

# LR8101

# LR8102

# HIOKI

Communication Command  
Instruction Manual

# DATA LOGGER



EN

1 LR8100 Communication commands manual .....	10
1.1 Introduction .....	10
2 Outline .....	11
2.1 Setup Procedure .....	11
2.2 Whats command? .....	14
2.3 Status Byte and Event Registers .....	24
2.4 Input and Output buffers .....	28
3 Command .....	29
3.1 Arguments .....	29
3.2 Standard .....	30
*CLS .....	30
*ESR? .....	31
*IDN? .....	32
*OPC .....	33
*OPC? .....	34
*OPT? .....	35
*RST .....	36
*STB? .....	37
*TST? .....	38
*WAI .....	39
*ESR0? .....	40
3.3 Execution control .....	41
*ABORT .....	41
*ERRor? .....	42
*HEADer .....	43
*STARt .....	44
*STATUS? .....	45
*STOP .....	46
*NRMFlag? .....	47
*WAITNextsmpI? .....	49
3.4 Configuration (CONFigure) .....	50
*ADDCoMment .....	50
*ADDDate .....	51
*ATSAve .....	52
*AUTOFolder .....	54
*EXTRECSamp .....	55
*FILEName .....	56
*RECTime .....	57
*SAMPle .....	58
*SAMPKind .....	59
*SAVECalc .....	60
*SAVEDeci .....	61
*SAVEFormat .....	62
*SAVEKind .....	63
*SAVELen .....	64
*SAVEMode .....	65
*SAVEPri .....	66
*SAVEReg .....	67
*SAVESep .....	68
*SAVETime .....	69

•SAVEWave .....	70
•START .....	71
•STARTTime .....	72
•STOP .....	73
•STOPTime .....	74
•SYNC:SET .....	75
•SYNC:CHECK? .....	76
•THINData .....	77
•THINOut .....	78
3.5 Channel (MODule) .....	79
•ADJUST? .....	79
•DATARate .....	80
•DFILter? .....	81
•FILTer .....	82
•IDN? .....	83
•INMOde .....	84
•PCOMode .....	86
•PCOStart .....	87
•PCOUnt .....	88
•PFILTer .....	89
•PINMOde .....	90
•PRANGe .....	91
•PRESet .....	92
•PSLOPe .....	93
•PSMooth .....	94
•PTHRe .....	95
•RANGe .....	96
•RJC .....	97
•SENSor .....	98
•STORE .....	99
•WIRE .....	100
3.6 Scaling (SCALing) .....	101
•KIND .....	101
•OFFSet .....	102
•SCUPLOW .....	103
•SENSE .....	104
•SET .....	105
•UNIT .....	106
•VOLT .....	107
•VOUPLOW .....	108
3.7 Comments (COMMent) .....	109
•ALMCH .....	109
•CH .....	111
•TITLe .....	112
•MODule .....	113
3.8 Triggering (TRIGger) .....	114
•DETECTDate? .....	114
•DETECTTime? .....	115
•MANUal .....	116
•MODE .....	117

•PRETrig .....	118
•SET .....	119
•SOURce .....	120
•SSOURce .....	121
•TIMER .....	122
•TIMIng .....	123
•TMINTvl .....	124
•ANALog:START:KIND .....	125
•ANALog:STOP:KIND .....	125
•CALCulate:START:KIND .....	125
•CALCulate:STOP:KIND .....	125
•PULSe:START:KIND .....	125
•PULSe:STOP:KIND .....	125
•ANALog:START:LEVEL .....	127
•ANALog:STOP:LEVEL .....	127
•CALCulate:START:LEVEL .....	127
•CALCulate:STOP:LEVEL .....	127
•PULSe:START:LEVEL .....	127
•PULSe:STOP:LEVEL .....	127
•ANALog:START:LOWEr .....	129
•ANALog:STOP:LOWEr .....	129
•CALCulate:START:LOWEr .....	129
•CALCulate:STOP:LOWEr .....	129
•PULSe:START:LOWEr .....	129
•PULSe:STOP:LOWEr .....	129
•ANALog:START:SIDE .....	131
•ANALog:STOP:SIDE .....	131
•CALCulate:START:SIDE .....	131
•CALCulate:STOP:SIDE .....	131
•PULSe:START:SIDE .....	131
•PULSe:STOP:SIDE .....	131
•ANALog:START:SLOPe .....	133
•ANALog:STOP:SLOPe .....	133
•CALCulate:START:SLOPe .....	133
•CALCulate:STOP:SLOPe .....	133
•PULSe:START:SLOPe .....	133
•PULSe:STOP:SLOPe .....	133
•ANALog:START:UPPEr .....	135
•ANALog:STOP:UPPEr .....	135
•CALCulate:START:UPPEr .....	135
•CALCulate:STOP:UPPEr .....	135
•PULSe:START:UPPEr .....	135
•PULSe:STOP:UPPEr .....	135
•LOGic:START:PATTErn .....	137
•LOGic:STOP:PATTErn .....	137
•EXTErnal:START:KIND .....	138
•EXTErnal:STOP:KIND .....	138
3.9 Alarm (ALARM) .....	139
•ACTive .....	139
•ARCD? .....	140

