRTime	Sets or queries the maximum recording time detection for free-run automatic recording mode.	3-16
:ALARm:STIMe?	Queries the time of the most recent system alarm condition change.	3-16

ASETup Group

Command	Function	Page
:ASETup?	Queries all settings related to auto setup.	3-17
:ASETup:EXECute	Execute auto setup.	3-17
:ASETup:TARGet	Sets or queries the target channel for auto setup.	3-17
:ASETup:UNDO	Cancels auto setup.	3-17

Command	Function	Page

CALibrate Group

Command	Function	Page
:CALibrate?	Queries all settings related to calibration.	3-18
:CALibrate[:EXECute]	Executes calibration.	3-18
:CALibrate:MODE	Sets or queries the ON/OFF state of auto calibration.	3-18

CHANnel Group

Command	Function	Page
:CHANnel<x>?	Queries all settings related to the vertical axis of the channel.	3-19
:CHANnel<x>:ACCL?	Queries all settings on the channel with the acceleration/voltage module installed.	3-19
:CHANnel<x>:ACCL:BIAS	Sets or queries the on/off status of the acceleration sensor’s bias current when an Acceleration/Voltage Module is installed in the specified channel (slot).	3-19
:CHANnel<x>:ACCL:BWIDth	Sets or queries the filter when an Acceleration Voltage Module is installed in the specified channel (slot).	3-19
:CHANnel<x>:ACCL:COUPling	Sets or queries input coupling when an Acceleration/Voltage Module is installed in the specified channel (slot)	3-19
:CHANnel<x>:ACCL:GAIN	Sets or queries the gain when an Acceleration/Voltage Module is installed in the specified channel (slot).	3-19
:CHANnel<x>:ACCL:SENSitivity	Sets or queries the sensitivity when an Acceleration/Voltage Module is installed in the specified channel (slot).	3-19
:CHANnel<x>:ACCL:UNIT	Sets or queries the unit of measurement of the upper and lower limit values when an Acceleration/Voltage Module is installed in the specified channel(slot).	3-20
:CHANnel<x>:ASET?	Queries whether the specified channel is able to set an auto setup or not.	3-20
:CHANnel<x>:DISPlay	Sets or queries the channel acquisition ON/OFF state.	3-20
:CHANnel<x>:FREQ?	Queries all settings when a frequency module is installed in the specified channel (slot).	3-20
:CHANnel<x>:FREQ:INPut?	Queries all settings related to the input when a frequency module is installed in the specified channel (slot).	3-20
:CHANnel<x>:FREQ:INPut:BWIDth	Sets or queries the bandwidth limit when a frequency module is installed in the specified channel (slot).	3-20
:CHANnel<x>:FREQ:INPut:CELimination	Sets or queries chattering elimination when a frequency module is installed in the specified channel (slot).	3-20
:CHANnel<x>:FREQ:INPut:COUPling	Sets or queries input coupling when a frequency module is installed in the specified channel (slot).	3-20
:CHANnel<x>:FREQ:INPut:HYSTeresis	Sets or queries hysteresis when a frequency module is installed in the specified channel (slot).	3-20
:CHANnel<x>:FREQ:INPut:PRESet	Sets or queries the preset when a frequency module is installed in the specified channel (slot).	3-21
:CHANnel<x>:FREQ:INPut:PROBe	Sets or queries the probe attenuation when a frequency module is installed in the specified channel (slot).	3-21
:CHANnel<x>:FREQ:INPut:PULLup	Sets or queries the pull-up on/off state when a frequency module is installed in the specified channel (slot).	3-21
:CHANnel<x>:FREQ:INPut:SLOPe	Sets or queries the slope when a frequency module is installed in the specified channel (slot).	3-21
:CHANnel<x>:FREQ:INPut:THReshold	Sets or queries the threshold level when a frequency module is installed in the specified channel (slot).	3-21
:CHANnel<x>:FREQ:INPut:VRANGe	Sets or queries the voltage range when a frequency module is installed in the specified channel (slot).	3-21
:CHANnel<x>:FREQ:LSCale?	Queries all settings related to linear scaling when a frequency module is installed in the specified channel (slot).	3-21
:CHANnel<x>:FREQ:LSCale:AVALue	Sets or queries linear scaling coefficient A when a frequency module is installed in the specified channel (slot).	3-21
:CHANnel<x>:FREQ:LSCale:BVALue	Sets or queries linear scaling coefficient B when a frequency module is installed in the specified channel (slot).	3-22
:CHANnel<x>:FREQ:LSCale:GETMeasure	Measures the X values of P1 and P2 for linear scaling when a frequency module is installed in the specified channel (slot).	3-22
:CHANnel<x>:FREQ:LSCale:MODE	Sets or queries linear scaling when a frequency module is installed in the specified channel (slot).	3-22
:CHANnel<x>:FREQ:LSCale:{P1X|P1Y|P2X|P2Y}	Sets or queries the X or Y value of P1 or P2 for linear scaling when a frequency module is installed in the specified channel (slot).	3-22
:CHANnel<x>:FREQ:LSCale:UNIT	Sets or queries the unit of measurement to attach to the result of linear scaling when a frequency module is installed in the specified channel (slot).	3-22

3.2 List of Commands

Command	Function	Page
:CHANnel<x>:FREQ:OFFSet	Sets or queries the offset value when a frequency module is installed in the specified channel (slot).	3-22
:CHANnel<x>:FREQ:SETup?	Queries all settings related to FV setup when a frequency module is installed in the specified channel (slot).	3-22
:CHANnel<x>:FREQ:SETup:CFRequency	Sets or queries the center frequency when a frequency module is installed in the specified channel (slot).	3-23
:CHANnel<x>:FREQ:SETup:DECeleration	Sets or queries the on/off state of deceleration prediction when a frequency module is installed in the specified channel (slot).	3-23
:CHANnel<x>:FREQ:SETup:DPULse	Sets or queries the distance per pulse when a frequency module is installed in the specified channel (slot).	3-23
:CHANnel<x>:FREQ:SETup:FILTer?	Queries all settings related to the filter when a frequency module is installed in the specified channel (slot).	3-23
:CHANnel<x>:FREQ:SETup:FILTer:SMOothing?	Queries all settings related to smoothing when a frequency module is installed in the specified channel (slot).	3-23
:CHANnel<x>:FREQ:SETup:FILTer:SMOothing:MODE	Sets or queries the on/off state of smoothing when a frequency module is installed in the specified channel (slot).	3-23
:CHANnel<x>:FREQ:SETup:FILTer:SMOothing:VALue	Sets or queries the moving average order of smoothing when a frequency module is installed in the specified channel (slot).	3-23
:CHANnel<x>:FREQ:SETup:FILTer:PAVerage?	Queries all settings related to pulse average when a frequency module is installed in the specified channel (slot).	3-23
:CHANnel<x>:FREQ:SETup:FILTer:PAVerage:MODE	Sets or queries the on/off state of pulse average mode when a frequency module is installed in the specified channel (slot).	3-24
:CHANnel<x>:FREQ:SETup:FILTer:PAVerage:VALue	Sets or queries the pulse average count when a frequency module is installed in the specified channel (slot).	3-24
:CHANnel<x>:FREQ:SETup:FUNCtion	Sets or queries the measuring mode when a frequency module is installed in the specified channel (slot).	3-24
:CHANnel<x>:FREQ:SETup:LRESet	Sets or queries the over limit reset when a frequency module is installed in the specified channel (slot).	3-24
:CHANnel<x>:FREQ:SETup:MPULse	Sets or queries whether the measurement pulse is positive or negative when a frequency module is installed in the specified channel (slot).	3-24
:CHANnel<x>:FREQ:SETup:PROTate	Sets or queries the number of pulses per rotation when a frequency module is installed in the specified channel (slot).	3-25
:CHANnel<x>:FREQ:SETup:RESet	Resets the pulse count when a frequency module is installed in the specified channel (slot)	3-25
:CHANnel<x>:FREQ:SETup:STOPpredict	Sets or queries the on/off state of stop prediction when a frequency module is installed in the specified channel (slot).	3-25
:CHANnel<x>:FREQ:SETup:TUNit	Sets or queries the time unit when a frequency module is installed in the specified channel (slot).	3-25
:CHANnel<x>:FREQ:SETup:UNIT	Sets or queries the pulse integration unit when a frequency module is installed in the specified channel (slot).	3-25
:CHANnel<x>:FREQ:SETup:UPULse	Sets or queries the unit/pulse when a frequency module is installed in the specified channel (slot).	3-25
:CHANnel<x>:FREQ:SETup:VUNit	Sets or queries the unit of velocity when a frequency module is installed in the specified channel (slot).	3-25
:CHANnel<x>:FREQ:VDIV	Sets or queries the Value/Div when a frequency module is installed in the specified channel (slot).	3-25
:CHANnel<x>:LABel	Sets or queries the waveform label of the specified channel.	3-26
:CHANnel<x>:MODule?	Queries the module installed in the channel (slot).	3-26
:CHANnel<x>:RECord	Sets or queries whether to record the specified channel or not.	3-26
:CHANnel<x>:STRain?	Queries all settings related to the Strain Module when a strain module is installed in the specified channel (slot).	3-26
:CHANnel<x>:STRain:BALance?	Queries the balance setting when a strain module is installed in the specified channel (slot).	3-26
:CHANnel<x>:STRain:BALance:CHANnel<x>	Sets or queries the channel on which balancing is to be executed when a strain module is installed in the specified channel (slot).	3-26
:CHANnel<x>:STRain:BALance:EXECute	Balances strain when a strain module is installed in the specified channel (slot).	3-26
:CHANnel<x>:STRain:BWIDth	Sets or queries the filter when a strain module is installed in the specified channel (slot).	3-26
:CHANnel<x>:STRain:EXCitation	Sets or queries the bridge voltage when a strain module is installed in the specified channel (slot).	3-27
:CHANnel<x>:STRain:GFACtor	Sets or queries the gauge factor when a strain module is installed in the specified channel (slot).	3-27
:CHANnel<x>:STRain:INVert	Sets or queries whether or not the display is inverted when a strain module is installed in the specified channel (slot).	3-27

Command	Function	Page
:CHANnel<x>:STRain:LSCale?	Queries all settings related to linear scaling when a strain module is installed in the specified channel (slot).	3-27
:CHANnel<x>:STRain:LSCale:AVALue	Sets or queries linear scaling coefficient A when a strain module is installed in the specified channel (slot).	3-27
:CHANnel<x>:STRain:LSCale:BVALue	Sets or queries offset value B when a strain module is installed in the specified channel (slot).	3-27
:CHANnel<x>:STRain:LSCale:DISPlaytype?	Queries all settings related to the display type when using linear scaling.	3-27
:CHANnel<x>:STRain:LSCale:DISPlaytype:MODE	Sets or queries the display format for linear scaling.	3-27
:CHANnel<x>:STRain:LSCale:DISPlaytype:DECimalnum	Sets or queries the decimal place when the display format for linear scaling is set to Floating.	3-28
:CHANnel<x>:STRain:LSCale:GETMeasure	Measures the X values of P1 and P2 for linear scaling when a strain module is installed in the specified channel (slot).	3-28
:CHANnel<x>:STRain:LSCale:MODE	Sets or queries the linear scaling method when a strain module is installed in the specified channel (slot). (The method can only be set to SHUNt with a strain module with DSUB, Shunt-Cal.)	3-28
:CHANnel<x>:STRain:LSCale:{P1X|P1Y|P2X|P2Y}	Sets or queries the X or Y value of P1 or P2 for linear scaling when a strain module is installed in the specified channel (slot).	3-28
:CHANnel<x>:STRain:LSCale:SHUNt	Executes shunt calibration when a strain module is installed in the specified channel (slot). (This command only works with a strain module with DSUB, Shunt-Cal.)	3-28
:CHANnel<x>:STRain:LSCale:UNIT	Sets or queries the unit of measurement to attach to the result of linear scaling when a strain module is installed in the specified channel (slot).	3-28
:CHANnel<x>:STRain:RANGe	Sets or queries the measuring range when a strain module is installed in the specified channel (slot).	3-28
:CHANnel<x>:STRain:UNIT	Sets or queries the unit of measurement when a strain module is installed in the specified channel (slot).	3-29
:CHANnel<x>:TEMPerature?	Queries all settings when a temperature, high precision voltage module is installed in the specified channel (slot).	3-29
:CHANnel<x>:TEMPerature:BURNout	Sets or queries whether or not burnout is detected when a temperature, high precision voltage isolation module is installed in the specified channel (slot).	3-29
:CHANnel<x>:TEMPerature:BWIDth	Sets or queries the bandwidth limit when a temperature, high precision, voltage nodule is installed in the specified channel (slot).	3-29
:CHANnel<x>:TEMPerature:COUPling	Sets or queries input coupling when a temperature, high precision voltage module is installed in the specified channel (slot).	3-29
:CHANnel<x>:TEMPerature:RJC	Sets or queries the RJC when a temperature, high precision voltage module is installed in the specified channel (slot).	3-29
:CHANnel<x>:TEMPerature:TYPE	Sets or queries the thermocouple type when a temperature, high precision voltage module is installed in the specified channel (slot).	3-29
:CHANnel<x>:TEMPerature:UNIT	Sets or queries the unit of measurement values when a temperature, high precision voltage module is installed in the specified channel (slot).	3-29
:CHANnel<x>:UNIT?	Queries the unit added to the channel.	3-30
:CHANnel<x>:VOLTage?	Queries all settings when a voltage module is installed in the specified channel (slot).	3-30
:CHANnel<x>[:VOLTage]:BWIDth	Sets or queries the bandwidth limit when a temperature, high precision voltage isolation module is installed in the specified channel (slot).	3-30
:CHANnel<x>[:VOLTage]:COUPling	Sets or queries input coupling when a voltage module is installed in the specified channel (slot).	3-30
:CHANnel<x>[:VOLTage]:INVert	Sets or queries whether or not the display is inverted when a voltage module is installed in the specified channel (slot).	3-30
:CHANnel<x>[:VOLTage]:LSCale?	Queries all settings related to linear scaling when a voltage module is installed in the specified channel (slot).	3-30
:CHANnel<x>[:VOLTage]:LSCale:AVALue	Sets or queries scaling coefficient A of linear scaling when a voltage module is installed in the specified channel (slot).	3-30
:CHANnel<x>[:VOLTage]:LSCale:BVALue	Sets or queries linear scaling offset value B when a voltage module is installed in the specified channel (slot).	3-31
:CHANnel<x>[:VOLTage]:LSCale:DISPlaytype?	Queries all settings related to the display type when using linear scaling.	3-31
:CHANnel<x>[:VOLTage]:LSCale:DISPlaytype:MODE	Sets or queries the display format for linear scaling.	3-31
:CHANnel<x>[:VOLTage]:LSCale:DISPlaytype:DECimalnum	Sets or queries the decimal place when the display format for linear scaling is set to Floating.	3-31
:CHANnel<x>[:VOLTage]:LSCale:DISPlaytype:SUBunit	Sets or queries the sub unit when the display format for linear scaling is set to Floating.	3-31

Command	Function	Page
:CHANnel<x>[:VOLTage]:LSCale:GETMeasure	Measures the X values of P1 and P2 for linear scaling when a voltage module is installed in the specified channel (slot).	3-31
:CHANnel<x>[:VOLTage]:LSCale:MODE	Sets or queries linear scaling when a voltage module is installed in the specified channel (slot).	3-31
:CHANnel<x>[:VOLTage]:LSCale:{P1X|P1Y|P2X|P2Y}	Sets or queries the X or Y value of P1 or P2 for linear scaling when a voltage module is installed in the specified channel (slot).	3-32
:CHANnel<x>[:VOLTage]:LSCale:UNIT	Sets or queries the unit of measurement to attach to the result of linear scaling when a voltage module is installed in the specified channel (slot).	3-32
:CHANnel<x>[:VOLTage]:PROBe	Sets or queries the probe type when a voltage module is installed in the specified channel (slot).	3-32
:CHANnel<x>[:VOLTage]:VDIV	Sets or queries the V/div value when a voltage module is installed in the specified channel (slot).	3-32

COMMunicate Group

:COMMunicate?	Queries all settings related to communications.	3-33
:COMMunicate:HEADer	Sets or queries whether or not to add a header to responses to queries.	3-33
:COMMunicate:LOCKout	Sets or clears local lockout.	3-33
:COMMunicate:OPSE	Sets or queries the overlap command that is used by the *OPC, *OPC? and *WAI commands.	3-33
:COMMunicate:OPSR?	Queries the value of the operation pending status register.	3-33
:COMMunicate:OVERlap	Sets or queries the commands to operate as overlap commands.	3-33
:COMMunicate:REMote	Sets remote or local. ON is remote mode	3-33
:COMMunicate:STATus?	Queries the line-specific status.	3-33
:COMMunicate:VERBose	Sets or queries whether to return the response to a query using full spelling or abbreviations.	3-33
:COMMunicate:WAIT	Waits for one of the specified extended events to occur.	3-33
:COMMunicate:WAIT?	Creates the response that is returned when the specified event occurs.	3-34

CONTrol Group

:CONTrol?	Queries all settings related to the station.	3-35
:CONTrol:FREE?	Queries all settings related to free-run measurement.	3-35
:CONTrol:FREE:LATCh	Executes the latch for the measurement data access for free-run measurement. (Unit firmware holds 'Acq memory writing address' of measurement group 1 data.)	3-35
:CONTrol:FREE:LCOunt?	Queries the sample counts, which are counted from the start of measurement, at the latch point for free-run measurement.	3-35
:CONTrol:FREE:LENGth<x>?	Queries the measured efficient data counts at the latch point for free-run measurement.	3-35
:CONTrol:FREE:STIMe?	Queries the start time of measurement for free-run measurement.	3-35
:CONTrol:FREE:GDELay<x>?	Queries the delay between the measurement start points of each measurement group.	3-35
:CONTrol:HDCapacity?	Queries the total capacity of internal HDD.	3-35
:CONTrol:HDFRee?	Queries the amount of free space in the internal HDD.	3-35

DATa Group

:DATa	Queries all settings related to waveform data.	3-36
:DATa:BYTeorder	Sets or queries the transmission order when using word format of two bytes or more.	3-36
:DATa:FRAW?	Queries the specified waveform data during free-run measurement The acquisition start point and the acquisition count are specified by the count of the measurement groups to which a specified trace belongs.	3-36

ETHernet Group

:ETHernet?	Queries all settings related to the network.	3-37
:ETHernet:TCP?	Queries all setting related to TCP.	3-37
:ETHernet:TCPip:DHCP	Sets or queries DHCP.	3-37
:ETHernet:TCPip:GATeway	Sets or queries default gateway.	3-37
:ETHernet:TCPip:IPADdress	Sets or queries the IP address.	3-37
:ETHernet:TCPip:NETMask	Sets or queries the subnet mask.	3-37

Command	Function	Page

FILE Group

Command	Function	Page
:FILE?	Queries all settings related to the internal hard disk.	3-38
:FILE:DELete	Deletes files.	3-38
:FILE:DIRectory?	Queries all settings related to the directory of the storage media.	3-38
:FILE[:DIRectory]:CDIRectory	Changes the current directory of the storage media.	3-38
:FILE[:DIRectory]:DRIVe	Sets the storage media to be controlled.	3-38
:FILE[:DIRectory]:FREE?	Queries the free disk space (bytes) on the target media.	3-38
:FILE[:DIRectory]:RMDirectory (Remove Directory)	Deletes the specified directory in the count directory. This is an overlap command.	3-38
:FILE[:DIRectory]:MDIRectory	Creates a new directory in the current directory.	3-38
:FILE[:DIRectory]:PATH?	Queries the current directory.	3-38
:FILE:LOAD:SETup:ABORt	Aborts loading of data.	3-38
:FILE:LOAD:SETup[:EXECute]	Loads data. This is an overlap command.	3-38
:FILE:SAVE?	Queries all settings related to the saving of files.	3-38
:FILE:SAVE:ANAMing	Sets or queries the auto naming function for saved files.	3-38
:FILE:SAVE:AREA:MODE	Sets or queries the specify save area function.	3-38
:FILE:SAVE:AREA:COUNt	Sets or queries the save area.	3-39
:FILE:SAVE:BINary?	Queries all settings related to the saving of waveform data.	3-39
:FILE:SAVE:{BINary|MEASure|SETup}:ABORt	Aborts the saving of data.	3-39
:FILE:SAVE:BINary:ACOunt	Sets or queries the target acquisition count when saving files.	3-39
:FILE:SAVE:{BINary|MEASure|SETup}[:EXECute]	Executes the saving of data to a file. This is an overlap command.	3-39
:FILE:SAVE:BINary:HISTory	Sets or queries the save target of the history memory of the data.	3-39
:FILE:SAVE:BINary:TALL (Trace All)	Sets or queries the selection method of the trace when saving files.	3-39
:FILE:SAVE:COMMent	Sets or queries the comment of data to be saved.	3-39
:FILE:SAVE:NAME	Sets or queries the name of the file to be saved.	3-39

GONogo Group

Command	Function	Page
:GONogo?	Queries all settings related to GO/NO-GO judgment.	3-40
:GONogo:ACONdition	Sets or queries the GO/NO-GO judgment action condition.	3-40
:GONogo:ACTion?	Queries all settings related to the action taken when the execution condition is met.	3-40
:GONogo:ACTion:BUZZer	Sets or queries whether or not a beep is sounded when the condition is met.	3-40
:GONogo:ACTion:SAVE[:MODE]	Sets or queries whether or not waveform data is saved to the storage media when the condition is met.	3-40
:GONogo:AREA	Sets or queries the waveform area that is judged.	3-40
:GONogo:COUNt?	Queries the number of performed GO/NO-GO judgments.	3-40
:GONogo:RSTatus?	Queries the most recent GO/NO-GO judgment.	3-40
:GONogo:LOGic	Sets or queries the GO/NO-GO logical condition.	3-40
:GONogo:MODE	Sets or queries the GO/NO-GO judgment mode.	3-40
:GONogo:NGCount?	Queries the GO/NO-GO judgment NO-GO count.	3-40
:GONogo:PARameter?	Queries all settings related to parameter judgment.	3-40
:GONogo:PARameter:ITEM<x>?	Queries all settings related to waveform parameters of the parameter judgment.	3-40
:GONogo:PARameter:ITEM<x>:CAUSe?	Queries whether or not the specified waveform parameter is the cause of a NO-GO judgment.	3-40
:GONogo:PARameter:ITEM<x>:MODE	Sets or queries the judgment criteria of the specified waveform parameter of the parameter judgment.	3-41
:GONogo:PARameter:ITEM<x>:TRACe	Sets or queries the target waveform of the measurement of the specified waveform parameters of the parameter judgment.	3-41
:GONogo:PARameter:ITEM<x>:TYPE?	Queries the item and the upper and lower limits of the measurement of the specified waveform parameter of the parameter judgment.	3-41
:GONogo:PARameter:ITEM<x>:TYPE:<Parameter>	Sets or queries the upper and lower limits of the judgment area for the specified judgment condition.	3-41
:GONogo:PARameter:ITEM<x>:VALue?	Queries the automated measurement value of the specified GO/NO-GO judgment parameter.	3-41
:GONogo:PARameter:ITEM<x>:PARam?	Queries the measurement item of the specified waveform parameter.	3-41

Command	Function	Page

HISTory Group

:HISTory?	Queries all settings related to the history memory function.	3-42
:HISTory:CLEar	Clears all history memory data (all data in memory).	3-42
:HISTory:RECord? MINimum	Queries the minimum record number.	3-42
:HISTory:DATE?	Queries the trigger date of the target record number.	3-42
:HISTory:TIME?	Queries the trigger time of the target record number.	3-42

INITialize Group

:INITialize:EXECute	Initializes settings.	3-43
:INITialize:UNDO	Undoes the initialization of settings.	3-43

MEASure Group

:MEASure?	Queries all settings related to the automated measurement of waveform parameters.	3-44
:MEASure:CHANnel<x>?	Queries the On/Off state of all of the waveform parameters of the specified channel.	3-44
:MEASure:AREA	Sets or queries the automatically measured waveform area for the waveform parameters.	3-44
:MEASure:MODE	Sets or queries the auto measurement mode of the waveform parameter.	3-44
:MEASure:CRANge (Count Range)	Sets or queries the waveform parameter measurement range.	3-44
:MEASure:CHANnel<x>:METHod	Sets or queries the high/low point setting method.	3-44
:MEASure:CHANnel<x>:DPRoximal?	Queries all settings related to distal, mesial, and proximal.	3-44
:MEASure:CHANnel<x>:DPRoximal:MODE	Sets or queries the distal, mesial, and proximal point mode setting.	3-44
:MEASure:CHANnel<x>:DPRoximal:PERCent	Sets or queries the distal, mesial, and proximal points as percentages.	3-44
:MEASure:CHANnel<x>:DPRoximal:UNIT	Sets or queries the distal, mesial, and proximal points.	3-45
:MEASure:CHANnel<x>:<Parameter>?	Queries all settings related to the specified waveform parameter (measurement item).	3-45
:MEASure:CHANnel<x>:<Parameter>:STATe	Sets or queries the on/off state of the measurement of the specified waveform parameter.	3-45
:MEASure:CHANnel<x>:<Parameter>:VALue?	Queries the measured value of the specified waveform parameter.	3-45
:MEASure:FILE?	Queries all settings related to the file format output data of automatic measurement results.	3-45
:MEASure:FILE:TINFomation (Time Information)	Sets or queries the ON/OFF state of the addition of the trigger time information in the file format output data of automated measurement results.	3-45
:MEASure:FILE:SEND?	Executes the file format output of the automatic measurement results.	3-46

MONitor Group

:MONitor:ASENd?	Outputs the numeric monitor data (ASCII format) of all channels.	3-47
:MONitor:ASENd:CHANnel<x>?	Outputs the numeric monitor data (ASCII format) of the specified channel.	3-47
:MONitor:BITS:CHANnel<x>?	Queries the valid bit length of the specified channel.	3-47
:MONitor:BYTeorder	Sets or queries the transmission order when using word format of two bytes or more.	3-47
:MONitor:FORMat:CHANnel<x>?	Sets or queries the format of the data to be transmitted .	3-47
:MONitor:OFFSet:CHANnel<x>?	Queries the offset value used to convert the numeric monitor data of the specified channel into physical values.	3-47
:MONitor:GAIN:CHANnel<x>?	Queries the gain value used to convert the numeric monitor data of the specified channel into physical values.	3-47
:MONitor:RANGe:CHANnel<x>?	Queries the range value used to convert the numeric monitor data of the specified channel into physical values.	3-48
:MONitor:SEND:{ALL|CHANnel<x>}?	Outputs the numeric monitor data in binary format.	3-48
:MONitor:VERBose	Sets or queries whether or not to add ‘label’ and ‘unit’ to the response format of MONitor:ASENd?.	3-48
:MONitor:LATCh:ASENd?	Outputs the numeric monitor data (ASCII format) of all channels at the latch.	3-48
:MONitor:LATCh:ASENd:CHANnel<x>?	Outputs the numeric monitor data (ASCII format) of the specified channels at the latch.	3-48
:MONitor:LATCh:EXECute	Latches the monitor data and the alarm data.	3-48

Command	Function	Page
:MONitor:LATCh:SEND:{ALL|CHANnel<x>}?	Outputs the numeric monitor data at the latch.	3-48
:MONitor:LATCh:ALARm:{ALL|CHANnel<x>}?	Outputs the channel alarm data at the latch.	3-48

MRECord Group

:MRECord?	Queries all settings related to automatic data recording.	3-49
:MRECord:STARt	Starts automatic data recording.	3-49
:MRECord:STOP	Aborts automatic data recording.	3-49
:MRECord:DESTination	Sets or queries the destination of the record of the automatic data recording to the internal media.	3-49
:MRECord:SCONdition (Start CONdition)	Sets or queries the start condition of automatic data recording.	3-49
:MRECord:STIMe	Sets or queries the start time if the start condition is the clock time.	3-49
:MRECord:ECONdition (End CONdition)	Sets or queries the end condition of automatic data recording.	3-49
:MRECord:ETIMe	Sets or queries the end time if the end condition is the clock time.	3-49
:MRECord:RTIMe	Sets or queries the recording time if the end condition is the recording time.	3-50
:MRECord:INTerval?	Queries all settings related to the recording interval.	3-50
:MRECord:INTerval:TIME	Sets or queries the recording interval if the recording interval mode is the TIME.	3-50
:MRECord:INTerval:MODE	Sets or queries the recording interval mode of automatic data recording.	3-50
:MRECord:COUNt	Sets or queries the number of recordings of automatic data recording.	3-50
:MRECord:AREA?	Queries all settings related to the method of the recording on the disk.	3-50
:MRECord:AREA:MODE	Sets or queries the method of recording (recording area) on the disk.	3-50
:MRECord:AREA:FNUMber	Sets or queries the method of recording (circular file number) on the disk.	3-50
:MRECord:ECLock?	Queries all settings related to recording when using the external sample clock.	3-50
:MRECord:ECLock:COUNt	Sets or queries the recording counts when using the external sample clock.	3-50
:MRECord:ECLock:INTerval	Sets or queries the recording interval counts when using the external sample clock.	3-50

MTRigger Group

:MTRigger	Activates manual triggering.	3-51

SELFtest Group

:SELFtest:HDD:EXECute?	Executes the self-test of the internal HDD and outputs the results.	3-52
:SELFtest:HDFormat	Formats the internal HDD.	3-52

SSTart Group

:SSTart	Executes single start.	3-53
:SSTart? {<NRf>}	Executes single start and waits for the completion.	3-53