•ARCDNum? .....	141
•BEEP .....	142
•BURN .....	143
•FILTer .....	144
•HISTory .....	145
•HOLD .....	146
•SOURce .....	147
•ANALog:KIND .....	148
•CALCulate:KIND .....	148
•PULSe:KIND .....	148
•ANALog:LEVEL .....	150
•CALCulate:LEVEL .....	150
•PULSe:LEVEL .....	150
•ANALog:LOWEr .....	152
•CALCulate:LOWEr .....	152
•PULSe:LOWEr .....	152
•ANALog:SIDE .....	154
•CALCulate:SIDE .....	154
•PULSe:SIDE .....	154
•ANALog:SLOPe .....	156
•CALCulate:SLOPe .....	156
•PULSe:SLOPe .....	156
•ANALog:SLP2:TIME .....	158
•CALCulate:SLP2:TIME .....	158
•PULSe:SLP2:TIME .....	158
•ANALog:STIME .....	160
•CALCulate:STIME .....	160
•PULSe:STIME .....	160
•ANALog:UPPEr .....	162
•CALCulate:UPPEr .....	162
•PULSe:UPPEr .....	162
•LOGic:PATTern .....	164
3.10 Environment (SYSTEM) .....	165
•ADDComment .....	165
•ADDDate .....	166
•ADJDate? .....	167
•BEEP .....	168
•CALCSplit .....	169
•CHECK .....	170
•CHECK:ROMRam .....	170
•CHECK:MODule .....	171
•CHECK:IF:LAN1 .....	172
•CHECK:IF:LAN2 .....	173
•CHECK:MEDia:SD .....	174
•CHECK:MEDia:USB .....	174
•CLBDate? .....	175
•CLOCK:OUT .....	176
•COMMunicate:LAN:DHCP .....	177
•COMMunicate:LAN2:DHCP .....	177
•COMMunicate:LAN:HOSTname .....	179

•COMMunicate:LAN2:HOSTname .....	179
•COMMunicate:LAN:IPADdress .....	180
•COMMunicate:LAN2:IPADdress .....	180
•COMMunicate:LAN:SMASk .....	182
•COMMunicate:LAN2:SMASk .....	182
•COMMunicate:LAN:GATeway .....	184
•COMMunicate:LAN2:GATeway .....	184
•COMMunicate:LAN:CONTRol .....	186
•COMMunicate:LAN2:CONTRol .....	186
•COMMunicate:LAN:DNS .....	188
•COMMunicate:LAN2:DNS .....	188
•COMMunicate:LAN:UPDate .....	190
•COMMunicate:LAN2:UPDate .....	190
•COMMunicate:LAN2:SEND:IPADdress .....	191
•COMMunicate:LAN2:SEND:PORT .....	192
•COMMunicate:LAN2:SEND:ENDian .....	193
•COMMunicate:LAN2:SEND:FORMat .....	194
•COMMunicate:LAN2:SEND:SYNC .....	195
•DATAClear .....	196
•DATE .....	197
•DATETime .....	198
•DFORmat .....	199
•DSEParator .....	200
•FILEName .....	201
•LANGuage .....	202
•MARK .....	203
•SAVEPri .....	204
•STARt .....	205
•THINData .....	206
•THINOut .....	207
•TIME .....	208
•TIMEZone .....	209
•TMAXis .....	210
•EXT:IO1:KIND .....	211
•EXT:IO1:SLOPe:STARt .....	212
•EXT:IO1:SLOPe:STOP .....	213
•EXT:IO2:KIND .....	214
•EXT:IO2:SLOPe:STARt .....	215
•EXT:IO2:SLOPe:STOP .....	216
•EXT:IO3:KIND .....	217
•EXT:IO3:SLOPe:STARt .....	218
•EXT:IO3:SLOPe:STOP .....	219
•EXT:IO4:KIND .....	220
•EXTFILTer .....	221
•EXTSLOPe .....	222
•NTP:ADDRess .....	223
•NTP:CHECK? .....	224
•NTP:KIND .....	225
•NTP:SYNC .....	226
•NTP:STARt .....	227