STARt Group

:STARt	The STARt group is used to start waveform acquisition.	3-54

STATus Group

:STATus?	Queries all settings related to the communication status function.	3-55
:STATus:CONDition?	Queries the contents of the condition register.	3-55
:STATus:EESE	Sets or queries the contents of the extended event enable register.	3-55
:STATus:EESR?	Queries the contents of the extended event register and clears the register.	3-55
:STATus:ERRor?	Queries the error code and message information.	3-55
:STATus:FILTer<x>	Sets or queries the transition filter.	3-55
:STATus:QENable	Sets or queries whether or not to store messages other than errors to the Error queue (ON/OFF).	3-55
:STATus:QMESsage	Sets or queries whether or not to attach message information to the response to the “STATus:ERRor?” query (ON/OFF).	3-55

Command	Function	Page

STOP Group

:STOP	Stops waveform acquisition.	3-56

SYSTem Group

:SYSTem?	Queries all settings related to the system.	3-57
:SYSTem:CLOCk?	Queries all settings related to the date and time.	3-57
:SYSTem:CLOCk:DATE	Sets or queries the date.	3-57
:SYSTem:CLOCk:TIME	Sets or queries the time.	3-57
:SYSTem:KEYLock	Sets or queries the ON/OFF state of the keylock of the main unit.	3-57
:SYSTem:LCD?	Queries all settings related to the LCD.	3-57
:SYSTem:LCD:BRIGhtness	Sets or queries the brightness of the LCD.	3-57
:SYSTem:LCD:CONTrast	Sets or queries the contrast of the LCD.	3-57
:SYSTem:LCD:MODE	Sets or queries the ON/OFF state of the LCD backlight.	3-57
:SYSTem:LCD:DTOut (Display Time Out)	Sets or queries the ON/OFF state of the function that brings the contents of the LCD back to the specified screen in 30 seconds.	3-57
:SYSTem:LCD:DMODe (Display MODe)	Sets or queries the contents of LCD.	3-57
:SYSTem:RCMode (Remote Control Mode)	Sets or queries the measurement stop mode when the remote terminal is controlling START/STOP.	3-58
:SYSTem:STATion?	Queries all settings related to the unit.	3-58
:SYSTem:STATion:GNUMber?	Queries the area group number of the unit.	3-58
:SYSTem:STATion:MODule<x>?	Returns the attributes of the specified slot.	3-58
:SYSTem:STATion:NUMBer?	Queries the number of the unit.	3-58
:SYSTem:STATion:NAME	Sets or queries the name of the unit.	3-58
:SYSTem:STATion:GNAMe	Sets or queries the name of the group.	3-58

TIMebase Group

:TIMebase?	Queries all settings related to the time base.	3-59
:TIMebase:MODUle<x>:GROUp	Sets or queries the measurement group to which the specified module belongs.	3-59
:TIMebase:SOURce	Sets or queries the time base.	3-59
:TIMebase:SRATe	Sets or queries the sample rate of measurement group 1.	3-59
:TIMebase:GROUp<x>:SRATe	Sets or queries the sample rate of measurement groups 2 to 4.	3-59

TRIGger Group

:TRIGger?	Queries all settings related to triggers.	3-60
:TRIGger:COMBination?	Queries all settings related to the combination trigger class.	3-60
:TRIGger:COMBination:CHANnel<x>?	Queries the settings of the specified channel in the combination trigger class.	3-60
:TRIGger:COMBination:CHANnel<x1>:HYSTeresis<x2>	Sets or queries the trigger hysteresis of the specified channel in the combination trigger class.	3-60
:TRIGger:COMBination:CHANnel<x1>:LEVel<x2>	Sets or queries the trigger level of the specified channel in the combination trigger class.	3-60
:TRIGger:COMBination:CHANnel<x>:TYPE	Sets or queries the trigger type of the specified channel in the combination trigger class.	3-60
:TRIGger:COMBination:EXTernal:TYPE	Sets or queries the external trigger type of the specified channel in the combination trigger class.	3-60
:TRIGger:COMBination:MODE	Sets or queries the combination mode of the combination trigger class.	3-61
:TRIGger:DELay	Sets or queries the delay (time from the trigger point to the trigger position).	3-61
:TRIGger:HOLDoff?	Queries all settings related to the hold off.	3-61
:TRIGger:HOLDoff:TIME	Sets or queries the hold off time.	3-61
:TRIGger:MMODe	Sets or queries the waveform acquisition mode for trigger measurement.	3-61
:TRIGger:POSition	Sets or queries the trigger position.	3-61
:TRIGger[:SIMPle]?	Queries all settings related to the simple trigger.	3-61
:TRIGger[:SIMPle]:HYSTeresis	Sets or queries the hysteresis of the simple trigger.	3-61
:TRIGger[:SIMPle]:LEVel	Sets or queries the trigger level of the simple trigger of the specified channel.	3-61
:TRIGger[:SIMPle]:SLOPe	Sets or queries the simple trigger type of the channel specified.	3-61
:TRIGger:SIMPle:SOURce	Sets or queries the trigger source of the simple trigger.	3-62
:TRIGger:TIMer?	Queries all settings related to the timer trigger.	3-62
:TRIGger:TIMer:DATE	Sets or queries the date of the timer trigger.	3-62
:TRIGger:TIMer:INTerval	Sets or queries the trigger time interval of the timer trigger.	3-62

Command	Function	Page
:TRIGger:TIMer:TIME	Sets or queries the time of the timer trigger.	3-62
:TRIGger:TYPE	Sets or queries the trigger type.	3-62

WAVeform Group

:WAVeform?	Queries all settings related to waveform data.	3-64
:WAVeform:ACOunt?	Queries the acquisition count.	3-64
:WAVeform:BITS?	Queries the bit length of the waveform data specified by ":WAVeform:TRACe".	3-64
:WAVeform:BYTeorder	Sets or queries the byte order when using word format of two bytes or more.	3-64
:WAVeform:DIVision?	Queries the Division value used when converting the waveform data specified by 'WAVeform:TRACe' to physical values.	3-64
:WAVeform:END	Sets or queries the final data point of the waveform (main waveform) specified by 'WAVeform:TRACe'.	3-64
:WAVeform:FORMat	Sets or queries the format of the data to be transmitted.	3-64
:WAVeform:GAIN?	Queries the gain value used when converting the waveform data specified by 'WAVeform:TRACe' to physical values.	3-65
:WAVeform:HMAX? (History MAX)	Queries the maximum number of history that can be acquired by the currently specified unit.	3-65
:WAVeform:LENGth?	Queries all waveform data points (main side) specified by 'WAVeform:TRACe'.	3-65
:WAVeform:MODule?	Queries the module corresponding to the waveform specified by	3-65
:WAVeform:OFFSet?	Queries the offset value used when converting the waveform data specified by :WAVeform:TRACe to physical values.	3-65
:WAVeform:RANGe?	Queries the range value when converting the waveform data specified by :WAVeform:TRACe to physical values.	3-65
:WAVeform:RECord	Sets or queries the target record number of the main waveform for the commands in the WAVeform group.	3-65
:WAVeform:RECord? MINimum	Queries the minimum record number of the history (main waveform).	3-65
:WAVeform:SEND?	Queries the waveform data (raw data) specified by 'WAVeform:TRACe'.	3-65
:WAVeform:SIGN?	Queries the existence of a sign when querying the waveform data specified by :WAVeform:TRACe using block data.	3-66
:WAVeform:SRATe?	Queries the sample rate of the record specified by :WAVeform:RECord.	3-66
:WAVeform:STARt	Sets or queries the first data point of the waveform (main waveform) specified by 'WAVEform:TRACe'.	3-66
:WAVeform:TRACe	Sets or queries the target waveform.	3-66
:WAVeform:TRIGger?	Queries the trigger position of the record specified by :WAVeform:RECord.	3-66
:WAVeform:TYPE?	Queries the acquisition mode of the waveform specified by :WAVeform:TRACe.	3-66
:WAVeform:GDELay?	The delay of 'the measurement start point' or 'the trigger point' of the waveform specified by 'WAVeform:TRACe' is returned.	3-66

Common Command Group

*CAL?	Performs calibration and queries the result.	3-67
*CLS	Clears the standard event register, extended event register, and error queue.	3-67
*ESE	Sets the standard event enable register or queries the current setting.	3-67
*ESR?	Queries the standard event register and clears the register.	3-67
*IDN?	Queries the instrument model.	3-67
*OPC	Sets a "1" to bit 0 (OPC bit) of the standard event register bit upon the completion of the specified overlap command.	3-67
*OPC?	The specified overlap command is completed, ASCII code '1' is returned.	3-67
*OPT?	Queries the installed options.	3-67
*RST	Initializes the current settings.	3-67
*SRE	Sets or queries the service request enable register value.	3-68
*STB? (STatus Byte)	Queries the status byte register.	3-68
*TST?	Performs a self-test and queries the result.	3-68
*WAI	Holds the subsequent command until the completion of the specified overlap operation.	3-68

3.3 ACQuire Group

The ACQuire group deals with waveform acquisition.

:ACQuire?

Function Queries all settings related to waveform acquisition.

Syntax
```
:ACQuire?
```

:ACQuire:CLOCk

Function Sets or queries the time base (internal/external clock).

Syntax
```
:ACQuire:CLOCk {INTernal|EXTernal}
```

Example
```
:ACQUIRE:CLOCK INTERNAL
:ACQUIRE:CLOCK?
-> :ACQUIRE:CLOCK INTERNAL
```

:ACQuire:COUNt

Function Sets or queries the waveform acquisition count for normal mode.

Syntax
```
:ACQuire:COUNt {<NRf>|INFinity}
:ACQuire:COUNt?
```
<NRf> = 2 to 65536

Example
```
:ACQUIRE:COUNT 2
:ACQUIRE:COUNT? -> :ACQUIRE:COUNT 2
```

Description For the SL1000, the Single N count can also be set using this command

:ACQuire:ECLock? (Ext CLock)

Function Sets or queries all settings related to the external sample clock.

Syntax
```
:ACQuire:ECLock?
```

:ACQuire:ECLock:PCOunt (Ext CLock Pretrigger COunt)

Function Sets or queries the pre-trigger count when using the external sample clock.

Syntax
```
:ACQuire:ECLock:PCOunt {<NRf>}
:ACQuire:ECLock:PCOunt?
```
<NRf> = 0 to 134217727

Example
```
:ACQUIRE:ECLOCK:PCOUNT 1000
:ACQUIRE:ECLOCK:PCOUNT?
-> :ACQUIRE:ECLOCK:PCOUNT 1000
```

:ACQuire:ECLock:COUNt (Ext CLock COUNt)

Function Sets or queries the sample count when using the external sample clock.

Syntax
```
:ACQuire:ECLock:COUNt {<NRf>|MAX}
:ACQuire:ECLock:COUNt?
```
<NRf> = 2 to 134217728

Example
```
:ACQUIRE:ECLOCK:COUNT 1000
:ACQUIRE:ECLOCK:COUNT?
-> :ACQUIRE:ECLOCK:COUNT 1000
```

:ACQuire:MMODe (Motion MODe)

Function Sets or queries the waveform acquisition operation mode.

Syntax
```
:ACQuire:MMODe {FREerun|TRIGger}
:ACQuire:MMODe?
```

Example
```
:ACQUIRE:MMODE TRIGGER
:ACQUIRE:MMODE?
-> :ACQUIRE:MMODE TRIGGER
```

:ACQuire:MODE

Function Sets or queries the waveform acquisition mode.

Syntax
```
:ACQuire:MODE {BAVerage|ENVelope|
NORMal}
:ACQuire:MODE?
```

Example
```
:ACQUIRE:MODE ENVELOPE
:ACQUIRE:MODE?
-> :ACQUIRE:MODE ENVELOPE
```

:ACQuire:TIME

Function Sets or queries the measurement time.

Syntax
```
:ACQuire:TIME {<NRf>,<NRf>,<NRf>,<NRf>,
<NRf>,<NRf>}
:ACQuire:TIME?
```
1st <NRf> = day (0 to 30)
2nd <NRf> = hour (0 to 23)
3rd <NRf> = minute (0 to 59)
4th <NRf> = second (0 to 59)
5th <NRf> = millisecond (0 to 990)
6th <NRf> = microsecond (0 to 990)

Example
```
:ACQUIRE:TIME 0,0,0,0,500,0
:ACQUIRE:TIME?
-> :ACQUIRE:TIME 0,0,0,0,500,0
```

:ACQuire:PROTate

Function Sets or queries the Pulse/Rotate setting when inputting the external clock.

Syntax
```
:ACQuire:PROTate {<NRf>}
:ACQuire:PROTate?
```
<NRf> = 1 to 24000

Example
```
:ACQUIRE:PROTATE 10
:ACQUIRE:PROTATE?
-> :ACQUIRE:PROTATE 10
```

:ACQuire:MHNum?

Function Queries the maximum history number for trigger measurement mode.

Syntax `:ACQuire:MHNum? {<NRf>}`
<NRf> = measurement points

Example `:ACQUIRE:MHNUM? 10000`
`->: ACQUIRE:MHNUM 5000`

Description The function returns the maximum history number that is calculated from the measurement points and number of channels in use.

3.4 ALARm Group

The ALARm group deals with alarms (warnings).

:ALARm?

Function Queries all settings related to alarms.

Syntax `:ALARm?`

:ALARm:ACK:EXECute

Function Clears alarm output.

Syntax `:ALARm:ACK:EXECute`

Example `:ALARM:ACK:EXECUTE`

:ALARm:ACOunt?

Function Queries the number of acquisitions, which is counted from the start of measurement, when an alarm occurs.

Syntax `:ALARm:ACOunt?`

Example `:ALARM:ACOUNT? -> :ALARM:ACOUNT 46`

:ALARm:COMBination

Function Sets or queries the AND/OR state of the alarms of each channel.

Syntax `:ALARm:COMBination {AND|OR}`
`:ALARm:COMBination?`

Example `:ALARM:COMBINATION AND`
`:ALARM:COMBINATION?`
`-> :ALARM:COMBINATION AND`

:ALARm:CONDition?

Function Queries the alarm output terminal condition.

Syntax `:ALARm:CONDition?`

Example `:ALARM:CONDITION? -> :ALARM:CONDITION 0`

Description If output is on, the command returns 1. If output is off, the command returns 0.

:ALARm:CHANnel<x>?

Function Queries all settings related to channel alarms.

Syntax `:ALARm:CHANnel<x>?`
<x> = 1 to 16

:ALARm:CHANnel<x>:CONDition?

Function Queries the alarm condition of the specified channel.

Syntax `:ALARm:CHANnel<x>:CONDition?`
<x> = 1 to 16

Example `:ALARM:CHANNEL1:CONDITION?`
`-> :ALARM:CHANNEL1:CONDITION 0`

Description If the alarm is on, the command returns 1. If the alarm is off, the command returns 0.

:ALARm:CHANnel<x1>:HYSTeresis<x2>

Function Sets or queries the alarm hysteresis of a channel.

Syntax `:ALARm:CHANnel<x1>:`
`HYSTeresis<x2> {HIGH|LOW|MIDDle}`
`:ALARm:CHANnel<x1>:HYSTeresis<x2>?`
<x1> = 1 to 16
<x2> = 1, 2
If TYPE is HIGH, LOW, only level 1 is used.
If TYPE is WLIn, WLOut, both level 1 and 2 are used.
(Level 1 is the upper limit. Level 2 is the lower limit.)
HIGH: ±10% (of 10 times V/Div)
MIDDle: ±5% (of 10 times V/Div)
LOW: ±1% (of 10 times V/Div)

Example `:ALARM:CHANNEL1:HYSTERESIS1 HIGH`
`:ALARM:CHANNEL1:HYSTERESIS1?`
`-> :ALARM:CHANNEL1:HYSTERESIS1 HIGH`

:ALARm:CHANnel<x1>:LEVel<x2>

Function Sets or queries the alarm level of a channel (when the input of the specified channel is not logic).

Syntax `:ALARm:CHANnel<x1>:`
`LEVel<x2> {<Voltage>|<NRf>|<Current>}`
`:ALARm:CHANnel<x1>:LEVel<x2>?`
<x1> = 1 to 16
<x2> = 1, 2
If TYPE is HIGH, LOW, only level 1 is used.
If TYPE is WLIn, WLOut, both level 1 and 2 are used.
(Level 1 is the upper limit. Level 2 is the lower limit.)

Example `:ALARM:CHANNEL1:LEVEL1 10`
`:ALARM:CHANNEL1:LEVEL1?`
`-> :ALARM:CHANNEL1:LEVEL1 10.000E+00`

Description The Au7Fe temperature measuring range is 0 to 280 K (-273 to 7°C).

:ALARm:CHANnel<x>:TYPE

Function Sets or queries the alarm type of a channel.

Syntax `:ALARm:CHANnel<x>:TYPE`
`{HIGH|LOW|OFF|WLIn|WLOut}`
`:ALARm:CHANnel<x>:TYPE?`
<x> = 1 to 16

Example `:ALARM:CHANNEL1:TYPE HIGH`
`:ALARM:CHANNEL1:TYPE?`
`-> :ALARM:CHANNEL1:TYPE HIGH`

:ALARm:CHANnel<x>:AVALue? (Alarm VALue)

Function Queries the measured value at the alarm occurrence on the specified channels as an ASCII string.

Syntax `:ALARm:CHANnel<x>:AVALue?`
<x> = 1 to 16

Example
```
:ALARM:CHANNEL1:AVALUE?
-> :ALARM:CHANNEL1:AVALUE "CH1 1.022V"
```

Description Output format is the same as the ASCII output format for the ‘Monitor group’.

:ALARm:CTIMe?

Function Queries the time of the most recent channel alarm condition change.

Syntax `:ALARm:CTIMe?`

Example
```
:ALARM:CTIME?
-> :ALARM:CTIME 2008,7,10,10,14,8,300
```

Description The function returns the time of the channel alarm condition change in order of year/month/day/hour/minute/second/microsecond. If no status changes occur after measurement starts, the function returns the time when measurement started. If measurement has not started, the function returns an undefined value.

:ALARm:HOLD

Function Sets or queries the alarm hold.

Syntax
```
:ALARm:HOLD {<Boolean>}
:ALARm:HOLD?
```

Example
```
:ALARM:HOLD ON
:ALARM:HOLD? -> :ALARM:HOLD 1
```

Description If the alarm hold is on, alarm output will continue until ALAR:ACK:EXEC is received even if the condition that caused the alarm is cleared.

:ALARm:MODE

Function Sets or queries the alarm operation mode.

Syntax
```
:ALARm:MODE {OFF|ON}
:ALARm:MODE?
```

Example
```
:ALARM:MODE ON
:ALARM:MODE? -> :ALARM:MODE ON
```

Description Off: The SL1000 does not output alarms.
On: The SL1000 detects alarms during measurement (if alarm hold is on, the alarm will continue to output even after measurement stops).

:ALARm:CMODe

Function Sets or queries the channel alarm operation mode.

Syntax
```
:ALARm:CMODe {<Boolean>}
:ALARm:CMOD?
```

Example
```
:ALARM:CMODE ON
:ALARM:CMODE? -> :ALARM:CMODE 1
```

Description Sets whether to detect or not detect the channel alarm.

:ALARm:SMODe

Function Sets or queries the system alarm operation mode.

Syntax
```
:ALARm:SMODe {<Boolean>}
:ALARm:SMODe?
```

Example
```
:ALARM:SMODE ON
:ALARM:SMODE? -> :ALARM:SMODE 1
```

Description Sets whether to detect or not detect the system alarm.

:ALARm:OTERminal

Function Sets or queries the alarm output terminal on/off state.

Syntax
```
:ALARm:OTERminal {<Boolean>}
:ALARm:OTERminal?
```

Example
```
:ALARM:OTERMINAL ON
:ALARM:OTERMINAL? -> :ALARM:OTERMINAL 1
```

Description If the alarm output terminal is switched off, it will remain off regardless of the alarm settings or conditions.

:ALARm:SOURce

Function Sets or queries the alarm detection source.

Syntax
```
:ALARm:SOURce {CHANnel|SYSTem}
:ALARm:SOURce?
```

Example
```
:ALARM:SOURCE CHANNEL
:ALARM:SOURCE? -> :ALARM:SOURCE CHANNEL
```

:ALARm:STATus?

Function Queries the channel alarm status.

Syntax `:ALARm:STATus?`

Example When the alarm condition is ON at CH1 and CH3.
```
:ALARM:STATUS? -> :ALARM:STATUS 40960
```

Description Since the :ALARm:CHANnel<x>:CONDition? command must be used repeatedly to acquire the alarm statuses of all channels, this command returns a bit pattern. The MSB of the returned value (bit 15) represents channel 1. The LSB (bit 0) represents channel 16.

:ALARm:SSTatus?

Function Queries the system alarm status value.

Syntax `:ALARm:SSTatus?`

Example `:ALARM:SSTATUS? -> :ALARM:SSTATUS 2`

Description The system status bit assignments are shown in the table below.

bit	symbol	Notes
0	(reserved)	
1	HDD_FULL	HDD disk Full
2	FUN_STOP	Fun stop
3	BUF_OVERRUN_PP	Real time record index data full
4	BUF_OVERRUN_HOST	Host (PC) buffer overrun
5	BUF_OVERRUN_UNIT	SL1000 buffer overrun
6	(system reserved)	
7	(reserved)	
8	(reserved)	
9	(reserved)	
10	(reserved)	
11	(reserved)	
12	(reserved)	
13	(reserved)	
14	(reserved)	
15	(reserved)	

:ALARm:SYSTem?

Function Queries all settings related to the system alarm.

Syntax `:ALARm:SYSTem?`

:ALARm:SYSTem:SOURce?

Function Queries all settings related to system alarm detection.

Syntax `:ALARm:SYSTem:SOURce?`

:ALARm:SYSTem:SOURce:BOVerrun

Function Sets or queries system alarm buffer overrun detection.

Syntax
```
:ALARm:SYSTem:SOURce:
BOVerrun {<Boolean>}
:ALARm:SYSTem:SOURce:BOVerrun?
```

Example
```
:ALARM:SYSTEM:SOURCE:BOVERRUN ON
:ALARM:SYSTEM:SOURCE:BOVERRUN?
-> :ALARM:SYSTEM:SOURCE:BOVERRUN 1
```

:ALARm:SYSTem:SOURce:FSTop

Function Sets or queries system alarm fan stop detection.

Syntax
```
:ALARm:SYSTem:SOURce:FSTop {<Boolean>}
:ALARm:SYSTem:SOURce:FSTop?
```

Example
```
:ALARM:SYSTEM:SOURCE:FSTOP ON
:ALARM:SYSTEM:SOURCE:FSTOP?
-> :ALARM:SYSTEM:SOURCE:FSTOP 1
```

:ALARm:SYSTem:SOURce:DFULl

Function Sets or queries system alarm HDD full detection.

Syntax
```
:ALARm:SYSTem:SOURce:DFULl {<Boolean>}
:ALARm:SYSTem:SOURce:DFULl?
```

Example
```
:ALARM:SYSTEM:SOURCE:DFULL ON
:ALARM:SYSTEM:SOURCE:DFULL?
-> :ALARM:SYSTEM:SOURCE:DFULL 1
```

:ALARm:SYSTem:SOURce:MRTime

Function Sets or queries the maximum recording time detection for free-run automatic recording mode.

Syntax
```
:ALARm:SYSTem:SOURce:MRTime {<Boolean>}
:ALARm:SYSTem:SOURce:MRTime?
```

Example
```
:ALARM:SYSTEM:SOURCE:MRTIME ON
:ALARM:SYSTEM:SOURCE:MRTIME?
-> :ALARM:SYSTEM:SOURCE:MRTIME 1
```

:ALARm:STIMe?

Function Queries the time of the most recent system alarm condition change.

Syntax `:ALARm:STIMe?`

Example
```
:ALARM:STIME?
-> :ALARM:STIME 2008,7,10,10,14,8,300
```

Description The function returns the time of the system alarm condition change in order of year/month/day/hour/minute/second/microsecond. If no status changes occur after measurement starts, the function returns the time when measurement started. If measurement has not started, the function returns an undefined value.

3.5 ASETup Group

The ASETup group deals with auto setup.

:ASETup?

Function Queries all settings related to auto setup.

Syntax `:ASETup?`

:ASETup:EXECute

Function Execute auto setup.

Syntax `:ASETup:EXECute`

Example `:ASETUP:EXECUTE`

Description
- When the measurement mode is free-run, only the settings related to the vertical axes are set up automatically.
- Do not send other commands while executing auto setup. The processing status can be inferred from Bit 9 of the status register.

:ASETup:TARGet

Function Sets or queries the target channel for auto setup.

Syntax `:ASETup:TARGet {ALL|<NRf>}`
<NRf> = 1 to 16

Example `:ASETUP:TARGET 1`
`:ASETUP:TARGET? -> :ASETUP:TARGET 1`

Description An error occurs if you specify a channel in which a module is not installed.

:ASETup:UNDO

Function Cancels auto setup.

Syntax `:ASETup:UNDO`

Example `:ASETUP:UNDO`

Description Do not send other commands while canceling auto setup. The processing status can be inferred from Bit 9 of the status register.

3.6 CALibrate Group

The CALibrate group deals with calibration.

:CALibrate?

Function Queries all settings related to calibration.

Syntax `:CALibrate?`

:CALibrate[:EXECute]

Function Executes calibration.

Syntax `:CALibrate:EXECute`

Example `:CALIBRATE:EXECUTE`

Description Unlike the common command *CAL2?, the function will not return the result at the end of calibration.

:CALibrate:MODE

Function Sets or queries the ON/OFF state of auto calibration.

Syntax
```
:CALibrate:MODE {AUTO|OFF}
:CALibrate:MODE?
```

Example
```
:CALIBRATE:MODE AUTO
:CALIBRATE:MODE?
-> :CALIBRATE:MODE AUTO
```

3.7 CHANnel Group

The CHANnel group deals with the vertical axis of each channel.

:CHANnel<x>?

Function Queries all settings related to the vertical axis of the channel.

Syntax `:CHANnel<x>?`
<x> = 1 to 16

Description An error occurs if a module is not installed in the channel (slot).

:CHANnel<x>:ACCL?

Function Queries all settings on the channel with the acceleration/voltage module installed.

Syntax `:CHANnel<x>:ACCL?`
<x> = 1 to 16

Description An error occurs if the acceleration/voltage module is not installed.

:CHANnel<x>:ACCL:BIAS

Function Sets or queries the on/off status of the acceleration sensor's bias current when an Acceleration/Voltage Module is installed in the specified channel (slot).

Syntax
```
:CHANnel<x>:ACCL:BIAS {<Boolean>}
:CHANnel<x>:ACCL:BIAS?
```
<x> = 1 to 16

Example
```
:CHANNEL1:ACCL:BIAS ON
:CHANNEL1:ACCL:BIAS?
-> :CHANNEL1:ACCL:BIAS 1
```

Description An error occurs if an Acceleration/Voltage Module is not installed.

:CHANnel<x>:ACCL:BWIDth

Function Sets or queries the filter when an Acceleration Voltage Module is installed in the specified channel (slot).

Syntax
```
:CHANnel<x>:ACCL:BWIDth {FULL|AUTO|
<Frequency>}
:CHANnel<x>:ACCL:BWIDth?
```
<x> = 1 to 16
<Frequency> = 4 kHz, 400 Hz, or 40 Hz

Example
```
:CHANNEL1:ACCL:BWIDTH AUTO
:CHANNEL1:ACCL:BWIDTH?
-> :CHANNEL1:ACCL:BWIDTH AUTO
```

Description An error occurs if an Acceleration/Voltage Module is not installed.

:CHANnel<x>:ACCL:COUPling

Function Sets or queries input coupling when an Acceleration/ Voltage Module is installed in the specified channel (slot)

Syntax
```
:CHANnel<x>:ACCL:COUPling {AC|DC|ACCL|
GND}
:CHANnel<x>:ACCL:COUPling?
```
<x> = 1 to 16

Example
```
:CHANNEL1:ACCL:COUPLING GND
:CHANNEL1:ACCL:COUPLING?
-> :CHANNEL1:ACCL:COUPLING GND
```

Description An error occurs if an Acceleration/Voltage Module is not installed.

:CHANnel<x>:ACCL:GAIN

Function Sets or queries the gain when an Acceleration/ Voltage Module is installed in the specified channel (slot).

Syntax
```
:CHANnel<x>:ACCL:GAIN {<NRf>}
:CHANnel<x>:ACCL:GAIN?
```
<x> = 1 to 16
<NRf> = 0.1, 0.2, 0.5, 1, 2, 5, 10, 20, 50, 100

Example
```
:CHANNEL1:ACCL:GAIN 100
:CHANNEL1:ACCL:GAIN?
-> :CHANNEL1:ACCL:GAIN 100.0
```

Description An error occurs if an Acceleration/Voltage Module is not installed.

:CHANnel<x>:ACCL:SENSitivity

Function Sets or queries the sensitivity when an Acceleration/ Voltage Module is installed in the specified channel (slot).

Syntax
```
:CHANnel<x>:ACCL:SENSitivity {<NRf>}
:CHANnel<x>:ACCL:SENSitivity?
```
<x> = 1 to 16
<NRf> = 0.1 to 2000

Example
```
:CHANNEL1:ACCL:SENSITIVITY 10
:CHANNEL1:ACCL:SENSITIVITY?
-> :CHANNEL1:ACCL:SENSITIVITY 10.00
```

Description An error occurs if an Accelerat ion/Voltage Module is not installed.

:CHANnel<x>:ACCL:UNIT

Function Sets or queries the unit of measurement of the upper and lower limit values when an Acceleration/Voltage Module is installed in the specified channel(slot).

Syntax
```
:CHANnel<x>:ACCL:UNIT {<String>}
:CHANnel<x>:ACCL:UNIT?
```
<x> = 1 to 16

Example
```
:CHANNEL1:ACCL:UNIT "ACCL"
:CHANNEL1:ACCL:UNIT?
-> :CHANNEL1:ACCL:UNIT "ACCL"
```

Description An error occurs if an Acceleration/Voltage Module is not installed.

:CHANnel<x>:ASET?

Function Queries whether the specified channel is able to set an auto setup or not.

Syntax
```
:CHANnel<x>:ASET?
```
<x> = 1 to 16

Example
```
:CHANNEL1:ASET? -> :CHANNEL1:ASET 1
```

Description If auto setup is able to set, the function returns 1. If auto setup is impossible to set, the function returns 0. An error occurs if the module is not installed in the specified channel (slot).

:CHANnel<x>:DISPlay

Function Sets or queries the channel acquisition ON/OFF state.

Syntax
```
:CHANnel<x>:DISPlay {<Boolean>}
:CHANnel<x>:DISPlay?
```
<x> = 1 to 16

Example
```
:CHANNEL1:DISPLAY ON
:CHANNEL1:DISPLAY?
-> :CHANNEL1:DISPLAY 1
```

Description The default value of this command is OFF. Please set to ON first to use the channel.
An error occurs if the module is not installed in the specified channel (slot).

:CHANnel<x>:FREQ?

Function Queries all settings when a frequency module is installed in the specified channel (slot).

Syntax
```
:CHANnel<x>:FREQ?
```
<x> = 1 to 16

Description An error occurs if the frequency module is not installed.

:CHANnel<x>:FREQ:INPut?

Function Queries all settings related to the input when a frequency module is installed in the specified channel (slot).

Syntax
```
:CHANnel<x>:FREQ:INPut?
```
<x> = 1 to 16

Description An error occurs if the frequency module is not installed.

:CHANnel<x>:FREQ:INPut:BWIDth

Function Sets or queries the bandwidth limit when a frequency module is installed in the specified channel (slot).

Syntax
```
:CHANnel<x>:FREQ:INPut:BWIDth {FULL|
<Frequency>}
:CHANnel<x>:FREQ:INPut:BWIDth?
```
<x> = 1 to16

<Frequency> = 100 Hz, 1 kHz, 10 kHz, 100 kHz

Example
```
:CHANNEL1:FREQ:INPUT:BWIDTH FULL
:CHANNEL1:FREQ:INPUT:BWIDTH?
-> :CHANNEL1:FREQ:INPUT:BWIDTH FULL
```

Description An error occurs if a frequency module is not installed.

:CHANnel<x>:FREQ:INPut:CELimination

Function Sets or queries chattering elimination when a frequency module is installed in the specified channel (slot).

Syntax
```
:CHANnel<x>:FREQ:INPut:
CELimination {<Time>}
:CHANnel<x>:FREQ:INPut:CELimination?
```
<x> = 1 to 16

<Time> = 0 to 1000 ms

Example
```
:CHANNEL1:FREQ:INPUT:CELIMINATION 100ms
:CHANNEL1:FREQ:INPUT:CELIMINATION?
-> :CHANNEL1:FREQ:INPUT:
CELIMINATION 0.100
```

Description An error occurs if a frequency module is not installed.

:CHANnel<x>:FREQ:INPut:COUPling

Function Sets or queries input coupling when a frequency module is installed in the specified channel (slot).

Syntax
```
:CHANnel<x>:FREQ:INPut:COUPling {AC|DC}
:CHANnel<x>:FREQ:INPut:COUPling?
```
<x> = 1 to 16

Example
```
:CHANNEL1:FREQ:INPUT:COUPLING DC
:CHANNEL1:FREQ:INPUT:COUPLING?
-> :CHANNEL1:FREQ:INPUT:COUPLING DC
```

Description An error occurs if a frequency module is not installed.

:CHANnel<x>:FREQ:INPut:HYSTeresis

Function Sets or queries hysteresis when a frequency module is installed in the specified channel (slot).

Syntax
```
:CHANnel<x>:FREQ:INPut:
HYSTeresis {HIGH|LOW|MIDDle}
:CHANnel<x>:FREQ:INPut:HYSTeresis?
```
<x> = 1 to 16

Example
```
:CHANNEL1:FREQ:INPUT:HYSTERESIS LOW
:CHANNEL1:FREQ:INPUT:HYSTERESIS?
-> :CHANNEL1:FREQ:INPUT:HYSTERESIS LOW
```

Description An error occurs if a frequency module is not installed.

:CHANnel<x>:FREQ:INPut:PRESet

Function Sets or queries the preset when a frequency module is installed in the specified channel (slot).

Syntax
```
:CHANnel<x>:FREQ:INPut:PRESet {AC100v|
AC200v|EMPichup|LOG12v|LOG24v|LOG3v|
LOG5v|PULLup|USER|ZERO}
:CHANnel<x>:FREQ:INPut:PRESet?
```
<x> = 1 to 16

Example
```
:CHANNEL1:FREQ:INPUT:PRESET USER
:CHANNEL1:FREQ:INPUT:PRESET?
-> :CHANNEL1:FREQ:INPUT:PRESET USER
```
Description An error occurs if a frequency module is not installed.

:CHANnel<x>:FREQ:INPut:PROBe

Function Sets or queries the probe attenuation when a frequency module is installed in the specified channel (slot).

Syntax
```
:CHANnel<x>:FREQ:INPut:PROBe {<NRf>}
:CHANnel<x>:FREQ:INPut:PROBe?
```
<x> = 1 to 16
<NRf> = 1, 10

Example
```
:CHANNEL1:FREQ:INPUT:PROBE 10
:CHANNEL1:FREQ:INPUT:PROBE?
-> :CHANNEL1:FREQ:INPUT:PROBE 10
```
Description An error occurs if a frequency module is not installed.

:CHANnel<x>:FREQ:INPut:PULLup

Function Sets or queries the pull-up on/off state when a frequency module is installed in the specified channel (slot).

Syntax
```
:CHANnel<x>:FREQ:INPut:
PULLup {<Boolean>}
:CHANnel<x>:FREQ:INPut:PULLup?
```
<x> = 1 to 16

Example
```
:CHANNEL1:FREQ:INPUT:PULLUP ON
:CHANNEL1:FREQ:INPUT:PULLUP?
-> :CHANNEL1:FREQ:INPUT:PULLUP 1
```
Description An error occurs if a frequency module is not installed.

:CHANnel<x>:FREQ:INPut:SLOPe

Function Sets or queries the slope when a frequency module is installed in the specified channel (slot).

Syntax
```
:CHANnel<x>:FREQ:INPut:
SLOPe {FALL|RISE}
:CHANnel<x>:FREQ:INPut:SLOPe?
```
<x> = 1 to 16

Example
```
:CHANNEL1:FREQ:INPUT:SLOPE FALL
:CHANNEL1:FREQ:INPUT:SLOPE?
-> :CHANNEL1:FREQ:INPUT:SLOPE FALL
```
Description An error occurs if a frequency module is not installed.

:CHANnel<x>:FREQ:INPut:THReshold

Function Sets or queries the threshold level when a frequency module is installed in the specified channel (slot).

Syntax
```
:CHANnel<x>:FREQ:INPut:
THReshold {<Voltage>}
:CHANnel<x>:FREQ:INPut:THReshold?
```
<x> = 1 to 16
<Voltage> = 100 to -100V

Example
```
:CHANNEL1:FREQ:INPUT:THRESHOLD 10
:CHANNEL1:FREQ:INPUT:THRESHOLD?
-> :CHANNEL1:FREQ:INPUT:
THRESHOLD 10.000E+00
```
Description An error occurs if a frequency module is not installed.

:CHANnel<x>:FREQ:INPut:VRANGe

Function Sets or queries the voltage range when a frequency module is installed in the specified channel (slot).

Syntax
```
:CHANnel<x>:FREQ:INPut:
VRANGe {<Voltage>}
:CHANnel<x>:FREQ:INPut:VRANGe?
```
<x> = 1 to 16
<Voltage> = 1 to 500 V

Example
```
:CHANNEL1:FREQ:INPUT:VRANGE 10
:CHANNEL1:FREQ:INPUT:VRANGE?
-> :CHANNEL1:FREQ:INPUT:VRANGE 10
```
Description An error occurs if a frequency module is not installed.

:CHANnel<x>:FREQ:LSCale?

Function Queries all settings related to linear scaling when a frequency module is installed in the specified channel (slot).

Syntax
```
:CHANnel<x>:FREQ:LSCale?
```
<x> = 1 to 16

Description An error occurs if a frequency module is not installed.

:CHANnel<x>:FREQ:LSCale:AVALue

Function Sets or queries linear scaling coefficient A when a frequency module is installed in the specified channel (slot).

Syntax
```
:CHANnel<x>:FREQ:LSCale:AVALue {<NRf>}
:CHANnel<x>:FREQ:LSCale:AVALue?
```
<x> = 1 to 16
<NRf> = -9.9999E+30 to 9.9999E+30

Example
```
:CHANNEL1:FREQ:LSCALE:AVALUE 10
:CHANNEL1:FREQ:LSCALE:AVALUE?
-> :CHANNEL1:FREQ:LSCALE:
AVALUE 10.0000E+00
```
Description An error occurs if a frequency module is not installed.

:CHANnel<x>:FREQ:LSCale:BVALue

Function Sets or queries linear scaling coefficient B when a frequency module is installed in the specified channel (slot).

Syntax
```
:CHANnel<x>:FREQ:LSCale:BVALue {<NRf>}
:CHANnel<x>:FREQ:LSCale:BVALue?
```
<x> = 1 to 16
<NRf> = -9.9999E+30 to 9.9999E+30

Example
```
:CHANNEL1:FREQ:LSCALE:BVALUE 10
:CHANNEL1:FREQ:LSCALE:BVALUE?
-> :CHANNEL1:FREQ:LSCALE:
BVALUE 10.0000E+00
```

Description An error occurs if a frequency module is not installed.

:CHANnel<x>:FREQ:LSCale:GETMeasure

Function Measures the X values of P1 and P2 for linear scaling when a frequency module is installed in the specified channel (slot).

Syntax
```
:CHANnel<x>:FREQ:LSCale:
GETMeasure {P1X|P2X}
```
<x> = 1 to 16

Example
```
:CHANNEL1:FREQ:LSCALE:GETMEASURE P1X
```

Description An error occurs if a frequency module is not installed.

:CHANnel<x>:FREQ:LSCale:MODE

Function Sets or queries linear scaling when a frequency module is installed in the specified channel (slot).

Syntax
```
:CHANnel<x>:FREQ:LSCale:
MODE {AXB|OFF|P12}
:CHANnel<x>:FREQ:LSCale:MODE?
```
<x> = 1 to 16

Example
```
:CHANNEL1:FREQ:LSCALE:MODE OFF
:CHANNEL1:FREQ:LSCALE:MODE?
-> :CHANNEL1:FREQ:LSCALE:MODE OFF
```

Description An error occurs if a frequency module is not installed.

:CHANnel<x>:FREQ:LSCale:{P1X|P1Y|P2X|P2Y}

Function Sets or queries the X or Y value of P1 or P2 for linear scaling when a frequency module is installed in the specified channel (slot).

Syntax
```
:CHANnel<x>:FREQ:LSCale:{P1X|P1Y|P2X|
P2Y} {<NRf>}
:CHANnel<x>:FREQ:LSCale:{P1X|P1Y|P2X|
P2Y}?
```
<x> = 1 to 16
For P1X and P2X,
<NRf> = -9.9999E+30 to 9.9999E+30
For P1Y and P2Y,
<NRf> = -9.9999E+25 to 9.9999E+25

Example
```
:CHANNEL1:FREQ:LSCALE:P1X 10
:CHANNEL1:FREQ:LSCALE:P1X?
-> :CHANNEL1:FREQ:LSCALE:P1X 10.0000E+00
```

Description An error occurs if a frequency module is not installed.

:CHANnel<x>:FREQ:LSCale:UNIT

Function Sets or queries the unit of measurement to attach to the result of linear scaling when a frequency module is installed in the specified channel (slot).

Syntax
```
:CHANnel<x>:FREQ:LSCale:
UNIT {<String>}
:CHANnel<x>:FREQ:LSCale:UNIT?
```
<x> = 1 to 16
<String> = Up to 4 characters

Example
```
:CHANNEL1:FREQ:LSCALE:UNIT "AAA"
:CHANNEL1:FREQ:LSCALE:UNIT?
-> :CHANNEL1:FREQ:LSCALE:UNIT "AAA"
```

Description An error occurs if a frequency module is not installed.

:CHANnel<x>:FREQ:OFFSet

Function Sets or queries the offset value when a frequency module is installed in the specified channel (slot).

Syntax
```
:CHANnel<x>:FREQ:OFFSet {<NRf>|
<Frequency>|<Time>}
:CHANnel<x>:FREQ:OFFSet?
```
<x> = 1 to 16

Example
```
:CHANNEL1:FREQ:OFFSET 1
:CHANNEL1:FREQ:OFFSET?
-> :CHANNEL1:FREQ:OFFSET 1.000000E+00
```

Description An error occurs if a frequency module is not installed.

Offset Range

Function	Max	Resolution
Frequency	701281, 720281: 500 kHz, 1000 times of V/Div Small value of above 701280: 200 kHz, 1000 times of V/Div Small value of above	1 1/1000 of V/Div 2 or 5 1/1000 of V/Div
Rotation (rpm)	50 krpm, 1000 times of V/Div Small value of above	1/2000 of V/Div 1/500 of V/Div
Rotation (rps)	1000 rps	1/2000 of V/Div
Period	50 sec	1/2000 of V/Div
Duty	100%	1/1000 of V/Div
Power Freq	–	–
Pulse width	50 sec	1/2000 of V/Div
Pulse Integ	1.0000E+22	1.0E-30
Speed	1.0000E+22	1.0E-30

Offset is not available when Power Freq is set.

:CHANnel<x>:FREQ:SETup?

Function Queries all settings related to FV setup when a frequency module is installed in the specified channel (slot).

Syntax
```
:CHANnel<x>:FREQ:SETup?
```
<x> = 1 to 16

Description An error occurs if a frequency module is not installed.

:CHANnel<x>:FREQ:SETup:CFRequency

Function Sets or queries the center frequency when a frequency module is installed in the specified channel (slot).

Syntax
```
:CHANnel<x>:FREQ:SETup:
CFRequency {<Frequency>}
:CHANnel<x>:FREQ:SETup:CFRequency?
```
<x> = 1 to 16

<NRf> = 50 Hz, 60 Hz, 400 Hz

Example
```
:CHANNEL1:FREQ:SETUP:CFREQUENCY 50
:CHANNEL1:FREQ:SETUP:CFREQUENCY?
-> :CHANNEL1:FREQ:SETUP:CFREQUENCY 50
```
Description An error occurs if a frequency module is not installed.

:CHANnel<x>:FREQ:SETup:DECeleration

Function Sets or queries the on/off state of deceleration prediction when a frequency module is installed in the specified channel (slot).

Syntax
```
:CHANnel<x>:FREQ:SETup:
DECeleration {<Boolean>}
:CHANnel<x>:FREQ:SETup:DECeleration?
```
<x> = 1 to 16

Example
```
:CHANNEL1:FREQ:SETUP:DECELERATION ON
:CHANNEL1:FREQ:SETUP:DECELERATION?
-> :CHANNEL1:FREQ:SETUP:DECELERATION 1
```
Description An error occurs if a frequency module is not installed.

:CHANnel<x>:FREQ:SETup:DPULse

Function Sets or queries the distance per pulse when a frequency module is installed in the specified channel (slot).

Syntax
```
:CHANnel<x>:FREQ:SETup:DPULse {<NRf>}
:CHANnel<x>:FREQ:SETup:DPULse?
```
<x> = 1 to 16

<NRf> = -9.9999E+30 to 9.9999E+30

Example
```
:CHANNEL1:FREQ:SETUP:DPULSE 1e15
:CHANNEL1:FREQ:SETUP:DPULSE?
-> :CHANNEL1:FREQ:SETUP:
DPULSE 1.00000E+15
```
Description An error occurs if a frequency module is not installed.

:CHANnel<x>:FREQ:SETup:FILTer?

Function Queries all settings related to the filter when a frequency module is installed in the specified channel (slot).

Syntax
```
:CHANnel<x>:FREQ:SETup:FILTer?
```
<x> = 1 to 16

Description An error occurs if a frequency module is not installed.

:CHANnel<x>:FREQ:SETup:FILTer:SMOothing?

Function Queries all settings related to smoothing when a frequency module is installed in the specified channel (slot).

Syntax
```
:CHANnel<x>:FREQ:SETup:FILTer:
SMOothing?
```
<x> = 1 to 16

Description An error occurs if a frequency module is not installed.

:CHANnel<x>:FREQ:SETup:FILTer:SMOothing:MODE

Function Sets or queries the on/off state of smoothing when a frequency module is installed in the specified channel (slot).

Syntax
```
:CHANnel<x>:FREQ:SETup:FILTer:
SMOothing:MODE {<Boolean>}
:CHANnel<x>:FREQ:SETup:FILTer:
SMOothing:MODE?
```
<x> = 1 to 16

Example
```
:CHANNEL1:FREQ:SETUP:FILTER:SMOOTHING:
MODE ON
:CHANNEL1:FREQ:SETUP:FILTER:SMOOTHING:
MODE?
-> :CHANNEL1:FREQ:SETUP:FILTER:
SMOOTHING:MODE 1
```
Description An error occurs if a frequency module is not installed.

:CHANnel<x>:FREQ:SETup:FILTer:SMOothing:VALue

Function Sets or queries the moving average order of smoothing when a frequency module is installed in the specified channel (slot).

Syntax
```
:CHANnel<x>:FREQ:SETup:FILTer:
SMOothing:VALue {<Time>}
:CHANnel<x>:FREQ:SETup:FILTer:
SMOothing:VALue?
```
<x> = 1 to 16

<Time> = 0 to 1000

Example
```
:CHANNEL1:FREQ:SETUP:FILTER:SMOOTHING:
VALUE 10ms
:CHANNEL1:FREQ:SETUP:FILTER:SMOOTHING:
VALUE?
-> :CHANNEL1:FREQ:SETUP:FILTER:
SMOOTHING:VALUE 0.0100
```
Description An error occurs if a frequency module is not installed.

:CHANnel<x>:FREQ:SETup:FILTer:PAVerage?

Function Queries all settings related to pulse average when a frequency module is installed in the specified channel (slot).

Syntax
```
:CHANnel<x>:FREQ:SETup:FILTer:PAVerage?
```
<x> = 1 to 16

Description An error occurs if a frequency module is not installed.

:CHANnel<x>:FREQ:SETup:FILTer:PAVerage:MODE

Function Sets or queries the on/off state of pulse average mode when a frequency module is installed in the specified channel (slot).

Syntax
```
:CHANnel<x>:FREQ:SETup:FILTer:
PAVerage:MODE {<Boolean>}
:CHANnel<x>:FREQ:SETup:FILTer:
PAVerage:MODE?
```
<x> = 1 to 16

Example
```
:CHANNEL1:FREQ:SETUP:FILTER:PAVERAGE:
MODE ON
:CHANNEL1:FREQ:SETUP:FILTER:PAVERAGE:
MODE?
-> :CHANNEL1:FREQ:SETUP:FILTER:
PAVERAGE:MODE 1
```

Description An error occurs if a frequency module is not installed.

:CHANnel<x>:FREQ:SETup:FILTer:PAVerage:VALue

Function Sets or queries the pulse average count when a frequency module is installed in the specified channel (slot).

Syntax
```
:CHANnel<x>:FREQ:SETup:FILTer:
PAVerage:VALue {<NRf>}
:CHANnel<x>:FREQ:SETup:FILTer:
PAVerage:VALue?
```
<x> = 1 to 16
<NRf> = 1 to 4096

Example
```
:CHANNEL1:FREQ:SETUP:FILTER:PAVERAGE:
VALUE 10
:CHANNEL1:FREQ:SETUP:FILTER:PAVERAGE:
VALUE?
-> :CHANNEL1:FREQ:SETUP:FILTER:
PAVERAGE:VALUE 10
```

Description An error occurs if a frequency module is not installed.

:CHANnel<x>:FREQ:SETup:FUNCtion

Function Sets or queries the measuring mode when a frequency module is installed in the specified channel (slot).

Syntax
```
:CHANnel<x>:FREQ:SETup:FUNCtion {DUTY|
FREQuency|PERiod|PFReq|PINTeg|PWIDth|
RPM|RPS|VELocity}
:CHANnel<x>:FREQ:SETup:FUNCtion?
```
<x> = 1 to 16

Example
```
:CHANNEL1:FREQ:SETUP:FUNCTION FREQUENCY
:CHANNEL1:FREQ:SETUP:FUNCTION?
-> :CHANNEL1:FREQ:SETUP:
FUNCTION FREQUENCY
```

Description An error occurs if a frequency module is not installed.

Setting and Command (Parameter)

Setting	Parameter
Frequency	FREQuency
Revolution (rpm)	RPM
Revolution (rps)	RPS
Period	PERiod
Duty	DUTY
Power Freq.	PFReq
Pulse Width	PWIDth
Pulse Integ	PINTeg
Velocity	VELocity

:CHANnel<x>:FREQ:SETup:LRESet

Function Sets or queries the over limit reset when a frequency module is installed in the specified channel (slot).

Syntax
```
:CHANnel<x>:FREQ:SETup:
LRESet {<Boolean>}
:CHANnel<x>:FREQ:SETup:LRESet?
```
<x> = 1 to 16

Example
```
:CHANNEL1:FREQ:SETUP:LRESET ON
:CHANNEL1:FREQ:SETUP:LRESET?
-> :CHANNEL1:FREQ:SETUP:LRESET 1
```

Description An error occurs if a frequency module is not installed.

:CHANnel<x>:FREQ:SETup:MPULse

Function Sets or queries whether the measurement pulse is positive or negative when a frequency module is installed in the specified channel (slot).

Syntax
```
:CHANnel<x>:FREQ:SETup:
MPULse {POSitive|NEGative}
:CHANnel<x>:FREQ:SETup:MPULse?
```
<x> = 1 to 16

Example
```
:CHANNEL1:FREQ:SETUP:MPULSE POSITIVE
:CHANNEL1:FREQ:SETUP:MPULSE?
-> :CHANNEL1:FREQ:SETUP:MPULSE POSITIVE
```

Description An error occurs if a frequency module is not installed.

:CHANnel<x>:FREQ:SETup:PROTate

Function Sets or queries the number of pulses per rotation when a frequency module is installed in the specified channel (slot).

Syntax
```
:CHANnel<x>:FREQ:SETup:PROTate {<NRf>}
:CHANnel<x>:FREQ:SETup:PROTate?
```
<x> = 1 to 16
<NRf> = 1 to 99999

Example
```
:CHANNEL1:FREQ:SETUP:PROTATE 10
:CHANNEL1:FREQ:SETUP:PROTATE?
-> :CHANNEL1:FREQ:SETUP:PROTATE 10
```

Description An error occurs if a frequency module is not installed.

:CHANnel<x>:FREQ:SETup:RESet

Function Resets the pulse count when a frequency module is installed in the specified channel (slot)

Syntax
```
:CHANnel<x>:FREQ:SETup:RESet
```
<x> = 1 to 16

Example
```
:CHANNEL1:FREQ:SETUP:RESET
```

Description An error occurs if a frequency module is not installed.

:CHANnel<x>:FREQ:SETup:STOPpredict

Function Sets or queries the on/off state of stop prediction when a frequency module is installed in the specified channel (slot).

Syntax
```
:CHANnel<x>:FREQ:SETup:
STOPpredict {<NRf>|OFF}
:CHANnel<x>:FREQ:SETup:STOPpredict?
```
<x> = 1 to 16
<NRf> = 1.5, 2, 3, 4, 5, 6, 7, 8, 9, 10

Example
```
:CHANNEL1:FREQ:SETUP:STOPPREDICT OFF
:CHANNEL1:FREQ:SETUP:STOPPREDICT?
-> :CHANNEL1:FREQ:SETUP:STOPPREDICT OFF
```

Description An error occurs if a frequency module is not installed.

:CHANnel<x>:FREQ:SETup:TUNit

Function Sets or queries the time unit when a frequency module is installed in the specified channel (slot).

Syntax
```
:CHANnel<x>:FREQ:SETup:TUNit {HOUR|
MIN|SEC}
:CHANnel<x>:FREQ:SETup:TUNit?
```
<x> = 1 to 16

Example
```
:CHANNEL1:FREQ:SETUP:TUNIT SEC
:CHANNEL1:FREQ:SETUP:TUNIT?
-> :CHANNEL1:FREQ:SETUP:TUNIT SEC
```

Description An error occurs if a frequency module is not installed.

:CHANnel<x>:FREQ:SETup:UNIT

Function Sets or queries the pulse integration unit when a frequency module is installed in the specified channel (slot).

Syntax
```
:CHANnel<x>:FREQ:SETup:UNIT {<String>}
:CHANnel<x>:FREQ:SETup:UNIT?
```
<x> = 1 to 16
<String> = Up to 4 characters

Example
```
:CHANNEL1:FREQ:SETUP:UNIT "AAA"
:CHANNEL1:FREQ:SETUP:UNIT?
-> :CHANNEL1:FREQ:SETUP:UNIT "AAA"
```

Description An error occurs if a frequency module is not installed.

:CHANnel<x>:FREQ:SETup:UPULse

Function Sets or queries the unit/pulse when a frequency module is installed in the specified channel (slot).

Syntax
```
:CHANnel<x>:FREQ:SETup:UPULse {<NRf>}
:CHANnel<x>:FREQ:SETup:UPULse?
```
<x> = 1 to 16
<NRf> = -9.9999E+30 to 9.9999E+30

Example
```
:CHANNEL1:FREQ:SETUP:UPULSE 1e15
:CHANNEL1:FREQ:SETUP:UPULSE?
-> :CHANNEL1:FREQ:SETUP:
UPULSE 1.00000E+15
```

Description An error occurs if a frequency module is not installed.

:CHANnel<x>:FREQ:SETup:VUNit

Function Sets or queries the unit of velocity when a frequency module is installed in the specified channel (slot).

Syntax
```
:CHANnel<x>:FREQ:SETup:VUNit {<String>}
:CHANnel<x>:FREQ:SETup:VUNit?
```
<x> = 1 to 16
<String> = Up to 4 characters

Example
```
:CHANNEL1:FREQ:SETUP:VUNIT "BBB"
:CHANNEL1:FREQ:SETUP:VUNIT?
-> :CHANNEL1:FREQ:SETUP:VUNIT "BBB"
```

Description An error occurs if a frequency module is not installed.

:CHANnel<x>:FREQ:VDIV

Function Sets or queries the Value/Div when a frequency module is installed in the specified channel (slot).

Syntax
```
:CHANnel<x>:FREQ:VDIV {<NRf>|
<Frequency>|<Time>}
:CHANnel<x>:FREQ:VDIV?
```
<x> = 1 to 16
{<NRf>|<Frequency>|<Time>} = See the SL1000 High Speed Data Acquisition Unit User's Manual for details

Example
```
:CHANNEL1:FREQ:VDIV 10
:CHANNEL1:FREQ:VDIV?
-> :CHANNEL1:FREQ:VDIV 10.0E+00
```

Description An error occurs if a frequency module is not installed.

:CHANnel<x>:LABel

Function Sets or queries the waveform label of the specified channel.

Syntax
```
:CHANnel<x>:LABel {<String>}
:CHANnel<x>:LABel?
```
<x> = 1 to 16
<String> = Up to 8 characters

Example
```
:CHANNEL1:LABEL "TRACE1"
:CHANNEL1:LABEL?
-> :CHANNEL1:LABEL "TRACE1"
```

Description An error occurs if a module is not installed in the channel (slot).

:CHANnel<x>:MODule?

Function Queries the module installed in the channel (slot).

Syntax
```
:CHANnel<x>:MODule?
```
<x> = 1 to 16

Example
```
:CHANNEL1:MODULE?
-> :CHANNEL1:MODULE M701250
```

Description The following values are returned from the module.

String	Model No.
NOMODULE	No module
M720210	720210 (HS100M12)
M720211	720211 (HS100M12)
M701250	701250 (HS10M12)
M720250	720250 (HS10M12)
M701251	701251 (HS1M16)
M701255	701255 (NONISO_10M12)
M701260	701260/67 (HV(with RMS))*
M701268	701268 (HV(with RMS, AAF))
M701261	701261 (UNIV)
M701262	701262 (UNIV_AAF)
M701265	701265 (TEMP/HPV)
M701266	701266 (TEMP/HPV)
M701270	701270 (STRAIN_NDIS)
M701271	701271 (STRAIN_DSUB)
M701275	701275 (ACCL/VOLT)
M701280	701280 (FREQ)
M701281	701281 (FREQ)
M720281	720281 (FREQ)

* "M701260" is returned also for the 701267 module.

:CHANnel<x>:RECord

Function Sets or queries whether to record the specified channel or not.

Syntax
```
:CHANnel<x>:RECord {<Boolean>}
:CHANnel<x>:RECord?
```
<x> = 1 to 16

Example
```
:CHANNEL1:RECORD ON
:CHANNEL1:RECORD? -> :CHANNEL1:RECORD 1
```

Description Specify automatic recording, and the target channel for file saving.
An error occurs if a module is not installed in the channel (slot).

:CHANnel<x>:STRain?

Function Queries all settings related to the Strain Module when a strain module is installed in the specified channel (slot).

Syntax
```
:CHANnel<x>:STRain?
```
<x> = 1 to 16

Description An error occurs if a strain module is not installed.

:CHANnel<x>:STRain:BALance?

Function Queries the balance setting when a strain module is installed in the specified channel (slot).

Syntax
```
:CHANnel<x>:STRain:BALance?
```
<x> = 1 to 16

Description An error occurs if a strain module is not installed.

:CHANnel<x>:STRain:BALance:CHANnel<x>

Function Sets or queries the channel on which balancing is to be executed when a strain module is installed in the specified channel (slot).

Syntax
```
:CHANnel<x>:STRain:BALance:
CHANnel<x> {<Boolean>}
:CHANnel<x>:STRain:BALance:CHANnel?
```
<x> = 1 to 16

Example
```
:CHANNEL1:STRAIN:BALANCE:CHANNEL1 ON
:CHANNEL1:STRAIN:BALANCE:CHANNEL1?
-> :CHANNEL1:STRAIN:BALANCE:CHANNEL1 1
```

Description An error occurs if a strain module is not installed.

:CHANnel<x>:STRain:BALance:EXECute

Function Balances strain when a strain module is installed in the specified channel (slot).

Syntax
```
:CHANnel<x>:STRain:BALance:EXECute
```
<x> = 1 to 16

Example
```
:CHANNEL1:STRAIN:BALANCE:EXECUTE
```

Description Balances channels that are switched on with the :CHANnel<x>:STRain:BALance:CHANnel<x> command.
An error occurs if a strain module is not installed.

:CHANnel<x>:STRain:BWIDth

Function Sets or queries the filter when a strain module is installed in the specified channel (slot).

Syntax
```
:CHANnel<x>:STRain:BWIDth {FULL|
<Frequency>}
:CHANnel<x>:STRain:BWIDth?
```
<x> = 1 to 16
<Frequency> = 10 Hz, 100 Hz, 1 kHz

Example
```
:CHANNEL1:STRAIN:BWIDTH FULL
:CHANNEL1:STRAIN:BWIDTH?
-> :CHANNEL1:STRAIN:BWIDTH FULL
```

Description An error occurs if a strain module is not installed.

:CHANnel<x>:STRain:EXCitation

Function Sets or queries the bridge voltage when a strain module is installed in the specified channel (slot).

Syntax
```
:CHANnel<x>:STRain:
EXCitation {<Voltage>}
:CHANnel<x>:STRain:EXCitation?
```
<x> = 1 to 16
<Voltage> = 2 V, 5 V, 10 V

Example
```
:CHANNEL1:STRAIN:EXCITATION 2V
:CHANNEL1:STRAIN:EXCITATION?
-> :CHANNEL1:STRAIN:
EXCITATION 2.000000E+00
```

Description An error occurs if a strain module is not installed.

:CHANnel<x>:STRain:GFACtor

Function Sets or queries the gauge factor when a strain module is installed in the specified channel (slot).

Syntax
```
:CHANnel<x>:STRain:GFACtor {<NRf>}
:CHANnel<x>:STRain:GFACtor?
```
<x> = 1 to 16
<NRf> = 1.90 to 2.20

Example
```
:CHANNEL1:STRAIN:GFACTOR 2.00
:CHANNEL1:STRAIN:GFACTOR?
-> :CHANNEL1:STRAIN:GFACTOR 2.00
```

Description An error occurs if a strain module is not installed.

:CHANnel<x>:STRain:INVert

Function Sets or queries whether or not the display is inverted when a strain module is installed in the specified channel (slot).

Syntax
```
:CHANnel<x>:STRain:INVert {<Boolean>}
:CHANnel<x>:STRain:INVert?
```
<x> = 1 to 16

Example
```
:CHANNEL1:STRAIN:INVERT ON
:CHANNEL1:STRAIN:INVERT?
-> :CHANNEL1:STRAIN:INVERT 1
```

Description An error occurs if a strain module is not installed.

:CHANnel<x>:STRain:LSCale?

Function Queries all settings related to linear scaling when a strain module is installed in the specified channel (slot).

Syntax
```
:CHANnel<x>:STRain:LSCale?
```
<x> = 1 to 16

Description An error occurs if a strain module is not installed.

:CHANnel<x>:STRain:LSCale:AVALue

Function Sets or queries linear scaling coefficient A when a strain module is installed in the specified channel (slot).

Syntax
```
:CHANnel<x>:STRain:LSCale:
AVALue {<NRf>}
:CHANnel<x>:STRain:LSCale:AVALue?
```
<x> = 1 to 16
<NRf> = -9.9999E+30 to 9.9999E+30

Example
```
:CHANNEL1:STRAIN:LSCALE:AVALUE 10
:CHANNEL1:STRAIN:LSCALE:AVALUE?
-> :CHANNEL1:STRAIN:LSCALE:
AVALUE 10.0000E+00
```

Description An error occurs if a strain module is not installed.

:CHANnel<x>:STRain:LSCale:BVALue

Function Sets or queries offset value B when a strain module is installed in the specified channel (slot).

Syntax
```
:CHANnel<x>:STRain:LSCale:
BVALue {<NRf>}
:CHANnel<x>:STRain:LSCale:BVALue?
```
<x> = 1 to 16
<NRf> = -9.9999E+30 to 9.9999E+30

Example
```
:CHANNEL1:STRAIN:LSCALE:BVALUE 5
:CHANNEL1:STRAIN:LSCALE:BVALUE?
-> :CHANNEL1:STRAIN:LSCALE:
BVALUE 5.00000E+00
```

Description An error occurs if a strain module is not installed.

:CHANnel<x>:STRain:LSCale:DISPlaytype?

Function Queries all settings related to the display type when using linear scaling.

Syntax
```
:CHANnel<x>:STRain:LSCale:DISPlaytype?
```
<x> = 1 to 16

Description An error occurs if a strain module is not installed.

:CHANnel<x>:STRain:LSCale:DISPlaytype:MODE

Function Sets or queries the display format for linear scaling.

Syntax
```
:CHANnel<x>:STRain:LSCale:DISPlaytype:
MODE {EXPonent|FLOating}
:CHANnel<x>:STRain:LSCale:DISPlaytype:
MODE?
```
<x> = 1 to 16

Example
```
:CHANNEL1:STRAIN:LSCALE:DISPLAYTYPE:
MODE EXPONENT
:CHANNEL1:STRAIN:LSCALE:DISPLAYTYPE:
MODE?
-> :CHANNEL1:STRAIN:LSCALE:DISPLAYTYPE:
MODE EXPONENT
```

:CHANnel<x>:STRain:LSCale:DISPlaytype:DECimalnum

Function Sets or queries the decimal place when the display format for linear scaling is set to Floating.

Syntax
```
:CHANnel<x>:STRain:LSCale:DISPlaytype:
DECimalnum {<NRf>|AUTO}
:CHANnel<x>:STRain:LSCale:DISPlaytype:
DECimalnum?
```
<x> = 1 to 16
<NRf> = 0 to 3

Example
```
:CHANNEL1:STRAIN:LSCALE:DISPLAYTYPE:
DECIMALNUM AUTO
:CHANNEL1:STRAIN:LSCALE:DISPLAYTYPE:
DECIMALNUM?
-> :CHANNEL1:STRAIN:LSCALE:DISPLAYTYPE:
DECIMALNUM AUTO
```

:CHANnel<x>:STRain:LSCale:GETMeasure

Function Measures the X values of P1 and P2 for linear scaling when a strain module is installed in the specified channel (slot).

Syntax
```
:CHANnel<x>:STRain:LSCale:
GETMeasure {P1X|P2X}
```
<x> = 1 to 16

Example
```
:CHANNEL1:STRAIN:LSCALE:GETMeasure P1X
```

Description An error occurs if a strain module is not installed.

:CHANnel<x>:STRain:LSCale:MODE

Function Sets or queries the linear scaling method when a strain module is installed in the specified channel (slot). (The method can only be set to SHUNt with a strain module with DSUB, Shunt-Cal.)

Syntax
```
:CHANnel<x>:STRain:LSCale:MODE {AXB|
OFF|P12|SHUNt}
:CHANnel<x>:STRain:LSCale:MODE?
```
<x> = 1 to 16

Example
```
:CHANNEL1:STRAIN:LSCALE:MODE AXB
:CHANNEL1:STRAIN:LSCALE:MODE?
-> :CHANNEL1:STRAIN:LSCALE:MODE AXB
```

Description An error occurs if a strain module is not installed.

:CHANnel<x>:STRain:LSCale:{P1X|P1Y|P2X|P2Y}

Function Sets or queries the X or Y value of P1 or P2 for linear scaling when a strain module is installed in the specified channel (slot).

Syntax
```
:CHANnel<x>:STRain:LSCale:{P1X|P1Y|
P2X|P2Y} {<NRf>}
:CHANnel<x>:STRain:LSCale:{P1X|P1Y|
P2X|P2Y}?
```
<x> = 1 to 16

For P1X and P2X,

<NRf> = -9.9999E+30 to 9.9999E+30

For P1Y and P2Y,

<NRf> = -9.9999E+25 to 9.9999E+25

Example
```
:CHANNEL1:STRAIN:LSCALE:P1X 10
:CHANNEL1:STRAIN:LSCALE:P1X?
-> :CHANNEL1:STRAIN:LSCALE:
P1X 10.0000E+00
```

Description An error occurs if a strain module is not installed.

:CHANnel<x>:STRain:LSCale:SHUNt

Function Executes shunt calibration when a strain module is installed in the specified channel (slot). (This command only works with a strain module with DSUB, Shunt-Cal.)

Syntax
```
:CHANnel<x>:STRain:LSCale:SHUNt
```
<x> = 1 to 16

Example
```
:CHANNEL1:STRAIN:LSCALE:SHUNT
```

Description An error occurs if a strain module is not installed.

:CHANnel<x>:STRain:LSCale:UNIT

Function Sets or queries the unit of measurement to attach to the result of linear scaling when a strain module is installed in the specified channel (slot).

Syntax
```
:CHANnel<x>:STRain:LSCale:
UNIT {<String>}
:CHANnel<x>:STRain:LSCale:UNIT?
```
<x> = 1 to 16
<String> = Up to 4 characters

Example
```
:CHANNEL1:STRAIN:LSCALE:UNIT "X"
:CHANNEL1:STRAIN:LSCALE:UNIT?
-> :CHANNEL1:STRAIN:LSCALE:UNIT "X"
```

Description An error occurs if a strain module is not installed.

:CHANnel<x>:STRain:RANGe

Function Sets or queries the measuring range when a strain module is installed in the specified channel (slot).

Syntax
```
:CHANnel<x>:STRain:RANGe {<NRf>}
:CHANnel<x>:STRain:RANGe?
```
<x> = 1 to 16
<NRf> = 0.25, 0.5, 1, 2.5, 5, 10 (mV/V)
500, 1000, 2000, 5000, 10000, 20000 (uSTR)

Example
```
:CHANNEL1:STRAIN:RANGE 5000
:CHANNEL1:STRAIN:RANGE?
-> :CHANNEL1:STRAIN:RANGE 5000
```

Description An error occurs if a strain module is not installed.

:CHANnel<x>:STRain:UNIT

Function Sets or queries the unit of measurement when a strain module is installed in the specified channel (slot).

Syntax
```
:CHANnel<x>:STRain:UNIT {MV|USTR}
:CHANnel<x>:STRain:UNIT?
```
<x> = 1 to 16

Example
```
:CHANNEL1:STRAIN:UNIT USTR
:CHANNEL1:STRAIN:UNIT?
-> :CHANNEL1:STRAIN:UNIT USTR
```

Description An error occurs if a strain module is not installed.

:CHANnel<x>:TEMPerature?

Function Queries all settings when a temperature, high precision voltage module is installed in the specified channel (slot).

Syntax
```
:CHANnel<x>:TEMPerature?
```
<x> = 1 to 16

Description An error occurs if a temperature, high precision voltage, universal module is not installed

:CHANnel<x>:TEMPerature:BURNout

Function Sets or queries whether or not burnout is detected when a temperature, high precision voltage isolation module is installed in the specified channel (slot).

Syntax
```
:CHANnel<x>:TEMPerature:
BURNout {<Boolean>}
:CHANnel<x>:TEMPerature:BURNout?
```
<x> = 1 to 16

Example
```
:CHANNEL1:TEMPERATURE:BURNOUT ON
:CHANNEL1:TEMPERATURE:BURNOUT?
-> :CHANNEL1:TEMPERATURE:BURNOUT 1
```

Description An error occurs if a temperature, high precision voltage, universal module is not installed.

:CHANnel<x>:TEMPerature:BWIDth

Function Sets or queries the bandwidth limit when a temperature, high precision, voltage nodule is installed in the specified channel (slot).

Syntax
```
:CHANnel<x>:TEMPerature:
BWIDth {FULL|<Frequency>}
:CHANnel<x>:TEMPerature:BWIDth?
```
<x> = 1 to 16

<Frequency> = 2, 8, 30 (Hz)

Example
```
:CHANNEL1:TEMPERATURE:BWIDTH 2.0HZ
:CHANNEL1:TEMPERATURE:BWIDTH?
-> :CHANNEL1:TEMPERATURE:BWIDTH 2.0E+00
```

Description An error occurs if a temperature, high precision voltage, universal module is not installed.

:CHANnel<x>:TEMPerature:COUPling

Function Sets or queries input coupling when a temperature, high precision voltage module is installed in the specified channel (slot).

Syntax
```
:CHANnel<x>:TEMPerature:
COUPling {TC|DC|GND}
:CHANnel<x>TEMPerature:COUPling?
```
<x> = 1 to 16

Example
```
:CHANNEL1:TEMPerature:COUPlING DC
:CHANNEL1:TEMPerature:COUPlING?
-> :CHANNEL1:TEMPerature:COUPlING DC
```

Description An error occurs if a temperature, high precision voltage, universal module is not installed.

:CHANnel<x>:TEMPerature:RJC

Function Sets or queries the RJC when a temperature, high precision voltage module is installed in the specified channel (slot).

Syntax
```
:CHANnel<x>:TEMPerature:RJC {<Boolean>}
:CHANnel<x>:TEMPerature:RJC?
```
<x> = 1 to 16

Example
```
:CHANNEL1:TEMPERATURE:RJC ON
:CHANNEL1:TEMPERATURE:RJC?
-> :CHANNEL1:TEMPERATURE:RJC 1
```

Description An error occurs if a temperature, high precision voltage, universal module is not installed.

:CHANnel<x>:TEMPerature:TYPE

Function Sets or queries the thermocouple type when a temperature, high precision voltage module is installed in the specified channel (slot).

Syntax
```
:CHANnel<x>:TEMPerature:TYPE {K|E|J|T|
L|U|N|R|S|B|W|Au7fe}
:CHANnel<x>:TEMPerature:TYPE?
```
<x> = 1 to 16

Example
```
:CHANNEL1:TEMPERATURE:TYPE K
:CHANNEL1:TEMPERATURE:TYPE?
-> :CHANNEL1:TEMPERATURE:TYPE K
```

Description An error occurs if a temperature, high precision voltage, universal module is not installed.

:CHANnel<x>:TEMPerature:UNIT

Function Sets or queries the unit of measurement values when a temperature, high precision voltage module is installed in the specified channel (slot).

Syntax
```
:CHANnel<x>:TEMPerature:UNIT {C|K}
:CHANnel<x>:TEMPerature:UNIT?
```
<x> = 1 to 16

Example
```
:CHANNEL1:TEMPERATURE:UNIT C
:CHANNEL1:TEMPERATURE:UNIT?
-> :CHANNEL1:TEMPERATURE:UNIT? C
```

Description An error occurs if a temperature, high precision voltage, universal module is not installed.

:CHANnel<x>:UNIT?

Function Queries the unit added to the channel.

Syntax
```
:CHANnel<x>:UNIT?
```
<x> = 1 to 16

Example When a voltage module is installed in channel 7 (slot 3) and the coupling is DC
```
:CHANNEL7:UNIT? -> :CHANNEL7:UNIT "V"
```

:CHANnel<x>:VOLTage?

Function Queries all settings when a voltage module is installed in the specified channel (slot).

Syntax
```
:CHANnel<x>:VOLTage?
```
<x> = 1 to 16

Description An error occurs if a voltage module is not installed.

:CHANnel<x>[:VOLTage]:BWIDth

Function Sets or queries the bandwidth limit when a temperature, high precision voltage isolation module is installed in the specified channel (slot).

Syntax
```
:CHANnel<x>[:VOLTage]:
BWIDth {FULL|<Frequency>}
:CHANnel<x>[:VOLTage]:BWIDth?
```
<x> = 1 to 16

<Frequency> =

720210: 10 kHz, 20 kHz, 40 kHz, 80 kHz, 160 kHz, 640 kHz, 1.28 MHz, 2 MHz

701250, 720250, 701255: 500 Hz, 5 kHz, 50 kHz, 500 kHz

701251: 400 Hz, 4 kHz, 40 kHz

701265: 2 Hz, 8 Hz, 30 Hz

720266: 0.1 Hz, 1 Hz, 8 Hz

720267: 100 Hz, 1 Hz, 10 Hz

701261 and coupling is not set to TC: 40 Hz, 400 Hz, 4 kHz

701262 and coupling is not set to TC: 40 Hz, 400 Hz, 4 kHz, Auto

720268: 400 Hz, 4 kHz, 40 kHz, Auto

Example
```
:CHANNEL1:VOLTAGE:BWIDTH FULL
:CHANNEL1:VOLTAGE:BWIDTH?
-> :CHANNEL1:VOLTAGE:BWIDTH FULL
```

Description An error occurs if a voltage module is not installed.
An error occurs if the coupling is TC.

:CHANnel<x>[:VOLTage]:COUPling

Function Sets or queries input coupling when a voltage module is installed in the specified channel (slot).

Syntax
```
:CHANnel<x>[:VOLTage]:COUPling {AC|DC|
GND|ACRMS|DCRMS|TC}
:CHANnel<x>[:VOLTage]:COUPling?
```
<x> = 1 to 16

Example
```
:CHANNEL1:VOLTAGE:COUPlING DC
:CHANNEL1:VOLTAGE:COUPlING?
-> :CHANNEL1:VOLTAGE:COUPLING DC
```

Description An error occurs if a voltage module is not installed.

701250, 701251, 701255, 720210,720211, 720250: `{AC|DC|GND}`

701260, 701267, 720268: `{AC|DC|GND|ACRMS|DCRMS}`

701261, 701262: `{AC|DC|GND|TC}`

701265, 720266: `{DC|GND|TC}`

701275: `{AC|DC|GND|ACCL}`

:CHANnel<x>[:VOLTage]:INVert

Function Sets or queries whether or not the display is inverted when a voltage module is installed in the specified channel (slot).

Syntax
```
:CHANnel<x>[:VOLTage]:
INVert {<Boolean>}
:CHANnel<x>[:VOLTage]:INVert?
```
<x> = 1 to 16

Example
```
:CHANNEL1:VOLTAGE:INVERT ON
:CHANNEL1:VOLTAGE:INVERT?
-> :CHANNEL1:VOLTAGE:INVERT 1
```

Description An error occurs if a voltage module is not installed.

:CHANnel<x>[:VOLTage]:LSCale?

Function Queries all settings related to linear scaling when a voltage module is installed in the specified channel (slot).

Syntax
```
:CHANnel<x>[:VOLTage]:LSCale?
```
<x> = 1 to 16

Description An error occurs if a voltage module is not installed.

:CHANnel<x>[:VOLTage]:LSCale:AVALue

Function Sets or queries scaling coefficient A of linear scaling when a voltage module is installed in the specified channel (slot).

Syntax
```
:CHANnel<x>[:VOLTage]:LSCale:
AVALue {<NRf>}
:CHANnel<x>[:VOLTage]:LSCale:AVALue?
```
<x> = 1 to 16

<NRf> = -9.9999E+30 to 9.9999E+30

Example
```
:CHANNEL1:VOLTAGE:LSCALE:BVALUE 10
:CHANNEL1:VOLTAGE:LSCALE:BVALUE?
-> :CHANNEL1:VOLTAGE:LSCALE:
BVALUE 10.0000E+00
```

Description An error occurs if a voltage module is not installed.

:CHANnel<x>[:VOLTage]:LSCale:BVALue

Function Sets or queries linear scaling offset value B when a voltage module is installed in the specified channel (slot).

Syntax
```
:CHANnel<x>[:VOLTage]:LSCale:
BVALue {<NRf>}
:CHANnel<x>[:VOLTage]:LSCale:BVALue?
```
<x> = 1 to 16
<NRf> = -9.9999E+30 to 9.9999E+30

Example
```
:CHANNEL1:VOLTAGE:LSCALE:BVALUE 10
:CHANNEL1:VOLTAGE:LSCALE:BVALUE?
-> :CHANNEL1:VOLTAGE:LSCALE:
BVALUE 10.0000E+00
```

Description An error occurs if a voltage module is not installed.

:CHANnel<x>[:VOLTage]:LSCale:DISPlaytype?

Function Queries all settings related to the display type when using linear scaling.

Syntax
```
:CHANnel<x>[:VOLTage]:LSCale:
DISPlaytype?
```
<x> = 1 to 16

:CHANnel<x>[:VOLTage]:LSCale:DISPlaytype:MODE

Function Sets or queries the display format for linear scaling.

Syntax
```
:CHANnel<x>[:VOLTage]:LSCale:
DISPlaytype:MODE {EXPonent|FLOating}
:CHANnel<x>[:VOLTage]:LSCale:D
ISPlaytype:MODE?
```
<x> = 1 to 16

Example
```
:CHANNEL1:VOLTAGE:LSCALE:DISPLAYTYPE:
MODE EXPONENT
:CHANNEL1:VOLTAGE:LSCALE:DISPLAYTYPE:
MODE?
-> :CHANNEL1:VOLTAGE:LSCALE:
DISPLAYTYPE:MODE EXPONENT
```

:CHANnel<x>[:VOLTage]:LSCale:DISPlaytype:DECimalnum

Function Sets or queries the decimal place when the display format for linear scaling is set to Floating.

Syntax
```
:CHANnel<x>[:VOLTage]:LSCale:
DISPlaytype:DECimalnum {<NRf>|AUTO}
:CHANnel<x>[:VOLTage]:LSCale:
DISPlaytype:DECimalnum?
```
<x> = 1 to 16
<NRf> = 0 to 3

Example
```
:CHANNEL1:VOLTAGE:LSCALE:DISPLAYTYPE:
DECIMALNUM AUTO
:CHANNEL1:VOLTAGE:LSCALE:DISPLAYTYPE:
DECIMALNUM?
-> :CHANNEL1:VOLTAGE:LSCALE:
DISPLAYTYPE:DECIMALNUM AUTO
```

:CHANnel<x>[:VOLTage]:LSCale:DISPlaytype:SUBunit

Function Sets or queries the sub unit when the display format for linear scaling is set to Floating.

Syntax
```
:CHANnel<x>[:VOLTage]:LSCale:
DISPlaytype:SUBunit {AUTO|NONE|PICO|
NANO|MICRo|MILI|KILO|MEGA|GIGA|TERA}
:CHANnel<x>[:VOLTage]:LSCale:
DISPlaytype:SUBunit?
```
<x> = 1 to 16

Example
```
:CHANNEL1:VOLTAGE:LSCALE:DISPLAYTYPE:
SUBUNIT AUTO
:CHANNEL1:VOLTAGE:LSCALE:DISPLAYTYPE:
SUBUNIT?
-> :CHANNEL1:VOLTAGE:LSCALE:
DISPLAYTYPE:SUBUNIT AUTO
```

:CHANnel<x>[:VOLTage]:LSCale:GETMeasure

Function Measures the X values of P1 and P2 for linear scaling when a voltage module is installed in the specified channel (slot).

Syntax
```
:CHANnel<x>[:VOLTage]:LSCale:
GETMeasure {P1X|P2X}
```
<x> = 1 to 16

Example
```
:CHANNEL1:VOLTAGE:LSCALE:GETMeasure P1X
```

Description An error occurs if a voltage module is not installed.

:CHANnel<x>[:VOLTage]:LSCale:MODE

Function Sets or queries linear scaling when a voltage module is installed in the specified channel (slot).

Syntax
```
:CHANnel<x>[:VOLTage]:LSCale:
MODE {AXB|OFF|P12}
:CHANnel<x>[:VOLTage]:LSCale:MODE?
```
<x> = 1 to 16

Example
```
:CHANNEL1:VOLTAGE:LSCALE:MODE AXB
:CHANNEL1:VOLTAGE:LSCALE:MODE?
-> :CHANNEL1:VOLTAGE:LSCALE:MODE AXB
```

Description An error occurs if a voltage module is not installed.

:CHANnel<x>[:VOLTage]:LSCale:{P1X|P1Y|P2X|P2Y}

Function Sets or queries the X or Y value of P1 or P2 for linear scaling when a voltage module is installed in the specified channel (slot).

Syntax
```
:CHANnel<x>[:VOLTage]:LSCale:{P1X|P1Y|
P2X|P2Y} {<NRf>}
:CHANnel<x>[:VOLTage]:LSCale:{P1X|P1Y|
P2X|P2Y}?
```
<x> = 1 to 16

For P1X and P2X,
  <NRf> = -9.9999E+30 to 9.9999E+30

For P1Y and P2Y,
  <NRf> = -9.9999E+25 to 9.9999E+25

Example
```
:CHANNEL1:VOLTAGE:LSCALE:P1X 10
:CHANNEL1:VOLTAGE:LSCALE:P1X?
-> :CHANNEL1:VOLTAGE:LSCALE:
P1X 10.0000E+00
```
Description An error occurs if a voltage module is not installed.

:CHANnel<x>[:VOLTage]:LSCale:UNIT

Function Sets or queries the unit of measurement to attach to the result of linear scaling when a voltage module is installed in the specified channel (slot).

Syntax
```
:CHANnel<x>[:VOLTage]:LSCale:
UNIT {<String>}
:CHANnel<x>[:VOLTage]:LSCale:UNIT?
```
<x> = 1 to 16

<String> = Up to 4 characters

Example
```
:CHANNEL1:VOLTAGE:LSCALE:UNIT "RPM"
:CHANNEL1:VOLTAGE:LSCALE:UNIT?
-> :CHANNEL1:VOLTAGE:LSCALE:UNIT "RPM"
```
Description An error occurs if a voltage module is not installed.

:CHANnel<x>[:VOLTage]:PROBe

Function Sets or queries the probe type when a voltage module is installed in the specified channel (slot).

Syntax
```
:CHANnel<x>[:VOLTage]:
PROBe {<NRf>|C10|C100}
:CHANnel<x>[:VOLTage]:PROBe?
```
<x> = 1 to 16

<NRf> = 1, 10, 100, 1000

Example
```
:CHANNEL1:VOLTAGE:PROBE 10
:CHANNEL1:VOLTAGE:PROBE?
-> :CHANNEL1:VOLTAGE:PROBE 10
```
Description An error occurs if a voltage module is not installed. Modules 701265, 701261, 701262 cannot be set or queried.

:CHANnel<x>[:VOLTage]:VDIV

Function Sets or queries the V/div value when a voltage module is installed in the specified channel (slot).

Syntax
```
:CHANnel<x>[:VOLTage]:VDIV {<Voltage>|
<Current>}
:CHANnel<x>[:VOLTage]:VDIV?
```
<x> = 1 to 16

<Voltage> = Depends on module. See the figure below.

Example
```
:CHANNEL1:VOLTAGE:VDIV 5V
:CHANNEL1:VOLTAGE:VDIV?
-> :CHANNEL1:VOLTAGE:VDIV 5.000E+00
```
Description An error occurs if a voltage module is not installed.

Voltage Module and VDIV Range (probe 1:1)

Type	VDIV Range
701250, 701255, 720250	5 mV to 20 V
701251	1 mV to 20 V
701260, 701267, 720268	20 mV to 200 V
701261,701262	5 mV to 20 V
701265	0.1 mV to 10 V
720266	0.1 mV to 20 V
701275	5 mV to 10 V
720210, 720211	10 mV to 20 V

3.8 COMMunicate Group

The COMMunicate group deals with communications.

:COMMunicate?

Function Queries all settings related to communications.

Syntax
```
:COMMunicate?
```

:COMMunicate:HEADer

Function Sets or queries whether or not to add a header to responses to queries.

Syntax
```
:COMMunicate:HEADer {<Boolean>}
:COMMunicate:HEADer?
```

Example
```
:COMMUNICATE:HEADER ON
:COMMUNICATE:HEADER?
-> :COMMUNICATE:HEADER 1
```

:COMMunicate:LOCKout

Function Sets or clears local lockout.

Syntax
```
:COMMunicate:LOCKout {<Boolean>}
:COMMunicate:LOCKout?
```

Example
```
:COMMUNICATE:LOCKOUT ON
:COMMUNICATE:LOCKOUT?
-> :COMMUNICATE:LOCKOUT 1
```

:COMMunicate:OPSE

Function Sets or queries the overlap command that is used by the *OPC, *OPC? and *WAI commands.

Syntax
```
:COMMunicate:OPSE {<Register>}
:COMMunicate:OPSE?
```
<Register> = 0 to 65535

Example
```
:COMMUNICATE:OPSE 65535
:COMMUNICATE:OPSE?
-> :COMMUNICATE:OPSE 584
```

Description Bits fixed to 0 are not set to 1.

:COMMunicate:OPSR?

Function Queries the value of the operation pending status register.

Syntax
```
:COMMunicate:OPSR?
```

Example
```
:COMMUNICATE:OPSR? -> 0
```

Description For details on the operation pending status register, see the figure for the :COMMunicate:WAIT? command.

:COMMunicate:OVERlap

Function Sets or queries the commands to operate as overlap commands.

Syntax
```
:COMMunicate:OVERlap {<Register>}
:COMMunicate:OVERlap?
```
<Register> = 0 to 65535

Example
```
:COMMUNICATE:OVERLAP 65535
:COMMUNICATE:OVERLAP?
-> :COMMUNICATE:OVERLAP 584
```

Description Bits fixed to 0 are not set to 1

:COMMunicate:REMote

Function Sets remote or local. ON is remote mode

Syntax
```
:COMMunicate:REMote {<Boolean>}
:COMMunicate:REMote?
```

Example
```
:COMMUNICATE:REMOTE ON
:COMMUNICATE:REMOTE?
-> :COMMUNICATE:REMOTE 1
```

:COMMunicate:STATus?

Function Queries the line-specific status.

Syntax
```
:COMMunicate:STATus?
```

Example
```
:COMMUNICATE:STATUS?
-> :COMMUNICATE:STATUS 0
```

Description For the SL1000, 0 is always returned.

:COMMunicate:VERBose

Function Sets or queries whether to return the response to a query using full spelling or abbreviations.

Syntax
```
:COMMunicate:VERBose {<Boolean>}
:COMMunicate:VERBose?
```

Example
```
:COMMUNICATE:VERBOSE ON
:COMMUNICATE:VERBOSE?
-> :COMMUNICATE:VERBOSE 1
```

:COMMunicate:WAIT

Function Waits for one of the specified extended events to occur.

Syntax
```
:COMMunicate:WAIT {<Boolean>}
```
<Register> = 0 to 65535

Example
```
:COMMUNICATE:WAIT 65535
```

:COMMunicate:WAIT?

Function Creates the response that is returned when the specified event occurs.

Syntax `:COMMunicate:WAIT? {<Register>}`
<Register> = 0 to 65535 (extended event register)

Example `:COMMUNICATE:WAIT? 65535 -> 1`

Description Operation pending status register/overlap enable register

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	0	0	ACS	0	0	0	0	0	0

When bit 6 (ACS) = 1: Access to the medium not complete.

3.9 CONTrol Group

The CONTrol group deals with operational control for the station.

:CONTrol?

Function Queries all settings related to the station.

Syntax `:CONTrol?`

:CONTrol:FREE?

Function Queries all settings related to free-run measurement.

Syntax `:CONTrol:FREE?`

:CONTrol:FREE:LATCh

Function Executes the latch for the measurement data access for free-run measurement. (Unit firmware holds 'Acq memory writing address' of measurement group 1 data.)

Syntax `:CONTrol:FREE:LATCh`

Example `:CONTROL:FREE:LATCH`

:CONTrol:FREE:LCOunt?

Function Queries the sample counts, which are counted from the start of measurement, at the latch point for free-run measurement.

Syntax `:CONTrol:FREE:LCOunt?`

Example
```
:CONTROL:FREE:LCOUNT?
-> :CONTROL:FREE:LCOUNT 67244
```

Description The function returns the acquisition counts of measurement group 1 using a 64 bit integer value.

:CONTrol:FREE:LENGth<x>?

Function Queries the measured efficient data counts at the latch point for free-run measurement.

Syntax `:CONT:FREE:LENGth<x>?`

<x> = 1 to 4 (Measurement Group)

Example
```
:CONTROL:FREE:LENGTH?
-> :CONTROL:FREE:LENGTH 67244
```

Description The function returns 'the possible maximum acquisition counts' when the acquisition data count exceeds the possible maximum count per a channel for free-run measurement.

:CONTrol:FREE:STIMe?

Function Queries the start time of measurement for free-run measurement.

Syntax `:CONTrol:FREE:STIMe?`

Example
```
:CONTROL:FREE:STIME?
-> :CONTROL:FREE:
STIME 2008,7,10,10,14,18,695967
```

Description The function returns the measurement start time in order of year/month/day/hour/minute/second/microsecond.

:CONTrol:FREE:GDELay<x>?

Function Queries the delay between the measurement start points of each measurement group.

Syntax `:CONTrol:FREE:GDELay<x>?`

<x> = 2 to 4 (Measurement Group)

Example
```
:CONTROL:FREE:GDELAY2?
-> :CONTROL:FREE:GDELAY2 14
```

Description The function returns the delay from measurement groups 2 to 4 from measurement group 1, expressed as measurement points of measurement group 1.

:CONTrol:HDCapacity?

Function Queries the total capacity of internal HDD.

Syntax `:CONTrol:HDCapacity?`

Example
```
:CONTROL:HDCAPACITY?
-> :CONTROL:HDCAPACITY 39053983
```

Description The function returns the value in units of Kbytes.

:CONTrol:HDFRee?

Function Queries the amount of free space in the internal HDD.

Syntax `:CONTrol:HDFRee?`

Example
```
:CONTROL:HDFREE?
-> :CONTROL:HDFREE 39044384
```

Description The function returns the value in units of Kbytes.

3.10 DATa Group

The DATa group deals with waveform data (internal data).

:DATa

Function Queries all settings related to waveform data.

Syntax `:DATa?`

:DATa:BYTeorder

Function Sets or queries the transmission order when using word format of two bytes or more.

Syntax
```
:DATa:BYTeorder {LSBFirst|MSBFirst}
:DATa:BYTeorder?
```

Example
```
:DATA:BYTEORDER LSBFIRST
:DATA:BYTEORDER?
-> :DATA:BYTEORDER LSBFIRST
```

Description This command is effective only for DATA group commands.

:DATa:FRAW?

Function Queries the specified waveform data during free-run measurement The acquisition start point and the acquisition count are specified by the count of the measurement groups to which a specified trace belongs.

Syntax `:DATA:FRAW? {<NRf>,<NRf>,<NRf>}`

First <NRf> = Specify trace (1 to 16)

Second <NRf> = Specify the acquisition start point specified by the count of the measurement groups to which a specified trace belongs.
The specified range is acquired in advance as the sampled counts of Measurement Group 1 with the 'CONTrol:FREE:+COunt?' command.
Specify the data using a 64 bit integer value.

Third <NRf> = Data points specified by the count of the measurement groups to which a specified trace belongs.

Example When acquiring the 1000th data point from the beginning of measurement on CH1
```
:DATA:FRAW? 1, 1000, 1
-> #12(sequence of data byte)
```

Description The function returns "#0" when parameter trouble (trace number, data points etc) and overrun occurs.

3.11 ETHernet Group

The ETHernet group deals with the network.

:ETHernet?

Function Queries all settings related to the network.

Syntax `:ETHernet?`

:ETHernet:TCP?

Function Queries all setting related to TCP.

Syntax `:ETHernet:TCP?`

:ETHernet:TCPip:DHCP

Function Sets or queries DHCP.

Syntax
```
:ETHernet:TCPip:DHCP {<Boolean>}
:ETHernet:TCPip:DHCP?
```

Example
```
:ETHERNET:TCPIP:DHCP ON
:ETHERNET:TCPIP:DHCP?
-> :ETHERNET:TCPIP:DHCP 0
```

:ETHernet:TCPip:GATeway

Function Sets or queries default gateway.

Syntax
```
:ETHernet:TCPip:GATeway {<NRf>,<NRf>,
<NRf>,<NRf>}
:ETHernet:TCPip:GATeway?
```
<NRf> = 0 to 255

Example
```
:ETHERNET:TCPIP:GATEWAY 192,168,0,1
:ETHERNET:TCPIP:GATEWAY?
-> :ETHERNET:TCPIP:GATEWAY 192,168,0,1
```

Description The function returns the default gateway acquired from the DHCP server when DHCP is ON.

:ETHernet:TCPip:IPADdress

Function Sets or queries the IP address.

Syntax
```
:ETHernet:TCPip:IPADdress {<NRf>,<NRf>,
<NRf>,<NRf>}
:ETHernet:TCPip:IPADdress?
```
<NRf> = 0 to 255

Example
```
:ETHERNET:TCPIP:IPADDRESS 192,168,0,2
:ETHERNET:TCPIP:IPADDRESS?
-> :ETHERNET:TCPIP:IPADDRESS 192,168,0,2
```

Description The function returns the IP address acquired from the DHCP server when DHCP is ON.

:ETHernet:TCPip:NETMask

Function Sets or queries the subnet mask.

Syntax
```
:ETHernet:TCPip:NETMask {<NRf>,<NRf>,
<NRf>,<NRf>}
:ETHernet:TCPip:NETMask?
```
<NRf> = 0 to 255

Example
```
:ETHERNET:TCPIP:NETMASK 255,255,252,0
:ETHERNET:TCPIP:NETMASK?
-> :ETHERNET:TCPIP:NETMASK 255,255,252,0
```

Description The function returns the subnet mask acquired from the DHCP server when DHCP is ON.

3.12 FILE Group

The FILE group deals with the internal hard disk.

:FILE?

Function Queries all settings related to the internal hard disk.

Syntax `:FILE?`

:FILE:DELete

Function Deletes files.

Syntax `:FILE:DELete {<Filename>}`

Example `:FILE:DELETE "CASE1.WDF"`

Description The target media to be deleted is selected using FILE[:DIRectory]:DRIVe. The target directory to be deleted is selected using ‘FILE[:DIRectory]: CDIRectory’.
Specify the file name with the extension.

:FILE:DIRectory?

Function Queries all settings related to the directory of the storage media.

Syntax `:FILE:DIRectory?`

:FILE[:DIRectory]:CDIRectory

Function Changes the current directory of the storage media.

Syntax
```
:FILE[:DIRectory]:
CDIRectory {<String>}
```

Example `:FILE:DIRECTORY:CDIRECTORY "NO_1"`

:FILE[:DIRectory]:DRIVe

Function Sets the storage media to be controlled.

Syntax
```
:FILE[:DIRectory]:DRIVe
{ATA,<NRf>,<NRf>}
```
The first <NRf> = Drive number (1 to 9)
The second <NRf> = Partition number (0 to 9)

Example `:FILE:DIRECTORY:DRIVE ATA,1,0`

:FILE[:DIRectory]:FREE?

Function Queries the free disk space (bytes) on the target media.

Syntax `:FILE[:DIRectory]:FREE?`

Example
```
:FILE:DIRECTORY:FREE?
-> :FILE:DIRECTORY:FREE 39.981449E+09
```

:FILE[:DIRectory]:RMDirectory (Remove Directory)

Function Deletes the specified directory in the count directory. This is an overlap command.

Syntax
```
:FILE[:DIRectory]:
RMDirecory {<String>}
```

Example `:FILE:DIRECTORY:RMDIRECTORY "NO_1"`

Description The string is specified with a relative path. Files will be deleted if the target directory to be deleted contains any files.

:FILE[:DIRectory]:MDIRectory

Function Creates a new directory in the current directory.

Syntax
```
:FILE[:DIRectory]:
MDIRectory {<String>}
```

Example `:FILE:DIRECTORY:MDIRECTORY "NO_1"`

Description The string is specified with a relative path.

:FILE[:DIRectory]:PATH?

Function Queries the current directory.

Syntax `:FILE[:DIRectory]:PATH?`

Example
```
:FILE:DIRECTORY:PATH?
-> :FILE:DIRECTORY:PATH "Path =
ATA,1,0"
```

:FILE:LOAD:SETup:ABORt

Function Aborts loading of data.

Syntax `:FILE:LOAD:SETup:ABORt`

Example `:FILE:LOAD:SETUP:ABORT`

:FILE:LOAD:SETup[:EXECute]

Function Loads data. This is an overlap command.

Syntax `:FILE:LOAD:SETup[:EXECute] {<Filename>}`

Example `:FILE:LOAD:SETUP:EXECUTE "CASE1"`

Description Describe <Filename> with 'No extension'.

:FILE:SAVE?

Function Queries all settings related to the saving of files.

Syntax `:FILE:SAVE?`

:FILE:SAVE:ANAMing

Function Sets or queries the auto naming function for saved files.

Syntax
```
:FILE:SAVE:ANAMing {DATE|NUMBering|OFF}
:FILE:SAVE:ANAMing?
```

Example
```
:FILE:SAVE:ANAMING NUMBERING
:FILE:SAVE:ANAMING?
-> :FILE:SAVE:ANAMING NUMBERING
```

Description For DATE, returns the file name in the date and time forrmat.
For NUMbering, returns the auto-numbered name.
For OFF, returns file name of ‘FILE:SAVE:NAME’.

:FILE:SAVE:AREA:MODE

Function Sets or queries the specify save area function.

Syntax
```
:FILE:SAVE:AREA:MODE {<Boolean>}
:FILE:SAVE:AREA:MODE?
```

Example
```
:FILE:SAVE:AREA:MODE ON
:FILE:SAVE:AREA:MODE?
-> :FILE:SAVE:AREA:MODE 1
```

Description The setting value of the ‘FILE:SAVE:AREA:COUNt’ command becomes valid when MODE is ON.

:FILE:SAVE:AREA:COUNt

Function Sets or queries the save area.

Syntax
```
:FILE:SAVE:AREA:COUNt {<NRf>,<NRf>}
:FILE:SAVE:AREA:COUNt?
```
<NRf> = 0 to 134217728

Example
```
:FILE:SAVE:AREA:COUNT 0,10000
:FILE:SAVE:AREA:COUNT?
-> :FILE:SAVE:AREA:COUNT 0,10000
```

:FILE:SAVE:BINary?

Function Queries all settings related to the saving of waveform data.

Syntax
```
:FILE:SAVE:BINary?
```

:FILE:SAVE:{BINary|MEASure|SETup}:ABORt

Function Aborts the saving of data.

Syntax
```
:FILE:SAVE:{BINary|MEASure|SETup}:ABORt
```

Example
```
:FILE:SAVE:BINARY:ABORT
```

:FILE:SAVE:BINary:ACOunt

Function Sets or queries the target acquisition count when saving files.

Syntax
```
:FILE:SAVE:BINary:ACOunt {<NRf>}
:FILE:SAVE:BINary:ACOunt?
```

Example
```
:FILE:SAVE:BINARY:ACOUNT 5
:FILE:SAVE:BINARY:ACOUNT?
-> :FILE:SAVE:BINARY:ACOUNT 5
```

Description Save the latest acquisition count data when 0 is specified .

:FILE:SAVE:{BINary|MEASure|SETup}[:EXECute]

Function Executes the saving of data to a file. This is an overlap command.

Syntax
```
:FILE:SAVE:{BINary|MEASure|SETup}
[:EXECute]
```

Example
```
:FILE:SAVE:BINARY:EXECUTE
```

:FILE:SAVE:BINary:HISTory

Function Sets or queries the save target of the history memory of the data.

Syntax
```
:FILE:SAVE:BINary:HISTORY {ONE|ALL}
:FILE:SAVE:BINary:HISTORY?
```

Example
```
:FILE:SAVE:BINARY:HISTORY ALL
:FILE:SAVE:BINARY:HISTORY?
-> :FILE:SAVE:BINARY:HISTORY ALL
```

:FILE:SAVE:BINary:TALL (Trace All)

Function Sets or queries the selection method of the trace when saving files.

Syntax
```
:FILE:SAVE:BINary:TALL {<Boolean>}
:FILE:SAVE:BINary:TALL?
```
On: Saves all channels for which ON is displayed.
Off: Channel not saved if not specified as ON by CHANnel<x>:RECord, even if the display is ON.

Example
```
:FILE:SAVE:BINARY:TALL ON
:FILE:SAVE:BINARY:TALL?
-> :FILE:SAVE:BINARY:TALL 1
```

:FILE:SAVE:COMMent

Function Sets or queries the comment of data to be saved.

Syntax
```
:FILE:SAVE:COMMent {<String>}
:FILE:SAVE:COMMent?
```
<String> = Up to 250 characters

Example
```
:FILE:SAVE:COMMENT "comment"
:FILE:SAVE:COMMENT?
-> :FILE:SAVE:COMMENT "comment"
```

Description “/”, “\”, “?”, “*”, “:”, “|”, “"”, “<”, “>” cannot be used in the text.

:FILE:SAVE:NAME

Function Sets or queries the name of the file to be saved.

Syntax
```
:FILE:SAVE:NAME {<Filename>}
:FILE:SAVE:NAME?
```

Example
```
:FILE:SAVE:NAME "CASE1"
:FILE:SAVE:NAME?
-> :FILE:SAVE:NAME "CASE1"
```

Description “aux”, “con”, “prn”, “nul”, “clock”, “com1 to com9”, “lpt1 to lpt9” cannot be used as the names of files.

3.13 GONogo Group

The GONogo group deals with GO/NO-GO judgment. GONogo group commands are only valid when the measuring mode is Triggered mode.
You cannot use the GO/NO-GO judgment function during synchronous operation.

:GONogo?

Function Queries all settings related to GO/NO-GO judgment.

Syntax `:GONogo?`

:GONogo:ACONdition

Function Sets or queries the GO/NO-GO judgment action condition.

Syntax
```
:GONogo:ACONdition {ALWays|FAILure|
SUCCess}
:GONogo:ACONdition?
```

Example
```
:GONOGO:ACONDITION FAILURE
:GONOGO:ACONDITION?
-> :GONOGO:ACONDITION FAILURE
```

:GONogo:ACTion?

Function Queries all settings related to the action taken when the execution condition is met.

Syntax `:GONogo:ACTion?`

:GONogo:ACTion:BUZZer

Function Sets or queries whether or not a beep is sounded when the condition is met.

Syntax
```
:GONogo:ACTion:BUZZer {<Boolean>}
:GONogo:ACTion:BUZZer?
```

Example
```
:GONOGO:ACTION:BUZZER OFF
:GONOGO:ACTION:BUZZER?
-> :GONOGO:ACTION:BUZZER 0
```

:GONogo:ACTion:SAVE[:MODE]

Function Sets or queries whether or not waveform data is saved to the storage media when the condition is met.

Syntax
```
:GONogo:ACTion:SAVE:MODE {<Boolean>}
:GONogo:ACTion:SAVE:MODE?
```

Example
```
:GONOGO:ACTION:SAVE:MODE OFF
:GONOGO:ACTION:SAVE:MODE?
-> :GONOGO:ACTION:SAVE:MODE 0
```

:GONogo:AREA

Function Sets or queries the waveform area that is judged.

Syntax
```
:GONogo:AREA {CURSor|FULL}
:GONogo:AREA?
```

Example
```
:GONOGO:AREA CURSOR
:GONOGO:AREA? -> :GONOGO:AREA CURSOR
```

:GONogo:COUNt?

Function Queries the number of performed GO/NO-GO judgments.

Syntax `:GONogo:COUNt?`

Example `:GONOGO:COUNT? -> :GONOGO:COUNT 0`

:GONogo:RSTatus?

Function Queries the most recent GO/NO-GO judgment.

Syntax `:GONogo:RSTatus?`

Example `:GONOGO:RSTATUS? -> :GONOGO:RSTATUS 0`

Description The command returns 0 when the judgment is GO and returns 1 when the judgment is NO-GO.

:GONogo:LOGic

Function Sets or queries the GO/NO-GO logical condition.

Syntax
```
:GONogo:LOGic {AND|OR}
:GONogo:LOGic?
```

Example
```
:GONOGO:LOGIC AND
:GONOGO:LOGIC? -> GONOGO:LOGIC AND
```

:GONogo:MODE

Function Sets or queries the GO/NO-GO judgment mode.

Syntax
```
:GONogo:MODE {OFF|PARameter}
:GONogo:MODE?
```

Example
```
:GONOGO:MODE PARAMETER
:GONOGO:MODE? -> :GONOGO:MODE PARAMETER
```

:GONogo:NGCount?

Function Queries the GO/NO-GO judgment NO-GO count.

Syntax `:GONogo:NGCount?`

Example `:GONOGO:NGCOUNT? -> :GONOGO:NGCOUNT 10`

:GONogo:PARameter?

Function Queries all settings related to parameter judgment.

Syntax `:GONogo:PARameter?`

:GONogo:PARameter:ITEM<x>?

Function Queries all settings related to waveform parameters of the parameter judgment.

Syntax `:GONogo:PARameter:ITEM<x>?`
<x> = 1 to 16

:GONogo:PARameter:ITEM<x>:CAUSe?

Function Queries whether or not the specified waveform parameter is the cause of a NO-GO judgment.

Syntax `:GONogo:PARameter:ITEM<x>:CAUSe?`
<x> = 1 to 16

Example
```
:GONOGO:PARAMETER:ITEM1:CAUSE?
-> :GONOGO:PARAMETER:ITEM1:CAUSE 0
```

Description When the waveform parameter is the cause of a NO-GO judgment, the command returns 1. Otherwise, the command returns 0.

:GONogo:PARameter:ITEM<x>:MODE

Function Sets or queries the judgment criteria of the specified waveform parameter of the parameter judgment.

Syntax
```
:GONogo:PARameter:ITEM<x>:
MODE {OFF|IN|OUT}
:GONogo:PARameter:ITEM<x>:MODE?
```
<x> = 1 to 16

Example
```
:GONOGO:PARAMETER:ITEM1:MODE IN
:GONOGO:PARAMETER:ITEM1:MODE?
-> :GONOGO:PARAMETER:ITEM1:MODE IN
```

:GONogo:PARameter:ITEM<x>:TRACe

Function Sets or queries the target waveform of the measurement of the specified waveform parameters of the parameter judgment.

Syntax
```
:GONogo:PARameter:ITEM<x>:TRACe {<NRf>}
```
<x> = 1 to 16
<NRf> = 1 to 16

Example
```
:GONOGO:PARAMETER:ITEM1:TRACE 1
:GONOGO:PARAMETER:ITEM1:TRACE?
-> :GONOGO:PARAMETER:ITEM1:TRACE 1
```

:GONogo:PARameter:ITEM<x>:TYPE?

Function Queries the item and the upper and lower limits of the measurement of the specified waveform parameter of the parameter judgment.

Syntax
```
:GONogo:PARameter:ITEM<x>:TYPE?
```
<x> = 1 to 16

Example
```
:GONOGO:PARAMETER:ITEM1:TYPE?
-> :GONOGO:PARAMETER:ITEM1:TYPE:
MAXIMUM 1.00000E+00,-1.00000E+00
```

:GONogo:PARameter:ITEM<x>:TYPE:<Parameter>

Function Sets or queries the upper and lower limits of the judgment area for the specified judgment condition.

Syntax
```
: GONogo:PARameter:ITEM<x>:TYPE:
<Parameter>{<{Voltage|DONTcare}>,
<{Voltage|DONTcare}>|<{Current|DONTcare},
<{Current|DONTcare}>|<{Time|DONTcare},
<{Time|DONTcare}>|<{Frequency|DONTcare},
<{Frequency|DONTcare}>|<{<NRf>|DONTcare}>,
<{<NRf>|DONTcare}>}
```
<x> = 1 to 16

<Parameter> =
```
{AMPLitude|AVERage|AVGFreq|
  AVGPeriod|BWIDth1|BWIDth2|DUTYcycle|
  FALL|FREQuency|HIGH|LOW|MAXimum|
  MIDDle|MINimum|NOVershoot|NWIDth|
  PERiod|PNUMber|POVershoot|PTOPeak|
  PWIDth|RISE|RMS|SDEViation|TY1Integ|
  TY2Integ}
```

Example
```
:GONOGO:PARAMETER:ITEM1:TYPE:
MAXIMUM 1V,-1V
:GONOGO:PARAMETER:ITEM1:TYPE:MAXIMUM?
-> :GONOGO:PARAMETER:ITEM1:TYPE:
MAXIMUM 1.00000E+00,-1.00000E+00
```

:GONogo:PARameter:ITEM<x>:VALue?

Function Queries the automated measurement value of the specified GO/NO-GO judgment parameter.

Syntax
```
:GONogo:PARameter:ITEM<x>:VALue?
```
<x> = 1 to 16

Example
```
:GONOGO:PARAMETER:ITEM1:VALUE?
-> :GONOGO:PARAMETER:ITEM1:
VALUE 500.00000E-03
```

:GONogo:PARameter:ITEM<x>:PARam?

Function Queries the measurement item of the specified waveform parameter.

Syntax
```
:GONogo:PARameter:ITEM<x>:PARam?
```
<x> = 1 to 16

Example
```
:GONOGO:PARAMETER:ITEM1:PARAM?
-> :GONOGO:PARAMETER:ITEM1:
PARAM MAXIMUM
```

3.14 HISTory Group

The HISTory group deals with the history memory.

:HISTory?

Function Queries all settings related to the history memory function.

Syntax `:HISTory?`

:HISTory:CLEar

Function Clears all history memory data (all data in memory).

Syntax `:HISTory:CLEar`

Example `:HISTORY:CLEAR`

:HISTory:RECord? MINimum

Function Queries the minimum record number.

Syntax `:HISTory:RECord? MINimum`

Example
```
:HISTORY:RECORD? MINIMUM
-> :HISTORY:RECORD -9
```

Description The function returns the oldest record number currently in the history.
The value is fixed to 0 during measurement.

:HISTory:DATE?

Function Queries the trigger date of the target record number.

Syntax `:HISTory:DATE? {<NRf>|MINimum}`
<NRf> = 0 to 4999

Example
```
:HISTORY:DATE? -1
-> :HISTORY:DATE "-0001 2007/07/09"
```

Description The function returns“--------” when a record number that is smaller than minimum is specified.

:HISTory:TIME?

Function Queries the trigger time of the target record number.

Syntax `:HISTory:TIME? {<NRf>|MINimum}`
<NRf> = 0 to -4999

Example
```
:HISTORY:TIME? -1
-> :HISTORY:TIME "-0001 10:17:09.814"
```

Description The function returns“--------” when a record number that is smaller than minimum is specified.

3.15 INITialize Group

The INITialize group deals with the initialization of settings.

:INITialize:EXECute

Function Initializes settings.

Syntax `:INITialize:EXECute`

Example `:INITIALIZE:EXECUTE`

Description
- The contents not initialized by this command are as follows.
  ETHernet group settings.
  Station name is set using SYSTem:STATion:NAME
  Group name is set using SYSTem:STATion:GNAMe
  Executing this command will clear the history memory.
- Do not send other commands during the initialization. The processing status can be inferred from Bit 9 of the status register.

:INITialize:UNDO

Function Undoes the initialization of settings.

Syntax `:INITialize:UNDO`

Example `:INITIALIZE:UNDO`

Description
- The cleared history data will not be recovered.
- Do not send other commands during the initialization. The processing status can be inferred from Bit 9 of the status register.

3.16 MEASure Group

The MEASure group deals with the automated measurement of waveform parameters. MEASure group commands are only valid when the measuring mode is Triggered mode.

:MEASure?

Function Queries all settings related to the automated measurement of waveform parameters.

Syntax `:MEASure?`

:MEASure:CHANnel<x>?

Function Queries the On/Off state of all of the waveform parameters of the specified channel.

Syntax `:MEASure:CHANnel<x>?`
<x> = 1 to 16

:MEASure:AREA

Function Sets or queries the automatically measured waveform area for the waveform parameters.

Syntax
```
:MEASure:AREA {CURSor|FULL}
:MEASure:AREA?
```

Example
```
:MEASURE:AREA CURSOR
:MEASURE:AREA? -> :MEASURE:AREA CURSOR
```

Description When FULL is specified, the entire history waveform memory is subject to computation. When CURSor is specified, only the range set with MEASsire:CRANge is subject to computation.

:MEASure:MODE

Function Sets or queries the auto measurement mode of the waveform parameter.

Syntax
```
:MEASure:MODE {OFF|ON}
:MEASure:MODE?
```

Example
```
:MEASURE:MODE ON
:MEASURE:MODE? -> :MEASURE:MODE ON
```

:MEASure:CRANge (Count Range)

Function Sets or queries the waveform parameter measurement range.

Syntax
```
:MEASure:CRANge {<NRf>,<NRf>}
:MEASure:CRANge?
```
<NRf> = 0 to 134217728

Example
```
:MEASURE:CRANGE 2500,7500
:MEASURE:CRANGE?
-> :MEASURE:CRANGE 2500,7500
```

:MEASure:CHANnel<x>:METHod

Function Sets or queries the high/low point setting method.

Syntax
```
:MEASure:CHANnel<x>:
METHod {AUTO|MAXMin}
:MEASure:CHANnel<x>:METHod?
```
<x> = 1 to 16

Example
```
:MEASURE:CHANNEL1:METHOD AUTO
:MEASURE:CHANNEL1:METHOD?
-> :MEASURE:CHANNEL1:METHOD AUTO
```

:MEASure:CHANnel<x>:DPRoximal?

Function Queries all settings related to distal, mesial, and proximal.

Syntax `:MEASure:CHANnel<x>:DPRoximal?`
<x> = 1 to 16

:MEASure:CHANnel<x>:DPRoximal:MODE

Function Sets or queries the distal, mesial, and proximal point mode setting.

Syntax
```
:MEASure:CHANnel<x>:DPRoximal:
MODE {PERCent|UNIT}
:MEASure:CHANnel<x>:DPRoximal:MODE?
```
<x> = 1 to 16

Example
```
:MEASURE:CHANNEL1:DPROXIMAL:MODE
PERCENT
:MEASURE:CHANNEL1:DPROXIMAL:MODE?
-> :MEASURE:CHANNEL1:DPROXIMAL:
MODE PERCENT
```

:MEASure:CHANnel<x>:DPRoximal:PERCent

Function Sets or queries the distal, mesial, and proximal points as percentages.

Syntax
```
:MEASure:CHANnel<x>:DPRoximal:
PERCent {<NRf>,<NRf>,<NRf>}
:MEASure:CHANnel<x>:DPRoximal:PERCent?
```
<x> = 1 to 16
<NRf> = 0 to 100 (%, in 0.1% steps)
The values specify the proximal, mesial, and distal points in that order.

Example
```
:MEASURE:CHANNEL1:DPROXIMAL:
PERCENT 40,60,80
:MEASURE:CHANNEL1:DPROXIMAL:PERCENT?
-> :MEASURE:CHANNEL1:DPROXIMAL:
PERCENT 40.0,60.0,80.0
```

:MEASure:CHANnel<x>:DPRoximal:UNIT

Function Sets or queries the distal, mesial, and proximal points.

Syntax
```
:MEASure:CHANnel<x>:DPRoximal:
UNIT {<Voltage>,<Voltage>,<Voltage>|
<Current>,<Current>,<Current>|
<NRf>,<NRf>,<NRf>}
:MEASure:CHANnel<x>:DPRoximal:UNIT?
```
<x> = 1 to 16

The settable ranges of <Voltage>, <Current>, and <NRf> vary depending on the range and offset settings. The values specify the proximal, mesial, and distal points in that order.

Example
```
:MEASURE:CHANNEL1:DPROXIMAL:UNIT
-2V,0V,2V
:MEASURE:CHANNEL1:DPROXIMAL:UNIT?
-> :MEASURE:CHANNEL1:DPROXIMAL:
UNIT -2.000E+00,0.0E+00,2.000E+00
```

:MEASure:CHANnel<x>:<Parameter>?

Function Queries all settings related to the specified waveform parameter (measurement item).

Syntax
```
:MEASure:CHANnel<x>:<Parameter>?
```
<x> = 1 to 16

<Parameter> = {AMPLitude|AVERage|AVGFreq|AVGPeriod|BWIDth1|BWIDth2|DUTYcycle|FALL|FREQuency|HIGH|LOW|MAXimum|MIDDle|MINimum|NOVershoot|NWIDth|PERiod|PNUMber|POVershoot|PTOPeak|PWIDth|RISE|RMS|SDEViation|TY1Integ|TY2Integ}

Example
```
:MEASURE:CHANNEL1:AMPLITUDE?
-> :MEASURE:CHANNEL1:AMPLITUDE:STATE 0
```

Description The reply is the same as the reply of MEASure: CHANnel<x>:< parameter>:STATe?

:MEASure:CHANnel<x>:<Parameter>:STATe

Function Sets or queries the on/off state of the measurement of the specified waveform parameter.

Syntax
```
:MEASure:CHANnel<x>:<Parameter>:
STATe {<Boolean>}
:MEASure:CHANnel<x>:<Parameter>:STATe?
```
<x> = 1 to 16

<Parameter> = {AMPLitude|AVERage|AVGFreq|AVGPeriod|BWIDth1|BWIDth2|DUTYcycle|FALL|FREQuency|HIGH|LOW|MAXimum|MIDDle|MINimum|NOVershoot|NWIDth|PERiod|PNUMber|POVershoot|PTOPeak|PWIDth|RISE|RMS|SDEViation|TY1Integ|TY2Integ}

Example
```
:MEASURE:CHANNEL1:AMPLITUDE:STATE ON
:MEASURE:CHANNEL1:AMPLITUDE:STATE?
-> :MEASURE:CHANNEL1:AMPLITUDE:STATE 1
```

:MEASure:CHANnel<x>:<Parameter>:VALue?

Function Queries the measured value of the specified waveform parameter.

Syntax
```
:MEASure:CHANnel<x>:<Parameter>:VALue?
{<NRf>}
```
<x> = 1 to 16

<NRf> = 1 to 48000

<Parameter> = {AMPLitude|AVERage|AVGFreq|AVGPeriod|BWIDth1|BWIDth2|DUTYcycle|FALL|FREQuency|HIGH|LOW|MAXimum|MIDDle|MINimum|NOVershoot|NWIDth|PERiod|PNUMber|POVershoot|PTOPeak|PWIDth|RISE|RMS|SDEViation|TY1Integ|TY2Integ}

Example
```
:MEASURE:CHANNEL1:AMPLITUDE:VALUE?
-> :MEASURE:CHANNEL1:AMPLITUDE:
VALUE 3.3333333E-03
```

Description
- NAN (not a number) is returned if measuement is not possible. Measurement can be impossible when the automatic measurement mode is not ON, or if the specified range of automatic measurement is not calculated as being 10M or more.
- <NRf> can be omitted. If omitted, the latest history parameter is queried. When <NRf> is used, queries from the latest waveform backward to the <NRf>'th waveform parameter value. If the specified history waveform does not exist, NAN is returned.

:MEASure:FILE?

Function Queries all settings related to the file format output data of automatic measurement results.

Syntax
```
:MEASure:FILE?
```

:MEASure:FILE:TINFomation (Time Information)

Function Sets or queries the ON/OFF state of the addition of the trigger time information in the file format output data of automated measurement results.

Syntax
```
:MEASure:FILE:TINFomation {<Boolean>}
```

Example
```
:MEASURE:FILE:TINFOMATION ON
:MEASURE:FILE:TINFOMATION?
-> :MEASURE:FILE:TINFOMATION 1
```

Description See diagram below.

:MEASure:FILE:SEND?

Function Executes the file format output of the automatic measurement results.

Syntax `:MEASure:SEND?`

Example `:MEASURE:SEND?`

Description Output the following CSV data in “block data format”.

SL1000					
Date*	**Time***	**CH5 P-P**	**CH5 Min**	**CH6 P-P**	**CH6 Min**
		V	**V**	**V**	**V**
2008/04/01	10:00:00:000000	0.0033+00	0.0033+00	0.0033+00	0.0033+00
2008/04/01	10:00:00:000000	0.0033+00	0.0033+00	0.0033+00	0.0033+00
2008/04/01	10:00:00:000000	0.0033+00	0.0033+00	0.0033+00	0.0033+00
2008/04/01	10:00:00:000000	0.0033+00	0.0033+00	0.0033+00	0.0033+00
2008/04/01	10:00:00:000000	0.0033+00	0.0033+00	0.0033+00	0.0033+00
2008/04/01	10:00:00:000000	0.0033+00	0.0033+00	0.0033+00	0.0033+00
2008/04/01	10:00:00:000000	0.0033+00	0.0033+00	0.0033+00	0.0033+00
2008/04/01	10:00:00:000000	0.0033+00	0.0033+00	0.0033+00	0.0033+00
2008/04/01	10:00:00:000000	0.0033+00	0.0033+00	0.0033+00	0.0033+00
2008/04/01	10:00:00:000000	0.0033+00	0.0033+00	0.0033+00	0.0033+00

* These two items will be appended when “with the trigger time information” is selected.

3.17 MONitor Group

The MONitor group deals with numeric monitor output.

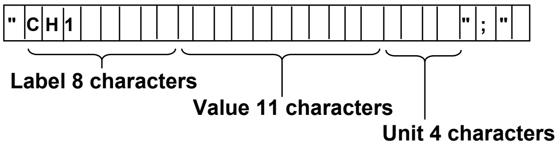
:MONitor:ASENd?

Function Outputs the numeric monitor data (ASCII format) of all channels.

Syntax `:MONitor:ASENd?`

Example `:MONITOR:ASEND? ->` Refer Format below

Description
- Measured values of valid channels are output with each measured value delimited by a semicolon (0x3b). Valid channels means analog channels that are inserted.
- For channels whose display is OFF, the measured value is output as off.
- Outputs the data with 'label' and 'unit' or only the measured value depending on the VERBose setting.
- Format



- The number of unit characters will exceed four if supplementary units exist.

:MONitor:ASENd:CHANnel<x>?

Function Outputs the numeric monitor data (ASCII format) of the specified channel.

Syntax `:MONitor:ASENd:CHANnel<x>?`
<x> = 1 to 16

Example `:MONITOR:ASEND:CHANNEL1?`
`-> "CH1 -1550.0mV"`

Description
- Outputs the data with 'label' and 'unit' or only the measured value depending on the VERBose setting.
- When display is OFF, the measured value is output as off.
- The number of unit characters will exceed four if supplementary units exist.

:MONitor:BITS:CHANnel<x>?

Function Queries the valid bit length of the specified channel.

Syntax `:MONitor:BITS:CHANnel<x>?`
<x> = 1 to 16

Example `:MONITOR:BITS:CHANNEL1?`
`-> :MONITOR:BITS:CHANNEL1 12`

Description For example, the function returns '12' if the specified channel is from the M701250 voltage module (valid Bit length is 12).

:MONitor:BYTeorder

Function Sets or queries the transmission order when using word format of two bytes or more.

Syntax `:MONitor:BYTeorder {LSBFirst|MSBFirst}`
`:MONitor:BYTeorder?`

Example `:MONITOR:BYTEORDER LSBFIRST`
`:MONITOR:BYTEORDER?`
`-> :MONITOR:BYTEORDER LSBFIRST`

Description This setting is effective to output the monitor data using the command, MONitor:SEND:{ALL|CHANnel<x>}.

:MONitor:FORMat:CHANnel<x>?

Function Sets or queries the format of the data to be transmitted .

Syntax `:MONitor:FORMat:CHANnel<x> {BYTE|WORD}`
`:MONitor:FORMat:CHANnel<x>?`
<x> = 1 to 16

Example `:MONITOR:FORMAT:CHANNEL1 WORD`
`:MONITOR:FORMAT:CHANNEL1?`
`-> :MONITOR:FORMAT:CHANNEL1 WORD`

Description This setting is useful when outputting the monitor data using the command, MONitor:SEND:{ALL|CHANnel<x>}

:MONitor:OFFSet:CHANnel<x>?

Function Queries the offset value used to convert the numeric monitor data of the specified channel into physical values.

Syntax `:MONitor:OFFSet:CHANnel<x>?`
<x> = 1 to 16

Example `:MONITOR:OFFSET:CHANNEL1?`
`-> :MONITOR:OFFSET:`
`CHANNEL1 0.0000000E+00`

Description The function returns the value including scaling data when using linear scaling.

:MONitor:GAIN:CHANnel<x>?

Function Queries the gain value used to convert the numeric monitor data of the specified channel into physical values.

Syntax `:MONitor:GAIN:CHANnel<x>?`
<x> = 1 to 16

Example `:MONITOR:GAIN:CHANNEL1?`
`-> :MONITOR:GAIN:CHANNEL1 208.33333E-06`

Description The function returns the value including scaling data when using linear scaling.

:MONitor:RANGe:CHANnel<x>?

Function Queries the range value used to convert the numeric monitor data of the specified channel into physical values.

Syntax `:MONitor:RANGe:CHANnel<x>?`
<x> = 1 to 16

Example `:MONITOR:RANGE:CHANNEL1?`
`-> :MONITOR:RANGE:`
`CHANNEL1 500.00000E-03`

Description The function returns the value including scaling data when using linear scaling.

:MONitor:SEND:{ALL|CHANnel<x>}?

Function Outputs the numeric monitor data in binary format.

Syntax `:MONitor:SEND:{ALL|CHANnel<x>}?`
<x> = 1 to 16

Example `:MON:SEND:CHAN1?`
`-> #?` (? digit byte) (data byte sequence)

Description
- The number of the output byte of the specified channel follows the setting of MONitor:FORMat:CHANnel<x>.
- For “ALL”, the data displays in ascending order. Channels whose displays are “OFF” will not be output.

:MONitor:VERBose

Function Sets or queries whether or not to add ‘label’ and ‘unit’ to the response format of MONitor:ASENd?.

Syntax `:MONitor:VERBose {<Boolean>}`
`:MONitor:VERBose?`

Example `:MONITOR:VERBOSE ON`
`:MONITOR:VERBOSE? -> :MONITOR:VERBOSE 1`

:MONitor:LATCh:ASENd?

Function Outputs the numeric monitor data (ASCII format) of all channels at the latch.

Syntax `:MONitor:LATCh:ASENd?`

Example `:MONITOR:LATCH:ASEND?`
`->` The response format is the same as the format of `:MONitor:ASENd?`

Description The function returns “0” for the measurement value if the latch is not executed.

:MONitor:LATCh:ASENd:CHANnel<x>?

Function Outputs the numeric monitor data (ASCII format) of the specified channels at the latch.

Syntax `:MONitor:LATCh:ASENd:CHANnel<x>?`
<x> = 1 to 16

Example `:MONITOR:LATCH:ASEND:CHANNEL1?`
`-> "CH1 -2396.7mV"`

Description The function returns “0” for the measurement value if the latch is not executed.

:MONitor:LATCh:EXECute

Function Latches the monitor data and the alarm data.

Syntax `:MONitor:LATCh:EXECute`

Example `:MONITOR:LATCH:EXEC`

Description Execute this command before using latch series output commands.

:MONitor:LATCh:SEND:{ALL|CHANnel<x>}?

Function Outputs the numeric monitor data at the latch.

Syntax `:MONitor:LATCh:SEND:{ALL|CHANnel<x>}?`
<x> = 1 to 16

Example `:MONITOR:LATCH:SEND:CHANNEL1?`
`-> #?` (? digit byte) (data byte sequence)

:MONitor:LATCh:ALARm:{ALL|CHANnel<x>}?

Function Outputs the channel alarm data at the latch.

Syntax `:MONitor:LATCh:ALARm:{ALL|CHANnel<x>}?`
<x> = 1 to 16

Example `:MONITOR:LATCH:ALARM:CHANNEL1?`
`-> #?` (? digit byte) (data byte sequence)

Description When “ALL” is specified, the response value returned is the value assigned by bit in channel number order from the leading bit. Transmissions are made in units of bytes. For example, the function returns one byte when the number of mounted channels is 1 to 8, and will return two bytes when the number of mounted channels is 9 to 16.

3.18 MRECord Group

The MRECord group deals with automatic data recording to the internal media.

:MRECord?

Function Queries all settings related to automatic data recording.

Syntax `:MRECord?`

:MRECord:STARt

Function Starts automatic data recording.

Syntax `:MRECord:STARt`

Example `:MRECORD:START`

Description
- When this command is issued, measurement starts at the same time if the measurement is discontinued.
- Please set ":MRECord:DESTination HDD", before starting recording.

:MRECord:STOP

Function Aborts automatic data recording.

Syntax `:MRECord:STOP`

Example `:MRECORD:STOP`

Description Measurement does not stop even if this command is issued.

:MRECord:DESTination

Function Sets or queries the destination of the record of the automatic data recording to the internal media.

Syntax
```
:MRECord:DESTination {HDD}
:MRECord:DESTination?
```

Example
```
:MRECORD:DESTINATION HDD
:MRECORD:DESTINATION?
-> :MRECORD:DESTINATION HDD
```

Description Specify the destination of the record of the automatic data recording using this command when executing automatic data recording.

:MRECord:SCONdition (Start CONdition)

Function Sets or queries the start condition of automatic data recording.

Syntax
```
:MRECord:SCONdition {TIME|ALARm|
ETRise|ETFall|ACQStart}
:MRECord:SCONdition?
```

Example
```
:MRECORD:SCONDITION ACQSTART
:MRECORD:SCONDITION?
-> :MRECORD:SCONDITION ACQSTART
```

:MRECord:STIMe

Function Sets or queries the start time if the start condition is the clock time.

Syntax
```
:MRECord:STIMe {<NRf>,<NRf>,<NRf>,
<NRf>,<NRf>,<NRf>}
:MRECord:STIMe?
```
1st <NRf> = year (2007 to 2099)
2nd <NRf> = month (1 to 12)
3rd <NRf> = day (1 to 31)
4th <NRf> = hour (0 to 23)
5th <NRf> = minute (0 to 59)
6th <NRf> = second (0 to 59)

Example
```
:MRECORD:STIME 2008,7,9,18,30,00
:MRECORD:STIME?
-> :MRECORD:STIME 2008,7,9,18,30,0
```

:MRECord:ECONdition (End CONdition)

Function Sets or queries the end condition of automatic data recording.

Syntax
```
:MRECord:ECONdition {CONTinue|TIME|
ALARm|ETRise|ETFall|ACQStop|TUP}
:MRECord:ECONdition?
```

Example
```
:MRECORD:ECONDITION ACQSTOP
:MRECORD:ECONDITION?
-> :MRECORD:ECONDITION ACQSTOP
```

:MRECord:ETIMe

Function Sets or queries the end time if the end condition is the clock time.

Syntax
```
:MRECord:ETIMe {<NRf>,<NRf>,<NRf>,
<NRf>,<NRf>,<NRf>}
:MRECord:ETIMe?
```
1st <NRf> = year (2007 to 2099)
2nd <NRf> = month (1 to 12)
3rd <NRf> = day (1 to 31)
4th <NRf> = hour (0 to 23)
5th <NRf> = minute (0 to 59)
6th <NRf> = second (0 to 59)

Example
```
:MRECORD:ETIME 2008,7,9,19,00,00
:MRECORD:ETIME?
-> :MRECORD:ETIME 2008,7,9,19,0,0
```

:MRECord:RTIMe

Function Sets or queries the recording time if the end condition is the recording time.

Syntax
```
:MRECord:RTIMe {<NRf>,<NRf>,<NRf>,
<NRf>,<NRf>}
:MRECord:RTIMe?
```
1st <NRf> = day (0 to 30)
2nd <NRf> = hour (0 to 23)
3rd <NRf> = minute (0 to 59)
4th <NRf> = second (0 to 59)
5th <NRf> = millisecond (0 to 999)
Setting range: 0 days 0 hours 0 minutes 0 seconds 001 miliseconds to 30 days 0 hours 0 minutes 0 seconds 000 miliseconds

Example
```
:MRECORD:RTIME 0,1,0,0,0
:MRECORD:RTIME?
-> :MRECORD:RTIME 0,1,0,0,0
```

:MRECord:INTerval?

Function Queries all settings related to the recording interval.

Syntax
```
:MRECord:INTerval?
```

:MRECord:INTerval:TIME

Function Sets or queries the recording interval if the recording interval mode is the TIME.

Syntax
```
:MRECord:INTerval:
TIME {<NRf>,<NRf>,<NRf>}
:MRECord:INTerval:TIME?
```
1st <NRf> = hour (0 to 23)
2nd <NRf> = minute (0 to 59)
3rd <NRf> = second (0 to 59)
Setting range: 0 hours 0 minutes 0 seconds to 24 hours 0 minutes 0 seconds

Example
```
:MRECORD:INTERVAL:TIME 0,30,0
:MRECORD:INTERVAL:TIME?
-> :MRECORD:INTERVAL:TIME 0,30,0
```

:MRECord:INTerval:MODE

Function Sets or queries the recording interval mode of automatic data recording.

Syntax
```
:MRECord:INTerval:MODE {OFF|TIME}
:MRECord:INTerval:MODE?
```

Example
```
:MRECORD:INTERVAL:MODE TIME
:MRECORD:INTERVAL:MODE?
-> :MRECORD:INTERVAL:MODE TIME
```

:MRECord:COUNt

Function Sets or queries the number of recordings of automatic data recording.

Syntax
```
:MRECord:COUNt {<NRf>|INFinity}
:MRECord:COUNt?
```
<NRf> = 1 to 100000

Example
```
:MRECORD:COUNT 10
:MRECORD:COUNT? -> :MRECORD:COUNT 10
```

:MRECord:AREA?

Function Queries all settings related to the method of the recording on the disk.

Syntax
```
:MRECord:AREA?
```

:MRECord:AREA:MODE

Function Sets or queries the method of recording (recording area) on the disk.

Syntax
```
:MRECord:AREA:MODE {CYCLic|SEQuential}
:MRECord:AREA:MODE?
```

Example
```
:MRECORD:AREA:MODE CYCLIC
:MRECORD:AREA:MODE?
-> :MRECORD:AREA:MODE CYCLIC
```

:MRECord:AREA:FNUMber

Function Sets or queries the method of recording (circular file number) on the disk.

Syntax
```
:MRECord:AREA:FNUMber {<NRf>}
:MRECord:AREA:FNUMber?
```
<NRf> = 1 to 2000

Example
```
:MRECORD:AREA:FNUMBER 10
:MRECORD:AREA:FNUMBER?
-> :MRECORD:AREA:FNUMBER 10
```

:MRECord:ECLock?

Function Queries all settings related to recording when using the external sample clock.

Syntax
```
:MRECord:ECLock?
```

:MRECord:ECLock:COUNt

Function Sets or queries the recording counts when using the external sample clock.

Syntax
```
:MRECord:ECLock:COUNt {<NRf>|MAX}
:MRECord:ECLock:COUNt?
```
<NRf> = 10 to 2000000000

Example
```
:MRECORD:ECLOCK:COUNT 100000
:MRECORD:ECLOCK:COUNT?
-> :MRECORD:ECLOCK:COUNT 100000
```

:MRECord:ECLock:INTerval

Function Sets or queries the recording interval counts when using the external sample clock.

Syntax
```
:MRECord:ECLock:INTerval {<NRf>}
:MRECord:ECLock:INTerval?
```
<NRf> = 10 to 1000000

Example
```
:MRECORD:ECLOCK:INTERVAL 150000
:MRECORD:ECLOCK:INTERVAL?
-> :MRECORD:ECLOCK:INTERVAL 150000
```

3.19 MTRigger Group

The MTRigger group deals with manual triggering.

:MTRigger

Function Activates manual triggering.

Syntax `:MTRigger`

Example `:MTRIGGER`

3.20 SELFtest Group

The SELFtest group deals with the self-test.

:SELFtest:HDD:EXECute?

Function Executes the self-test of the internal HDD and outputs the results.

Syntax `:SELFtest:HDD:EXECute?`

Example `:SELFTEST:HDD:EXECUTE?`
`-> :SELFTEST:HDD:EXECUTE 0`

Description The function returns ‘0’ when the self-test is terminated normally, ‘1’ when terminated abnormally, and ‘2’ when an HDD is not mounted.

:SELFtest:HDFormat

Function Formats the internal HDD.

Syntax `:SELFtest:HDFormat`

Example `:SELFTEST:HDFORMAT`

3.21 SSTart Group

The SSTart group deals with the execution of single start. Sets the ACQ operation mode to trigger, the trigger mode to single, and starts waveform acquisition.

:SSTart

Function Executes single start.

Syntax `:SSTart`

Example `:SSTART`

:SSTart? {<NRf>}

Function Executes single start and waits for the completion.

Syntax `:SSTart? {<NRf>}`

<NRf> = 1 to 36000 (100 ms unit: wait time, START and wait)
0 (START only. No wait.)
-36000 to -1 (100 ms resolution: wait period, START and wait)

Example `:SSTART? 100 -> :SSTART 0`

Description
- If the specified time period is positive, data acquisition is started in the SINGLE TRIGGER mode and waits for the operation to stop.
- If the specified time period is 0, data acquisition is started and 0 is returned without waiting for the operation to stop.
- If the specified time period is negative (-), the instrument simply waits for the operation to stop. Data acquisition is not started.

3.22 STARt Group

The START group is used to start waveform acquisition

:STARt

Function The STARt group is used to start waveform acquisition.

Syntax `:STARt`

Example `:START`

Description Use ‘STOP’ to stop waveform acquisition.

3.23 STATus Group

The STATus group deals with the settings and the inquiries related to the status report.

:STATus?

Function Queries all settings related to the communication status function.

Syntax
```
:STATus?
```

:STATus:CONDition?

Function Queries the contents of the condition register.

Syntax
```
:STATus:CONDition?
```

Example
```
:STATUS:CONDITION? -> 4101
```

:STATus:EESE

Function Sets or queries the contents of the extended event enable register.

Syntax
```
:STATus:EESE <Register>
:STATus:EESE?
```
<Register> = 0 to 65535

Example
```
:STATUS:EESE #B00000000
:STATUS:EESE? -> 0
```

:STATus:EESR?

Function Queries the contents of the extended event register and clears the register.

Syntax
```
:STATus:EESR?
```

Example
```
:STATUS:EESR? -> 0
```

:STATus:ERRor?

Function Queries the error code and message information.

Syntax
```
:STATus:ERRor?
```

Example
```
:STATUS:ERROR?
-> 109, "Missing parameter;CHAN1:VDIV"
```

Description
- When there is no error, 0 (No error) is returned.
- You can specify whether or not to add the message using the “STATus:QMESsage” command.

:STATus:FILTer<x>

Function Sets or queries the transition filter.

Syntax
```
:STATus:FILTer<x> {RISE|FALL|BOTH|
NEVer}
:STATus:FILTer<x>?
```
<x> = 1 to 16

Example
```
:STATUS:FILTER2 RISE
:STATUS:FILTER2?
-> :STATUS:FILTER2 RISE
```

Description
- Specify how each bit of the condition register is to change to set the event.
- If RISE is specified, the event is set when the bit changes from 0 to 1.

:STATus:QENable

Function Sets or queries whether or not to store messages other than errors to the Error queue (ON/OFF).

Syntax
```
:STATus:QENable {<Boolean>}
:STATus:QENable?
```

Example
```
:STATUS:QENABLE ON
:STATUS:QENABLE? -> :STATUS:QENABLE 1
```

:STATus:QMESsage

Function Sets or queries whether or not to attach message information to the response to the “STATus:ERRor?” query (ON/OFF).

Syntax
```
:STATus:QMESsage {<Boolean>}
:STATus:QMESsage?
```

Example
```
:STATUS:QMESSAGE ON
:STATUS:QMESSAGE? -> :STATUS:QMESSAGE 1
```

3.24 STOP Group

The STOP group deals with waveform acquisition stop.

:STOP

Function Stops waveform acquisition.

Syntax `:STOP`

Example `:STOP`

Description Recording will stop during the recording when the waveform acquisition is stopped.

3.25 SYSTem Group

The SYStem group deals with the system.

:SYSTem?

Function Queries all settings related to the system.

Syntax `:SYSTem?`

:SYSTem:CLOCk?

Function Queries all settings related to the date and time.

Syntax `:SYSTem:CLOCk?`

:SYSTem:CLOCk:DATE

Function Sets or queries the date.

Syntax
```
:SYSTem:CLOCk:DATE {<String>}
:SYSTem:CLOCk:DATE?
```
<String> = YYYY/MM/DD

YYYY: 2000 to 2099, MM: 1 to 12, DD: 1 to 31

Example
```
:SYSTEM:CLOCK:DATE "2008/07/10"
:SYSTEM:CLOCK:DATE?
-> :SYSTEM:CLOCK:DATE "2008/07/10"
```

:SYSTem:CLOCk:TIME

Function Sets or queries the time.

Syntax
```
:SYSTem:CLOCk:TIME {<String>}
:SYSTem:CLOCk:TIME?
```
<String> = HH:MM:SS

HH: 0 to 23, MM: 0 to 59, SS: 0 to 59

Example
```
:SYSTEM:CLOCK:TIME "13:57:00"
:SYSTEM:CLOCK:TIME?
-> :SYSTEM:CLOCK:TIME "13:57:01"
```

:SYSTem:KEYLock

Function Sets or queries the ON/OFF state of the keylock of the main unit.

Syntax
```
:SYSTem:KEYLock {<Boolean>}
:SYSTem:KEYLock?
```
Example
```
:SYSTEM:KEYLOCK ON
:SYSTEM:KEYLOCK? -> :SYSTEM:KEYLOCK 1
```
Description START/STOP key is locked. The status of the keylock on the main unit is saved even if turning the power to the main unit OFF and ON.

:SYSTem:LCD?

Function Queries all settings related to the LCD.

Syntax `:SYSTem:LCD?`

:SYSTem:LCD:BRIGhtness

Function Sets or queries the brightness of the LCD.

Syntax
```
:SYSTem:LCD:BRIGhtness {<NRf>}
:SYSTem:LCD:BRIGhtness?
```
<NRf> = 1 to 9

Example
```
:SYSTEM:LCD:BRIGHTNESS 3
:SYSTEM:LCD:BRIGHTNESS?
-> :SYSTEM:LCD:BRIGHTNESS 3
```

:SYSTem:LCD:CONTrast

Function Sets or queries the contrast of the LCD.

Syntax
```
:SYSTem:LCD:CONTrast {<NRf>}
:SYSTem:LCD:CONTrast?
```
<NRf> = 1 to 10

Example
```
:SYSTEM:LCD:CONTRAST 4
:SYSTEM:LCD:CONTRAST?
-> :SYSTEM:LCD:CONTRAST 4
```

:SYSTem:LCD:MODE

Function Sets or queries the ON/OFF state of the LCD backlight.

Syntax
```
:SYSTem:LCD:MODE {<Boolean>}
:SYSTem:LCD:MODE?
```
Example
```
:SYSTEM:LCD:MODE ON
:SYSTEM:LCD:MODE? -> :SYSTEM:LCD:MODE 1
```

:SYSTem:LCD:DTOut (Display Time Out)

Function Sets or queries the ON/OFF state of the function that brings the contents of the LCD back to the specified screen in 30 seconds.

Syntax
```
:SYSTem:LCD:DTOut {<Boolean>}
:SYSTem:LCD:DTOut?
```
Example
```
:SYSTEM:LCD:DTOUT ON
:SYSTEM:LCD:DTOUT?
-> :SYSTEM:LCD:DTOUT 1
```

:SYSTem:LCD:DMODe (Display MODe)

Function Sets or queries the contents of LCD.

Syntax
```
:SYSTem:LCD:DMODe {MODule|ERRor|CPARam}
:SYSTem:LCD:DMODe?
```
Example
```
:SYSTEM:LCD:DMODE MODULE
:SYSTEM:LCD:DMODE?
-> :SYSTEM:LCD:DMODE MODULE
```

:SYSTem:RCMode (Remote Control Mode)

Function Sets or queries the measurement stop mode when the remote terminal is controlling START/STOP.

Syntax
```
:SYSTem:RCMode {<Boolean>}
:SYSTem:RCMode?
```

Example
```
:SYSTEM:RCMODE ON
:SYSTEM:RCMODE? -> :SYSTEM:RCMODE 1
```

Description Set “ON” or “1” to stop measurement using Low->High. Set “OFF” or “0” to not stop measurement with the remote terminal. Note that the function that does not stop measurement with the remote terminal is only valid in Trigger mode. Initial value is “1”.

:SYSTem:STATion?

Function Queries all settings related to the unit.

Syntax
```
:SYSTem:STATion?
```

:SYSTem:STATion:GNUMber?

Function Queries the area group number of the unit.

Syntax
```
:SYSTem:STATion:GNUMber?
```

Example
```
:SYSTEM:STATION:GNUMBER?
-> :SYSTEM:STATION:GNUMBER 0
```

Description
- The function returns the rotary switch 1 value (setting of the area group number).
- The function retains and returns the state when the unit is powered ON.

:SYSTem:STATion:MODule<x>?

Function Returns the attributes of the specified slot.

Syntax
```
:SYSTem:STATion:MODule<x>?
```

Example
```
:SYSTEM:STATION:MODULE1?
-> :SYSTEM:STATION:MODULE1 "M720210,
HS100M12,2,1,0x21"
```

Description The function returns the module's kind, abbreviation, number of channels, channel start number, and module FPGA version text. The function returns “NOMODULE, NOMODULE, 0,1,0x11” if no module is installed in the specified slot

String	Abbreviation	Model
NOMODULE	NOMODULE	No module
M701250	HS10M12	701250
M701251	HS1M16	701251
M701255	NONISO 10M12	701255
M701260	HV(with RMS)	701260/67*
M701261	UNIVERSAL	701261
M701262	UNIVERSA(AAF)	701262
M701265	TEMP/HPV	701265
M701270	STRAIN NDIS	701270
M701271	STRAIN DSUB	701271
M701275	ACCL/VOLT	701275
M701280	FREQ	701280
M701281	FREQ	701281
M720281	FREQ	720281
M720210	HS100M12	720210
M720211	HS100M12	720211
M720250	HS10M12	720250
M720266	TEMP/HPV	720266
M720268	HV(with RMS, AAF)	720268

“M701260” is returned also for the 701267 module.

:SYSTem:STATion:NUMBer?

Function Queries the number of the unit.

Syntax
```
:SYSTem:STATion:NUMBer?
```

Example
```
:SYSTEM:STATION:NUMBER?
-> :SYSTEM:STATION:NUMBER 0
```

Description The function returns the value of the rotary switch 2 (setting of the unit number).

Attention The function retains and returns the state when the unit is powered ON.

:SYSTem:STATion:NAME

Function Sets or queries the name of the unit.

Syntax
```
:SYSTem:STATion:NAME {<Strings>}
:SYSTem:STATion:NAME?
```

<Strings> = up to 32 characters (alphabet (capital, lower capital) and numeric number, %,_,(,))

Example
```
:SYSTEM:STATION:NAME "SL1000_1"
:SYSTEM:STATION:NAME?
-> :SYSTEM:STATION:NAME "SL1000_1"
```

:SYSTem:STATion:GNAMe

Function Sets or queries the name of the group.

Syntax
```
:SYSTem:STATion:GNAMe {<Strings>}
:SYSTem:STATion:GNAMe?
```

<Strings> = up to 32 characters (alphabet (capital, lower capital) and numeric number, %,_,(,))

Example
```
:SYSTEM:STATION:GNAME "Group1"
:SYSTEM:STATION:GNAME?
-> :SYSTEM:STATION:GNAME "Group1"
```

3.26 TIMebase Group

The TIMebase group deals with the time base.

:TIMebase?

Function Queries all settings related to the time base.

Syntax `:TIMebase?`

:TIMebase:MODUle<x>:GROUp

Function Sets or queries the measurement group to which the specified module belongs.

Syntax
```
:TIMebase:MODUle<x>:GROUp {<NRf>}
:TIMebase:MODUle<x>:GROUp?
```
<x> = 1 to 8
<NRf> = 1 to 4

Example
```
:TIMEBASE:MODULE1:GROUP 1
:TIMEBASE:MODULE1:GROUP?
-> :TIMEBASE:MODULE1:GROUP 1
```

Description The function returns “0” if the queries references a nonexistent module. One or more modules should be set in the measurement group 1.

:TIMebase:SOURce

Function Sets or queries the time base.

Syntax
```
:TIMebase:SOURce {EXTernal|INTernal}
:TIMebase:SOURce?
```

Example
```
:TIMEBASE:SOURCE INTERNAL
:TIMEBASE:SOURCE?
-> :TIMEBASE:SOURCE INTERNAL
```

:TIMebase:SRATe

Function Sets or queries the sample rate of measurement group 1.

Syntax
```
:TIMebase:SRATe {<Frequency>}
:TIMebase:SRATe?
```
<Frequency> = 5 Hz, 10 Hz, 20 Hz, 50 Hz, 100 Hz, 200 Hz, 500 Hz, 1 kHz, 2 kHz, 5 kHz, 10 kHz, 20 kHz, 50 kHz, 100 kHz, 200 kHz, 500 kHz, 1 MHz, 2 MHz, 5 MHz, 10 MHz, 20 MHz, 50 MHz, 100 MHz

Example
```
:TIMEBASE:SRATE 100MHz
:TIMEBASE:SRATE?
-> :TIMEBASE:SRATE 100.00000E+06
```

:TIMebase:GROUp<x>:SRATe

Function Sets or queries the sample rate of measurement groups 2 to 4.

Syntax
```
:TIMebase:GROUp<x>:SRATe {<Frequency>}
:TIMebase:GROUp<x>:SRATe?
```
<x> = 2 to 4
<Frequency> = 5 Hz, 10 Hz, 20 Hz, 50 Hz, 100 Hz, 200 Hz, 500 Hz, 1 kHz, 2 kHz, 5 kHz, 10 kHz, 20 kHz, 50 kHz, 100 kHz, 200 kHz, 500 kHz, 1 MHz, 2 MHz, 5 MHz, 10 MHz, 20 MHz, 50 MHz, 100 MHz

Example
```
:TIMEBASE:GROUP2:SRATE 100MHz
:TIMEBASE:GROUP2:SRATE?
-> :TIMEBASE:GROUP2:SRATE 100.00000E+06
```

Description The sample rate which is set for measurement groups 2 to 4 can not be set to a faster value than measurement group 1. When the sample rate of measurement group 1 is set to 5 in steps of 1,2,5, the sample rate immediately thereafter (of measurement groups 2 to 4) can not be set to 2 in steps of 1,2,5. For example, you can not set 2 MHz for the sample rate of measurement groups 2 to 4 when the sample rate of measurement group 1 is 5 MHz.

3.27 TRIGger Group

The TRIGer group deals with triggers.

:TRIGger?

Function Queries all settings related to triggers.

Syntax `:TRIGger?`

:TRIGger:COMBination?

Function Queries all settings related to the combination trigger class.

Syntax `:TRIGger:COMBination?`

:TRIGger:COMBination:CHANnel<x>?

Function Queries the settings of the specified channel in the combination trigger class.

Syntax `:TRIGger:COMBination:CHANnel<x>?`

<x> = 1 to 16

:TRIGger:COMBination:CHANnel<x1>:HYSTeresis<x2>

Function Sets or queries the trigger hysteresis of the specified channel in the combination trigger class.

Syntax
```
:TRIGger:COMBination:CHANnel<x1>:
HYSTeresis<x2> {HIGH|LOW|MIDDLE}
:TRIGger:COMBination:CHANnel<x1>:
HYSTeresis<x2>?
```
<x1> = 1 to 16

<x2> = 1, 2

If TYPE is RISE|FALL|HIGH|LOW|BISLope, only 1 is used.

If TYPE is WLIn|WLOut|WINIn|WINOut, both 1 and 2 are used.

(1 is the upper limit. 2 is the lower limit.)

Example
```
:TRIGGER:COMBINATION:CHANNEL1:
HYSTERESIS1 HIGH
:TRIGGER:COMBINATION:CHANNEL1:
HYSTERESIS1?
-> :TRIGGER:COMBINATION:CHANNEL1:
HYSTERESIS1 HIGH
```

:TRIGger:COMBination:CHANnel<x1>:LEVel<x2>

Function Sets or queries the trigger level of the specified channel in the combination trigger class.

Syntax
```
:TRIGger:COMBination:CHANnel<x1>:
LEVel<x2> {<Voltage>|<NRf>|
 <Current>}
:TRIGger:COMBination:CHANnel<x1>:
LEVel<x2>?
```
<x1> = 1 to 16

<x2> = 1, 2

If TYPE is RISE|FALL|HIGH|LOW|BISLope, only 1 is used.

If TYPE is WLIn|WLOut|WINIn|WINOut, both 1 and 2 are used.

(1 is the upper limit. 2 is the lower limit.)

{<Voltage>|<NRf>| <Current>}= Setting range varies by the range and offset settings.

Example
```
:TRIGGER:COMBINATION:CHANNEL1:LEVEL1 10
:TRIGGER:COMBINATION:CHANNEL1:LEVEL1?
->:TRIGGER:COMBINATION:CHANNEL1:
LEVEL1 10.000E+00
```

Description The setting range is the range that can be measured by the current range setting. The setting range is not reflected in the linear scale information.

:TRIGger:COMBination:CHANnel<x>:TYPE

Function Sets or queries the trigger type of the specified channel in the combination trigger class.

Syntax
```
:TRIGger:COMBination:CHANnel<x>:
TYPE {OFF|RISE|FALL|HIGH|LOW|BISLope|
WLIn|WLOut|WINIn|WINOut}
:TRIGger:COMBination:CHANnel<x>:TYPE?
```
<x> = 1 to 16

Example
```
:TRIGGER:COMBINATION:CHANNEL1:TYPE RISE
:TRIGGER:COMBINATION:CHANNEL1:TYPE?
-> :TRIGGER:COMBINATION:CHANNEL1:
TYPE RISE
```

:TRIGger:COMBination:EXTernal:TYPE

Function Sets or queries the external trigger type of the specified channel in the combination trigger class.

Syntax
```
:TRIGger:COMBination:EXTernal:
TYPE {OFF|RISE|FALL}
:TRIGger:COMBination:EXTernal:TYPE?
```

Example
```
:TRIGGER:COMBINATION:EXTERNAL:TYPE RISE
:TRIGGER:COMBINATION:EXTERNAL:TYPE?
-> :TRIGGER:COMBINATION:EXTERNAL:
TYPE RISE
```

:TRIGger:COMBination:MODE

Function Sets or queries the combination mode of the combination trigger class.

Syntax
```
:TRIGger:COMBination:MODE {AND|OR}
:TRIGger:COMBination:MODE?
```

Example
```
:TRIGGER:COMBINATION:MODE AND
:TRIGGER:COMBINATION:MODE?
-> :TRIGGER:COMBINATION:MODE AND
```

:TRIGger:DELay

Function Sets or queries the delay (time from the trigger point to the trigger position).

Syntax
```
:TRIGger:DELay {<Time>}
:TRIGger:DELay?
```

<Time> = 0 to 10 s

Resolution depends on the sample rate.

(1/sample rate10) × (1/10)

Minimum resolution is 10 nsec.

Therefore, 10 nsec is set as the resolution when the sample rate is faster than10 M.

Example
```
:TRIGGER:DELAY 0.1
:TRIGGER:DELAY?
-> :TRIGGER:DELAY 100.00000E-03
```

Description Fixed to 0 when the time base clock is set to external clock.

:TRIGger:HOLDoff?

Function Queries all settings related to the hold off.

Syntax
```
:TRIGger:HOLDoff?
```

:TRIGger:HOLDoff:TIME

Function Sets or queries the hold off time.

Syntax
```
:TRIGger:HOLDoff:TIME {<Time>}
:TRIGger:HOLDoff:TIME?
```

<Time> = 0 to 10 s (resolution 10 ns)

Example
```
:TRIGGER:HOLDOFF:TIME 0.1
:TRIGGER:HOLDOFF:TIME?
-> :TRIGGER:HOLDOFF:TIME 100.00000E-03
```

:TRIGger:MMODe

Function Sets or queries the waveform acquisition mode for trigger measurement.

Syntax
```
:TRIGger:MMODe {NORMal|NSINgle|SINGle}
:TRIGger:MMODe?
```

Example
```
:TRIGGER:MMODE NSINGLE
:TRIGGER:MMODE?
-> :TRIGGER:MMODE NSINGLE
```

:TRIGger:POSition

Function Sets or queries the trigger position.

Syntax
```
:TRIGger:POSition {<NRf>}
:TRIGger:POSition?
```

<NRf> = 0 to 100 (%, 0.1 step)

Example
```
:TRIGGER:POSITION 50
:TRIGGER:POSITION?
-> :TRIGGER:POSITION 50.000
```

:TRIGger[:SIMPle]?

Function Queries all settings related to the simple trigger.

Syntax
```
:TRIGger:SIMPle?
```

:TRIGger[:SIMPle]:HYSTeresis

Function Sets or queries the hysteresis of the simple trigger.

Syntax
```
:TRIGger[:SIMPle]:HYSTeresis {HIGH|
LOW|MIDDle}
:TRIGger[:SIMPle]:HYSTeresis?
```

Example
```
:TRIGGER:SIMPLE:HYSTERESIS HIGH
:TRIGGER:SIMPLE:HYSTERESIS?
-> :TRIGGER:SIMPLE:HYSTERESIS HIGH
```

Description The hysteresis cannot be set or queried when the trigger source is set to EXTernal, LINE, or TIME.

:TRIGger[:SIMPle]:LEVel

Function Sets or queries the trigger level of the simple trigger of the specified channel.

Syntax
```
:TRIGger[:SIMPle]:LEVel {<Voltage>|
<NRf>|<Current>}
:TRIGger[:SIMPle]:LEVel?
```

{<Voltage>|<NRf>| <Current>} = Varies depending on the measurement range and offset setting.

Example
```
:TRIGGER:SIMPLE:LEVEL 1v
:TRIGGER:SIMPLE:LEVEL?
-> :TRIGGER:SIMPLE:LEVEL 1.000E+00
```

Description
- The hysteresis cannot be set or queried when the trigger source is set to EXTernal, LINE, or TIME.
- The setting range is the range that can be measured by the current range setting.
- The setting range is not reflected in the linear scale information.

:TRIGger[:SIMPle]:SLOPe

Function Sets or queries the simple trigger type of the channel specified.

Syntax
```
:TRIGger[:SIMPle]:SLOPe {RISE|FALL|
BISLope}
:TRIGger[:SIMPle]:SLOPe?
```

Example
```
:TRIGGER:SIMPLE:SLOPE RISE
:TRIGGER:SIMPLE:SLOPE?
-> :TRIGGER:SIMPLE:SLOPE RISE
```

Description The simple trigger type cannot set or queried when the trigger source is set to EXTernal, LINE, or TIME.

:TRIGger:SIMPle:SOURce

Function Sets or queries the trigger source of the simple trigger.

Syntax
```
:TRIGger:SIMPle:SOURce {<NRf>|
EXTernal|LINE|TIME}
:TRIGger:SIMPle:SOURce?
```
<NRf> = 1 to 16

Example
```
:TRIGGER:SIMPLE:SOURCE 1
:TRIGGER:SIMPLE:SOURCE?
-> :TRIGGER:SIMPLE:SOURCE 1
```

Description An error occurs if a channel is specified for which no modules are installed.

:TRIGger:TIMer?

Function Queries all settings related to the timer trigger.

Syntax
```
:TRIGger:TIMer?
```

:TRIGger:TIMer:DATE

Function Sets or queries the date of the timer trigger.

Syntax
```
:TRIGger:TIMer:DATE {<String>}
:TRIGger:TIMer:DATE?
```
<String> = YYYY/MM/DD

YYYY: 2000 to 2099, MM: 1 to 12, DD: 1 to 31

Example
```
:TRIGGER:TIMER:DATE "2008/07/01"
:TRIGGER:TIMER:DATE?
-> :TRIGGER:TIMER:DATE "2008/07/01"
```

:TRIGger:TIMer:INTerval

Function Sets or queries the trigger time interval of the timer trigger.

Syntax
```
:TRIGger:TIMer:INTerval {MIN1|MIN2|
MIN3|MIN4|MIN5|MIN6|MIN7|MIN8|MIN9|
MIN10|MIN15|MIN20|MIN25|MIN30|MIN40|
MIN45|MIN50|HOUR1|HOUR2|HOUR3|HOUR4|
HOUR5|HOUR6|HOUR7|HOUR8|HOUR9|HOUR10|
HOUR11|HOUR12|HOUR18|HOUR24}
:TRIGger:TIMer:INTerval?
```

Example
```
:TRIGGER:TIMER:INTERVAL HOUR1
:TRIGGER:TIMER:INTERVAL?
-> :TRIGGER:TIMER:INTERVAL HOUR1
```

:TRIGger:TIMer:TIME

Function Sets or queries the time of the timer trigger.

Syntax
```
:TRIGger:TIMer:TIME {<String>}
:TRIGger:TIMer:TIME?
```
<String> = HH:MM:SS

HH: 0 to 23, MM: 0 to 59, SS: 0 to 59

Example
```
:TRIGGER:TIMER:TIME "12:34:56"
:TRIGGER:TIMER:TIME?
-> :TRIGGER:TIMER:TIME "12:34:56"
```

:TRIGger:TYPE

Function Sets or queries the trigger type.

Syntax
```
:TRIGger:TYPE {COMBination|SIMPle}
:TRIGger:TYPE?
```

Example
```
:TRIGGER:TYPE SIMPLE
:TRIGGER:TYPE? -> :TRIGGER:TYPE SIMPLE
```

Trigger Level Range and Resolution

Excluding Frequency Module (701280, 701281, 720281)

Input	Set Range		Resolution	
Voltage	± (V/div) × 10		V/div	1/100
			Ex. 1 V/div	0.01 V
			500 mV/div	0.005 V
			200 mV/div	0.002 V
Temperature	Every type of measuring range		0.1 (C, K, F All)	
Strain	± (Measuring range)		1 μSTR	
			0.0005 mV/V	
Acceleration	× 0.1	± 100000 unit	0.01 unit	
	× 0.2	± 50000 unit		
	× 0.5	± 20000 unit		
	× 1	± 10000 unit		
	× 2	± 5000 unit		
	× 5	± 2000 unit		
	× 10	± 1000 unit		
	× 20	± 500 unit		
	× 50	± 200 unit		
	× 100	± 100 unit		

Frequency Module (701280, 701281, 720281)

Input	Set Range	Resolution	
Frequency	± (V/div) × 10	0.0005 div or 0.001 div	
		min 0.00 1Hz	
		Ex 1 Hz/div	0.001 Hz
		2 Hz/div	0.002 Hz
		5 Hz/div	0.005 Hz
		10 Hz/div	0.005 Hz
RPM	± (V/div) × 10	0.0005 div or 0.001 div	
		Ex 1000 rpm/div	0.5 rpm
		2000 rpm/div	2 rpm
		5000 rpm/div	5 rpm
RPS	± (V/div) × 10	0.0005 div or 0.001 div	
		Ex 1 rps/div	0.5 mrps
		2 rps/div	0.002 rps
		5 rps/div	0.005 rps
Period	± (V/div) × 10	0.0005 div or 0.001 div	
		Ex 1 msec/div	0.5 μsec
		2 msec/div	2 μsec
		5 msec/div	5 μsec
Duty	± (V/div) × 10	0.001 div	
		1%/div	0.001%
		2%/div	0.002%
		5%/div	0.005%
Power Freq	Center frequency ± (V/div) × 10	2 Hz/div	0.002 Hz
		1 Hz/div	0.001 Hz
		0.5 Hz/div	0.001 Hz
		0.2 Hz/div	0.001 Hz
		0.1 Hz/div	0.001 Hz
Pulse Width	± (V/div) × 10	0.0005 div or 0.001 div	
		Ex 1 msec/div	0.5 μsec
		2 msec/div	2 μsec
		5 msec/div	5 μsec
Integ	± (V/div) × 10	Float Setting	
Velocity	± (V/div) × 10	Float Setting	

3.28 WAVeform Group

The WAVeform group deals with acquired waveform data.

:WAVeform?

Function Queries all settings related to waveform data.

Syntax `:WAVeform?`

:WAVeform:ACOunt?

Function Queries the acquisition count.

Syntax `:WAVeform:ACOunt?`

Example
```
:WAVEFORM:ACOUNT?
-> :WAVEFORM:ACOUNT 10
```

Description The function returns the acquired acquisition count. The function returns "0" when data has not been acquired at least once.

:WAVeform:BITS?

Function Queries the bit length of the waveform data specified by ":WAVeform:TRACe".

Syntax `:WAVeform:BITS?`

Example `:WAVEFORM:BITS? -> :WAVEFORM:BITS 16`

Description
- The bit length which is output with this command is not the same as 'the valid bit length of A/D'.
- For the SL1000, all analog Ch will be '16'.

:WAVeform:BYTeorder

Function Sets or queries the byte order when using word format of two bytes or more.

Syntax
```
:WAVeform:BYTeorder {LSBFirst|MSBFirst}
:WAVeform:BYTeorder?
```

Example
```
:WAVEFORM:BYTEORDER LSBFIRST
:WAVEFORM:BYTEORDER?
-> :WAVEFORM:BYTEORDER LSBFIRST
```

Description This command is valid for only the commands of the WAVEfrom group.

:WAVeform:DIVision?

Function Queries the Division value used when converting the waveform data specified by 'WAVeform:TRACe' to physical values.

Syntax `:WAVeform:DIVision?`

Example
```
:WAVEFORM:DIVISION?
-> :WAVEFORM:DIVISION 24.000000E+03
```

Description
- The data is output per the current 'WAVeform:FORMat' setting of the specified trace.
- The Division value varies according to the kind of input module as follows.

Input	BYTE	WORD
Voltage	93.75	24000
Tempearture	25.6	0.1
Strain	187.5	48000
Acceleration	93.75	24000
Frequency	93.75	24000
RPM	93.75	24000
RPS	93.75	24000
Period	93.75	24000
Duty	93.75	24000
Power Freq	93.75	24000
Pulses Width	93.75	24000
Pulse Integ	93.75	24000
Velocity	93.75	24000

:WAVeform:END

Function Sets or queries the final data point of the waveform (main waveform) specified by 'WAVeform:TRACe'.

Syntax
```
:WAVeform:END {<NRf>}
:WAVeform:END?
```
<NRf> = 0 to (total number of data points-1)

Example
```
:WAVEFORM:END 100000
:WAVEFORM:END? -> :WAVEFORM:END 100000
```

Description The total number of data points can be queried using :WAVeform:CAPTure:LENGth?

:WAVeform:FORMat

Function Sets or queries the format of the data to be transmitted.

Syntax
```
:WAVeform:FORMat {ASCii|BYTE|WORD}
:WAVeform:FORMat?
```

Example
```
:WAVEFORM:FORMAT WORD
:WAVEFORM:FORMAT?
-> :WAVEFORM:FORMAT WORD
```

:WAVeform:GAIN?

Function Queries the gain value used when converting the waveform data specified by 'WAVeform:TRACe' to physical values.

Syntax `:WAVeform:GAIN?`

Example `:WAVEFORM:GAIN?`
`-> :WAVEFORM:GAIN 2.0833333333333E-03`

Description The function returns the value including the scaling data when using linear scale.

:WAVeform:HMAX? (History MAX)

Function Queries the maximum number of history that can be acquired by the currently specified unit.

Syntax `:WAVeform:HMAX?`

Example `:WAVEFORM:HMAX? -> :WAVEFORM:HMAX 64`

:WAVeform:LENGth?

Function Queries all waveform data points (main side) specified by ‘WAVeform:TRACe’.

Syntax `:WAVeform:LENGth?`

Example `:WAVEFORM:LENGTH?`
`-> :WAVEFORM:LENGTH 500500`

Description The function returns the actual number of acquired points when measurement stopped (the points actually acquired from the measurement group to which the specified trace belonged.)

:WAVeform:MODule?

Function Queries the module corresponding to the waveform specified by

Syntax `:WAVeform:MODule?`

Example `:WAVEFORM:MODULE?`
`-> :WAVEFORM:MODULE M720210`

Description The following values are returned from the module.

String	Module
NOMODULE	No Module
M701250	701250
M701251	701251
M701255	701255
M701260	701260/67*
M701261	701261
M701262	701262
M701265	701265
M701270	701270
M701271	701271
M701275	701275
M701280	701280
M701281	701281
M720210	720210
M720211	720211
M720250	720250
M720266	720266
M720268	720268
M720281	720281

* “M701260” is returned also for the 701267 module.

:WAVeform:OFFSet?

Function Queries the offset value used when converting the waveform data specified by :WAVeform:TRACe to physical values.

Syntax `:WAVeform:OFFSet?`

Example `:WAVEFORM:OFFSET?`
`-> :WAVEFORM:OFFSET 0.0000000E+00`

Description The function returns the value including linear scaling data when using linear scale.

:WAVeform:RANGe?

Function Queries the range value when converting the waveform data specified by :WAVeform:TRACe to physical values.

Syntax `:WAVeform:RANGe?`

Example `:WAVEFORM:RANGE?`
`-> :WAVEFORM:RANGE 5.0000000E+00`

Description The function returns the value including linear scaling data when using linear scale.

:WAVeform:RECord

Function Sets or queries the target record number of the main waveform for the commands in the WAVeform group.

Syntax `:WAVeform:RECord {MINimum|<NRf>}`
`:WAVeform:RECord?`
<NRf> = 0 to -4999

Example `:WAVEFORM:RECORD 0`
`:WAVEFORM:RECORD? -> :WAVEFORM:RECORD 0`

Description Specifying ‘MINimum’ sets the record to the minimum record number (the record number of the oldest data). The parameter “0” means the latest record. Please set a minus number to get a historical record.

:WAVeform:RECord? MINimum

Function Queries the minimum record number of the history (main waveform).

Syntax `:WAVeform:RECord? MINimum`

Example `:WAVEFORM:RECORD? MINIMUM`
`-> :WAVEFORM:RECORD -9`

:WAVeform:SEND?

Function Queries the waveform data (raw data) specified by ‘WAVeform:TRACe’.

Syntax `:WAVeform:SEND? {<NRf>}`
<NRf> = 1 to 5000 (varies depending on the measurement time, sample rate setting).

Example `:WAVEFORM:SEND?`
`-> #?` (? digit byte) (data byte sequence) or,
`<NRf>,<NRf>,<NRf>...`

Description <NRf> can be omitted. If <NRf> is included, waveform data is queried <NRf> times in order from the record number specified by ‘WAVeform:RECord’ -<NRf>.

:WAVeform:SIGN?

Function Queries the existence of a sign when querying the waveform data specified by :WAVeform:TRACe using block data.

Syntax `:WAVeform:SIGN?`

Example `:WAVEFORM:SIGN? -> :WAVEFORM:SIGN 1`

Description For the SL1000, ‘1’ is always returned.

:WAVeform:SRATe?

Function Queries the sample rate of the record specified by :WAVeform:RECord.

Syntax `:WAVeform:SRATe?`

Example `:WAVEFORM:SRATE?`
`-> :WAVEFORM:SRATE 5.0000000E+06`

Description The function returns the sample rate of the measurement group to which the trace specified by ‘WAVeform:SRATe’ belongs.

:WAVeform:STARt

Function Sets or queries the first data point of the waveform (main waveform) specified by ‘WAVEform:TRACe’.

Syntax `:WAVeform:STARt {<NRf>}`
`:WAVeform:STARt?`
<NRf> = 0 to (total number of data points - 1)

Example `:WAVEFORM:START 0`
`:WAVEFORM:START? -> :WAVEFORM:START 0`

Description The total number of data points can be queried using :WAVeform:LENGth?

:WAVeform:TRACe

Function Sets or queries the target waveform.

Syntax `:WAVeform:TRACe {<NRf>}`
<NRf> = 1 to 16

Example `:WAVEFORM:TRACE 1`
`:WAVEFORM:TRACE? -> :WAVEFORM:TRACE 1`

Description An error occurs if a module is not installed in the channel.

:WAVeform:TRIGger?

Function Queries the trigger position of the record specified by :WAVeform:RECord.

Syntax `:WAVeform:TRIGger?`

Example `:WAVEFORM:TRIGGER?`
`-> :WAVEFORM:TRIGGER 250000`

Description Queries the number of points from the first point of the record to the trigger position.

:WAVeform:TYPE?

Function Queries the acquisition mode of the waveform specified by :WAVeform:TRACe.

Syntax `:WAVeform:TYPE?`

Example `:WAVEFORM:TYPE?`
`-> :WAVEFORM:TYPE NORMAL`

Description AVERage, ENVelope, BAVerage, or NORMal is returned.

:WAVeform:GDELay?

Function The delay of ‘the measurement start point’ or ‘the trigger point’ of the waveform specified by ‘WAVeform:TRACe’ is returned.

Syntax `:WAVeform:GDELay?`
`:WAveform:GDELay? -> {<NRf>,<NRf>}`
<NRf> = the delay of the measurement start point
<NRf> = the delay of the trigger point

Example `:WAVEFORM:GDELAY?`
`-> :WAVEFORM:GDELAY 3,3`

Description The function converts and returns the delay of the measurement group to which the specified waveform data belogs relative to measurement group 1. The target record number references the setting value of ‘WAVeform:RECord’.

3.29 Common Command Group

The commands in the common command group are defined in IEEE488.2-1987 and are independent of the instrument's functions.

*CAL?

Function Performs calibration and queries the result.
Syntax `*CAL?`
Example `*CAL? -> 0`
Description If the calibration terminates normally, "0" is returned. If an error is detected, "1" is returned.

*CLS

Function Clears the standard event register, extended event register, and error queue.
Syntax `*CLS`
Example `*CLS`
Description If the *CLS command is located immediately after the program message terminator, the output queue is also cleared.

*ESE

Function Sets the standard event enable register or queries the current setting.
Syntax `*ESE {<NRf>}`
`*ESE?`
<NRf> = 0 to 255
Example `*ESE 251`
`*ESE? -> 251`
Description
- Specify the value as a sum of decimal values of each bit.
- For example, specifying "*ESE 251" will cause the standard enable register to be set to "11111011." In this case, bit 2 of the standard event register is disabled which means that bit 5 (ESB) of the status byte register is not set to "1," even if a "query error" occurs.
- The default value is "*ESE 0" (all bits disabled). This result is not saved by the power-off.
- A query using *ESE? will not clear the contents of the standard event enable register.

*ESR?

Function Queries the standard event register and clears the register.
Syntax `*ESR?`
Example `*ESR? 188`
Description
- A sum of decimal values of each bit is returned. You can check what type of events occurred when an SRQ is generated.
- For example, if a value of "32" is returned, this indicates that the standard event register is set to "00100000." In this case, you can see that the SRQ occurred due to a "command syntax error."
- A query using *ESR? will clear the contents of the standard event register.

*IDN?

Function Queries the instrument model.
Syntax `*IDN?`
Example `*IDN? -> YOKOGAWA,720120,0,F1.10`
Description The information is returned in the following form: <Manufacturer>,<Model>,<Serial No.>,<Firmware version>.

*OPC

Function Sets a "1" to bit 0 (OPC bit) of the standard event register bit upon the completion of the specified overlap command.
Syntax `*OPC`
Example `*OPC`
Description The COMMunicate:OPSE command is used to specify the overlap command. If *OPC is not the last command of the message, the operation is not guaranteed.

*OPC?

Function The specified overlap command is completed, ASCII code '1' is returned.
Syntax `*OPC?`
Example `*OPC? -> 1`
Description The COMMunicate:OPSE command is used to specify the overlap command. If *OPC? is not the last command of the message, the operation is not guaranteed.

*OPT?

Function Queries the installed options.
Syntax `*OPT?`
Example `*OPT? -> 128MW,HD,PROBEPOWER`
Description
- The information of installed memory length: Total capacity of the waveform data memory '128MW'.
- Installing the internal HD: 'HD'
- Loading the probe power option: 'PROBEPOWER'
- Loading the ether net option: 'ETHER'

*RST

Function Initializes the current settings.
Syntax `*RST`
Example `*RST`
Description Also clears the *OPC and *OPC? commands that were sent earlier.

*SRE

Function Sets or queries the service request enable register value.

Syntax
```
*SRE {<NRf>}
*SRE?
```
<NRf> = 0 to 255

Example
```
*SRE 239
*SRE? -> 175
```

Description
- Specify the value as a sum of decimal values of each bit.
- For example, specifying "*SRE 239" will cause the service request enable register to be set to "11101111." In this case, bit 4 of the service request enable register is disabled which means that bit 4 (MAV) of the status byte register is not set to "1," even if "the output queue is not empty."
  Bit 6 (MSS) of the status byte register is the MSS bit itself, and therefore, is ignored.
- The default value is "*SRE 0" (all bits disabled). This result is not saved by the power-off.
- A query using *SRE? will not clear the contents of the service request enable register.

*STB? (STatus Byte)

Function Queries the status byte register.

Syntax
```
*STB?
```

Example
```
*STB? -> 4
```

Description
- The sum of the bits is returned as a decimal value.
- Since the register is read without executing serial polling, bit 6 is a MSS bit not RQS.
  For example, if a value of "4" is returned, this indicates that the status byte register is set to "00000100." In this case, you can see that "the error queue is not empty" (an error occurred).
- A query using *STB? will not clear the contents of the status byte register.

*TST?

Function Performs a self-test and queries the result.

Syntax
```
*TST?
```

Example
```
*TST? -> 0
```

Description
- The test is of each internal memory, and is executed in the order: system memory test, register test, and Acq memory test.
- If the test passes, the command returns 0. If the test fails, the command returns 1.

*WAI

Function Holds the subsequent command until the completion of the specified overlap operation.

Syntax
```
*WAI
```

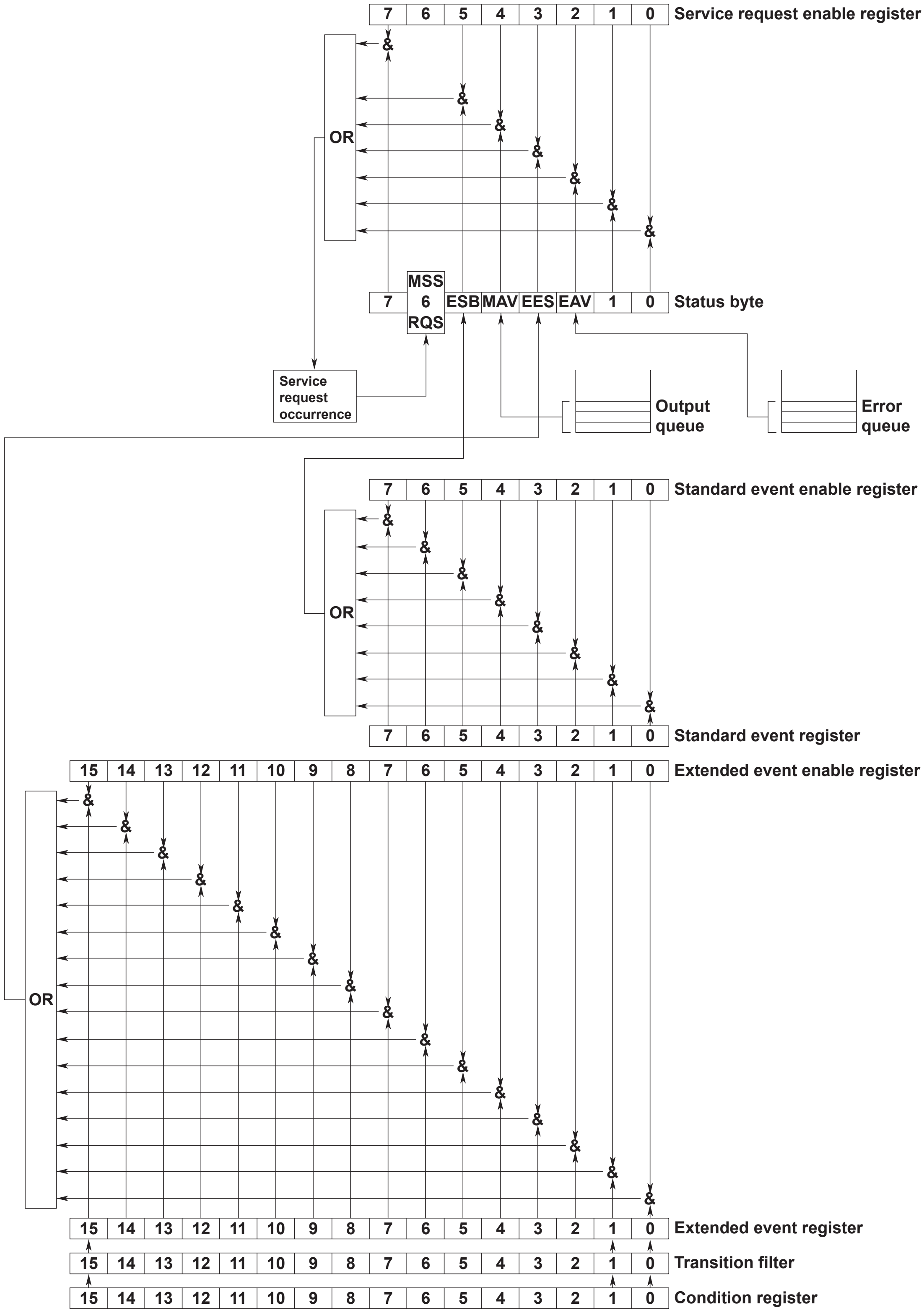
Example
```
*WAI
```

Description The COMMunicate:OPSE command is used to specify the overlap command.

4.1 Overview of the Status Report

Status Reports

The figure below shows the status report that is read by serial polling. This status report is an extended version of the status report defined in IEEE 488.2-1992



Overview of the Registers and Queues

Name	Functions	Writing	Reading
Status byte	–	–	Serial polling (RQS) *STB?(MSS)
Service request enable register	Masks status byte	*SRE	*SRE?
Standard event register	Changes in device status	–	*ESR?
Standard event enable register	Masks standard event register	*ESE	*ESE?
Extended event register	Changes in device status	–	STATus:EESR?
Extended event enable register	Masks extended event register	STATus:EESE	STATus:EESE?
Condition register	Current instrument status	–	STATus:CONDition?
Transition filter	Conditions that change the extended event register	STATus:FILTer<x>	STATus:FILTer<x>?
Output queue	Stores a response message to a query	All query commands	
Error queue	Stores the error number and message	–	STATus:ERRor?

Registers and Queues That Affect the Status Byte

Registers that affect the bits of the status byte are shown below.

Standard event register	Sets bit 5 (ESB) of the status byte to 1 or 0.
Output queue	Sets bit 4 (MAV) of the status byte to 1 or 0.
Extended event register	Sets bit 3 (EES) of the status byte to 1 or 0.
Error queue	Sets bit 2 (EAV) of the status byte to 1 or 0.

Enable Registers

Registers that are used to mask a bit so that the bit will not affect the status byte even when it is set to 1, are shown below.

Status byte	Mask the bits using the service request enable register.
Standard event register	Mask the bits using the standard event enable register.
Extended event register	Mask the bits using the extended event enable register.

Writing/Reading from Registers

The *ESE command is used to set the bits in the standard event enable register to 1's or 0's. The *ESE? command is used to query whether the bits in the standard event enable register are 1's or 0's. For details regarding these commands, see chapter 3.

4.2 Status Byte

Status Byt

	RQS						
7	6	ESB	MAV	EES	EAV	1	0
	MSS						

- **Bits 0 and 7**
  Not used (always 0)
- **Bits 1**
  Not used (always 0)
- **Bit 2 EAV (Error Available)**
  Set to 1 when the error queue is not empty. In other words, this bit is set to 1 when an error occurs. See the page 4-6.
- **Bit 3 EES (Extend Event Summary Bit)**
  Set to 0 when the logical product of the extended event register and the corresponding enable register is 1. In other words, this bit is set to 1 when an event takes place inside the instrument. See the page 4-5.
- **Bit 4 MAV (Message Available)**
  Set to “1” when the output queue is not empty. In other words, this bit is set to 1 when there are data to be transmitted. See the page 4-6.
- **Bit 5 ESB (Event Summary Bit)**
  Set to 0 when the logical product of the standard event register and the corresponding enable register is 1. In other words, this bit is set to 1 when an event takes place inside the instrument. See the next page.
- **Bit 6 RQS(Request Service)/**
  MSS(Master Status Summary)
  Set to 1 when the logical AND of the status byte excluding Bit 6 and the service request enable register is not 0. In other words, this bit is set to 1 when the instrument is requesting service from the controller.
  RQS is set to 1 when the MSS bit changes from 0 to 1, and cleared when serial polling is carried out or when the MSS bit changes to 0.

Bit Masking

To mask a bit in the status byte so that it does not cause an SRQ, set the corresponding bit of the service request enable register to 0.
For example, to mask bit 2 (EAV) so that service is not requested when an error occurs, set bit 2 of the service request enable register to 0. This can be done using the *SRE command. To query whether each bit of the service request enable register is 1 or 0, use *SRE?. For details on the *SRE command, see chapter 3.

Operation of the Status Byte

A service request is issued when bit 6 of the status byte becomes 1. Bit 6 is set to 1 when any of the other bits becomes a 1 (when the corresponding bit of the service request enable register is also set to 1).
For example, if an event occurs and the logical AND of the standard event register and the corresponding enable register becomes a 1, then bit 5 (ESB) is set to 1. In this case, if bit 5 of the service request enable register is 1, bit 6 (MSS) will be set to 1, thus requesting service from the controller.
In addition, you can also check what type of event occurred by reading the contents of the status byte.

Reading from the Status Byte

The following two methods are provided for reading the status byte.

- **Inquiry using the *STB? query**
  Making an inquiry using the *STB? query sets bit 6 to MSS. This causes the MSS to be read. After completion of the read-out, none of the bits in the status byte will be cleared.
- **Serial polling**
  Execution of a serial polling changes bit 6 to RQS. This causes RQS to be read. After completion of the read-out, only RQS is cleared. It is not possible to read MSS using serial polling.

Clearing the Status Byte

No method is provided for forcibly clearing all the bits in the status byte. The bits that are cleared for each operation are shown below.

- **When a query is made using the *STB? command**
  No bits are cleared.
- **When serial polling is executed**
  Only the RQS bit is cleared.
- **When a *CLS command is received.**
  When the *CLS command is received, the status byte itself is not cleared, but the contents of the standard event register (which affects the bits in the status byte) are cleared. As a result, the corresponding bits in the status byte are cleared, except bit 4 (MAV), since the output queue cannot be emptied by the *CLS command. However, the output queue will also be cleared if the *CLS command is received just after a program message terminator.

4.3 Standard Event Register

Standard Event Registr

7	6	5	4	3	2	1	0
PON	URQ	CME	EXE	DDE	QYE	RQC	OPC

- **Bit 7 PON (Power ON)**
  Set to 1 when the power is turned ON.
- **Bit 6 URQ (User Request)**
  Not used (always 0)
- **Bit 5 CME (Command Error)**
  Set to 1 when the command syntax is incorrect.
  Example Incorrectly spelled command name; "9" used in octal data.
- **Bit 4 EXE (Execution Error)**
  Set to 1 when the command syntax is correct but the command cannot be executed in the current state.
  Example Received a command with a parameter outside the range or attempted to output a hard copy while waveform acquisition is in progress.
- **Bit 3 DDE (Device Dependent Error)**
  Set to 1 when execution of the command is not possible due to an internal problem in the instrument that is not a command error or an execution error.
- **Bit 2 QYE (Query Error)**
  Set to 1 if the output queue is empty or if the data is missing even after a query has been sent.
  Example No response data; data is lost due to an overflow in the output queue.
- **Bit 1 RQC (Request Control)**
  Not used (always 0)
- **Bit 0 OPC (Operation Complete)**
  Set to 1 when the operation designated by the *OPC command (see chapter 3) has been completed.

Bit Masking

To mask a bit in the standard event register so that it does not cause bit 5 (ESB) of the status byte to change, set the corresponding bit in the standard event enable register to 0. Refer to Chapter 2.
For example, to mask bit 2 (QYE) so that ESB will not be set to 1, even if a query error occurs, set bit 2 of the standard event enable register to 0. This can be done using the *ESE command. To inquire whether each bit of the standard event enable register is 1 or 0, use the *ESE?. For details on the *ESE command, see chapter 3.

Operation of the Standard Event Register

The standard event register is provided for eight different kinds of event which can occur inside the instrument. Bit 5 (ESB) of the status byte is set to 1 when any of the bits in this register becomes 1 (or when the corresponding bit of the standard event enable register becomes 1).
Example

1. A query error occurs.
2. Bit 2 (QYE) is set to 1.
3. Bit 5 (ESB) of the status byte is set to 1 if bit 2 of the standard event enable register is 1.

It is also possible to check what type of event has occurred inside the instrument by reading the contents of the standard event register.

Reading from the Standard Event Register

The contents of the standard event register can be read by the *ESR command. After the register is read, it is cleared.

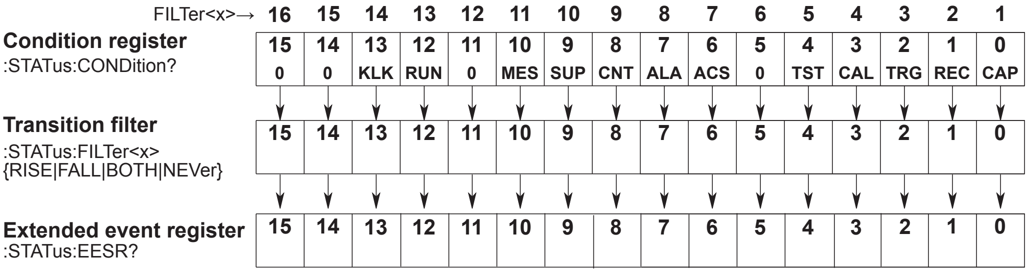
Clearing the Standard Event Register

The standard event register is cleared in the following three cases.

- When the contents of the standard event register are read using the *ESR command.
- When a *CLS command is received.
- When the instrument is power cycled.

4.4 Extended Event Register

Reading the extended event register tells you whether changes in the condition register (reflecting internal conditions) have occurred. A filter can be applied which allows you to decide which events are reported to the extended event registe.



FILTer<x>→	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1
Condition register :STATus:CONDition?	15 0	14 0	13 KLK	12 RUN	11 0	10 MES	9 SUP	8 CNT	7 ALA	6 ACS	5 0	4 TST	3 CAL	2 TRG	1 REC	0 CAP
Transition filter :STATus:FILTer<x> {RISE|FALL|BOTH|NEVer}	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Extended event register :STATus:EESR?	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

The meaning of each bit of the condition register is as follows:

Bit 0	CAP(Capture)	Set to 1 while waveform acquisition is in progress (including trigger wait and during pretrigger). .
Bit 1	REC	Set to 1 while recording is in progress..
Bit 2	TRG (Awaiting trigger)	Set to 1 when waiting for a trigger.
Bit 3	CAL (Calibration)	Set to 1 while calibration is in progress.
Bit 4	TST (Testing)	Set to 1 while self-test is in progress.
Bit 5	Reserved	–
Bit 6	ACS (Accessing)	Set to 1 while a storage drive is being accessed.
Bit 7	ALA (Alarming)	Set to 1 Set to 1 during system alarms.
Bit 8	CNT (CONecting)	Set to 1 while connected from a host.
Bit 9	SUP (Setup)	Set to 1 while auto setup or initialization is in progress.
Bit 10	MES (Measuring)	Set to 1 when automated measurement of waveform parameters or GO/NOGO is in progress.
Bit 11	Reserved	–
Bit 12	RUN (Running))	Set to 1 during measurement (including during operations linked with the lighting of the start key, for example during sequential and other computations).
Bit 13	KLK (Key lock)	Set to 1 while the key lock is engaged.
Bit 14	Reserved	–
Bit 15	Reserved	–

The transition filter parameters detect changes in the specified bit (numerical suffix, 1 to 16) of the condition register in the following manner and overwrite the extended event register.

RISE	The specified bit of the extended event register is set to 1 when the bit of the condition register changes from 1 to 0.
FALL	The specified bit of the extended event register is set to 1 when the bit of the condition register changes from 0 to 1.
BOTH	The bit of the extended event register is set to 1 when the bit of the condition register changes from 0 to 1 or from 1 to 0.
NEVer	Always 0.

4.5 Output Queue and Error Queue

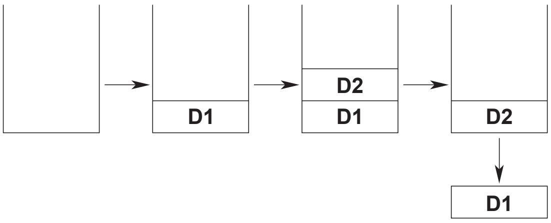
Output Queue

The output queue is provided to store response messages to queries. For example, if you send the WAVeform:SEND? command, which requests the output of acquired data, the data is stored in the output queue until it is read.

As shown below, data are stored in order and read from the oldest ones first. The output queue is emptied in the following cases (in addition to when read-out is performed).

- When a new message is received from the controller.
- When a deadlock occurs.
- When a device clear command (DCL or SDC) is received.
- When the instrument is power cycled.

The output queue cannot be emptied using the *CLS command. To see whether the output queue is empty or not, check bit 4 (MAV) of the status byt.



Error Queue

The error queue stores the error No. and message when an error occurs. For example, if the controller sends an incorrect program message, the error number and message "113, "Undefined header"" are stored in the error queue when the error is displayed.

The STATus:ERRor? query can be used to read the contents of the error queue. As with the output queue, the messages are read from the oldest ones first.

When the error queue overflows, the last message is replaced by the message "350, "Queue overflow"."

The error queue is also cleared for the following cases:

- When a *CLS command is received.
- When the instrument is power cycled.

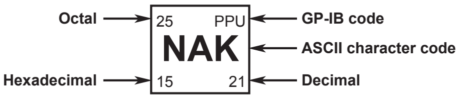
To see whether the error queue is empty or not, check bit 2 (EAV) of the status byte.

Appendix 1 ASCII Character Codes

The following table shows the ASCII character codes.

	0	1	2	3	4	5	6	7
0	0 NUL 0 0	20 DEL 10 16	40 0 SP 20 32	60 16 0 30 48	100 0 @ 40 64	120 16 P 50 80	140 0 ‘ 60 96	160 16 p 70 112
1	1 GTL SOH 1 1	21 LLO DC1 11 17	41 1 ! 21 33	61 17 1 31 49	101 1 A 41 65	121 17 Q 51 81	141 1 a 61 97	161 17 q 71 113
2	2 STX 2 2	22 DC2 12 18	42 2 ” 22 34	62 18 2 32 50	102 2 B 42 66	122 18 R 52 82	142 2 b 62 98	162 18 r 72 114
3	3 ETX 3 3	23 DC3 13 19	43 3 # 23 35	63 19 3 33 51	103 3 C 43 67	123 19 S 53 83	143 3 c 63 99	163 19 s 73 115
4	4 SDC EOT 4 4	24 DCL DC4 14 20	44 4 $ 24 36	64 20 4 34 52	104 4 D 44 68	124 20 T 54 84	144 4 d 64 100	164 20 t 74 116
5	5 PPC ENQ 5 5	25 PPU NAK 15 21	45 5 % 25 37	65 21 5 35 53	105 5 E 45 69	125 21 U 55 85	145 5 e 65 101	165 21 u 75 117
6	6 ACK 6 6	26 SYN 16 22	46 6 & 26 38	66 22 6 36 54	106 6 F 46 70	126 22 V 56 86	146 6 f 66 102	166 22 v 76 118
7	7 BEL 7 7	27 ETB 17 23	47 7 ’ 27 39	67 23 7 37 55	107 7 G 47 71	127 23 W 57 87	147 7 g 67 103	167 23 w 77 119
8	10 GET BS 8 8	30 SPE CAN 18 24	50 8 (28 40	70 24 8 38 56	110 8 H 48 72	130 24 X 58 88	150 8 h 68 104	170 24 x 78 120
9	11 TCT HT 9 9	31 SPD EM 19 25	51 9) 29 41	71 25 9 39 57	111 9 I 49 73	131 25 Y 59 89	151 9 i 69 105	171 25 y 79 121
A	12 LF A 10	32 SUB 1A 26	52 10 * 2A 42	72 26 : 3A 58	112 10 J 4A 74	132 26 Z 5A 90	152 10 j 6A 106	172 26 z 7A 122
B	13 VT B 11	33 ESC 1B 27	53 11 + 2B 43	73 27 ; 3B 59	113 11 K 4B 75	133 27 [5B 91	153 11 k 6B 107	173 27 { 7B 123
C	14 FF C 12	34 FS 1C 28	54 12 , 2C 44	74 28 < 3C 60	114 12 L 4C 76	134 28 \ 5C 92	154 12 l 6C 108	174 28 | 7C 124
D	15 CR D 13	35 GS 1D 29	55 13 - 2D 45	75 29 = 3D 61	115 13 M 4D 77	135 29] 5D 93	155 13 m 6D 109	175 29 } 7D 125
E	16 SO E 14	36 RS 1E 30	56 14 . 2E 46	76 30 > 3E 62	116 14 N 4E 78	136 30 ^ 5E 94	156 14 n 6E 110	176 30 ~ 7E 126
F	17 SI F 15	37 US 1F 31	57 15 / 2F 47	77 UNL ? 3F 63	117 15 O 4F 79	137 UNT _ 5F 95	157 15 o 6F 111	177 DEL (RUBOUT) 7F 127
	Address commands	Universal commands	Listener address		Talker address		Secondary commands	

Example



Appendix 2 Error Messages

This section describes the error messages related to communications.

- The messages can be displayed in English or Japanese on the SL1000. However, when the messages are read from a PC or other similar computers, the messages are displayed in English.
- If servicing is required, contact your nearest YOKOGAWA dealer for repairs.
- Only error messages related to communications are listed here. For other error messages, see the *SL1000 High-Speed Data Acquisition Unit User's Manual* (IM 720120-01E) and the *SL1000 Control API User's Manual* (IM 720320-01E.)

• Communication syntax errors	100 to 199	Details given below.
• Communication execution errors	200 to 299	
• Communication query errors	400 to 499	
• System communication error	399	
• Warning	50	
• Other Error	350	

Communication Syntax Errors (100 to 199)

Code	Messages	Corrective Action	Reference Page
102	Syntax error.	Invalid syntax.	Chapters 2, 3
103	Invalid separator.	Use a comma to separate the data.	2-1
104	Data type error.	Write using the correct data form.	2-6 to 2-7
105	GET not allowed.	GET is not supported for responses to interface messages.	–
108	Parameter not allowed.	Check the number of data points.	2-6, Chapter 3
109	Missing parameter.	Enter the required data.	2-6, Chapter 3
111	Header separator error.	Use a space to separate the header and data.	2-1
112	Program mnemonic too long.	Check the mnemonic (alphanumerical character string).	Chapter 3
113	Undefined header.	Check the header.	2-4, Chapter 3
114	Header suffix out of range.	Check the header.	2-4, Chapter 3
120	Numeric data error.	A number is required in the <NRf> form.	2-6
123	Exponent too large.	Use a smaller exponent for <NR3> format.	2-6, Chapter 3
124	Too many digits.	The value must be less than equal to 255 digits.	2-6, Chapter 3
128	Numeric data not allowed.	Enter in a format other than <NRf> format.	2-6, Chapter 3
131	Invalid suffix.	Check the unit of the <Voltage>, <Time>, <Frequency>, and <Current>.	2-6
134	Suffix too long.	Check the unit of the <Voltage>, <Time>, <Frequency>, and <Current>.	2-6
138	Suffix not allowed.	No units are allowed other than <Voltage>, <Time>, <Frequency>, and <Current>.	2-6
141	Invalid character data.	Select character data from the selections available in {...|...|...}.	2-7, Chapter 3
144	Character data too long.	Check the spelling of the character strings in {...|...|...}.	2-7, Chapter 3
148	Character data not allowed.	Write in a data form other than {...|...|...}.	2-5, Chapter 3
150	String data error.	Enclose <String> in double quotation or single quotation marks.	2-7
151	Invalid string data.	<String> is too long or contains characters which cannot be used.	2-7, Chapter 3
158	String data not allowed.	Enter in a data format other than <Character string>.	2-6, Chapter 3
161	Invalid block data.	<Block data> is not allowed.	2-7, Chapter 3
168	Block data not allowed.	<Block data> is not allowed.	2-7, Chapter 3
171	Invalid expression.	Equations cannot be used.	Chapter 3
178	Expression data not allowed.	Equations cannot be used.	Chapter 3
181	Invalid outside macro definition.	The SL1000 does not support the IEEE488.2 macro specifications.	–

Communication Execution Errors (200 to 299)

Code	Messages	Corrective Action	Reference Page
221	Setting conflict.	Check the relevant settings.	Chapter 3
222	Data out of range.	Check the range.	Chapter 3
223	Too much data.	Check the length of the data.	Chapter 3
224	Illegal parameter value.	Check the range.	Chapter 3
241	Hardware missing.	Check the installed options.	–
260	Expression error.	Equations cannot be used.	–
270	Macro error.	The SL1000 does not support the IEEE488.2 macro specifications.	–
272	Macro execution error.	The SL1000 does not support the IEEE488.2 macro specifications.	–
273	Illegal macro label.	The SL1000 does not support the IEEE488.2 macro specifications.	–
275	Macro definition too long.	The SL1000 does not support the IEEE488.2 macro specifications.	–
276	Macro recursion error.	The SL1000 does not support the IEEE488.2 macro specifications.	–
277	Macro redefinition not allowed.	The SL1000 does not support the IEEE488.2 macro specifications.	–
278	Macro header not found.	The SL1000 does not support the IEEE488.2 macro specifications.	–

Communication Query Errors (400 to 499)

Code	Messages	Corrective Action	Reference Page
410	Query INTERRUPTED.	Check transmission/reception order.	2-2
420	Query UNTERMINATED.	Check transmission/reception order.	2-2
430	Query DEADLOCKED.	Limit the length of the program message including <PMT> to 1024 bytes or less.	2-2
440	Query UNTERMINATED after indefinite response.	Do not specify a query after the *IDN? or *OPT? command.	–

System Communication Error (399)

Code	Messages	Corrective Action	Reference Page
399	Fatal error in the communication driver.	Maintenance service is required.	–

Warning (50)

Code	Messages	Corrective Action	Reference Page
50	*OPC/? exists in message.	Place the *OPC or *OPC? command at the end of the program message.	–

Other Error (350)

Code	Messages	Corrective Action	Reference Page
350	Queue overflow.	Read the error queue.	4-6

Note

Code 350 indicates overflow of error queue. This code is returned as a response to the STATus:ERRor? query; it does not appear on the screen.