•FTP:ADDRESS .....	228
•FTP:AUTODEL .....	229
•FTP:CERTIFICATE .....	230
•FTP:CHECK? .....	231
•FTP:DIR .....	232
•FTP:FILE:HOST .....	233
•FTP:FILE:IP .....	234
•FTP:FILE:TIME .....	235
•FTP:PASSWORD .....	236
•FTP:PASV .....	237
•FTP:PORT .....	238
•FTP:PROGRESS? .....	239
•FTP:SECURITY .....	240
•FTP:STATE? .....	241
•FTP:USE .....	242
•FTP:USER .....	243
•RTOU .....	244
3.11 Memory (MEMORY) .....	245
•ADATA? .....	245
•AFETCH? .....	247
•AMAXPOINT? .....	248
•APOINT .....	249
•AREAL? .....	250
•BDATA? .....	251
•BFETCH? .....	253
•BREAL? .....	254
•CHSTORE? .....	255
•FCHSTORE? .....	256
•GETREAL .....	257
•MAXPOINT? .....	258
•POINT .....	259
•TAFETCH? .....	260
•TARCH? .....	261
•TAREAL? .....	262
•TCHSTORE? .....	263
•TFCHSTORE? .....	264
•TOPPOINT? .....	265
•TVFETCH? .....	266
•TVRCH? .....	267
•TVREAL? .....	268
•VDATA? .....	269
•VFETCH? .....	271
•VREAL? .....	272
3.12 Screen (DISPLAY) .....	273
•MARK .....	273
•MARK? .....	274
•MARKJUMP? .....	275
3.13 Calculations (CALCULATE) .....	276
•MEASURE .....	276
•MEAS:ANSWER? .....	277

!MEAS:FILE .....	278
!MEAS:INTegra .....	279
!MEAS:KIND .....	280
!MEAS:LEN .....	281
!MEAS:LEVEL .....	282
!MEAS:REG .....	283
!MEAS:SET .....	284
!MEAS:TARGet .....	286
!MEAS:TIME .....	287
!WAVE:KIND .....	288
!WAVE:STR .....	289
!WAVE:ARITHmetic:COEF:A .....	290
!WAVE:ARITHmetic:COEF:B .....	290
!WAVE:ARITHmetic:COEF:C .....	290
!WAVE:ARITHmetic:COEF:D .....	290
!WAVE:ARITHmetic:COEF:E .....	290
!WAVE:ARITHmetic:OPERator:A .....	292
!WAVE:ARITHmetic:OPERator:B .....	292
!WAVE:ARITHmetic:OPERator:C .....	292
!WAVE:ARITHmetic:OPERator:D .....	292
!WAVE:MOVE:POINT .....	294
!WAVE:RESet:BASE .....	295
!WAVE:RESet:INT .....	296
!WAVE:RESet:KIND .....	297
!WAVE:RESet:TIME .....	298
!WAVE:SOURce:SR1 .....	299
!WAVE:SOURce:SR2 .....	299
!WAVE:SOURce:SR3 .....	299
!WAVE:SOURce:SR4 .....	299
3.14 Error (ERRor) .....	301
!BIT:ERRor? .....	301
!BIT:WARNing? .....	302
!BIT:WARNing:CLEAr .....	303
!LOG:ERRor? .....	304
!LOG:ERRor:CLEAr .....	305
!LOG:WARNing? .....	306
!LOG:WARNing:CLEAr .....	307
3.15 Media (MEDia) .....	308
!SD:FINFo:SET? .....	308
!USB:FINFo:SET? .....	308
!SD:FREE? .....	309
!USB:FREE? .....	309
!SD:FORMat? .....	310
!USB:FORMat? .....	310
!SD:FLISt:SET? .....	311
!USB:FLISt:SET? .....	311
!SD:LOAD:SET .....	312
!USB:LOAD:SET .....	312
!SD:SAVE:DATA:MEM .....	314
!USB:SAVE:DATA:MEM .....	314

•SD:SAVE:DATA:CSV .....	314
•USB:SAVE:DATA:CSV .....	314
•SD:SAVE:DATA:MF4 .....	314
•USB:SAVE:DATA:MF4 .....	314
•SD:SAVE:SET .....	314
•USB:SAVE:SET .....	314
•SD:SAVE:A2L:LAN1 .....	314
•USB:SAVE:A2L:LAN1 .....	314
•SD:SAVE:A2L:LAN2 .....	314
•USB:SAVE:A2L:LAN2 .....	314
•SD:SAVE:CALC:CSV .....	314
•USB:SAVE:CALC:CSV .....	314
4 Troubleshooting .....	317
4.1 Wired LAN .....	317

# 1

# LR8100 Communication commands manual

## 1.1 Introduction

- This manual explains the communication commands for Model LR8100 Memory HiLoggers.
- Please refer to the instruction manual for Model LR8100 for details regarding command settings.
- **In the measurement using the communication commands described in this instruction manual, data can be acquired in real time at intervals of 50 ms at the shortest when the number of channels synchronized for sampling is 750 channels (up to 75 channels per unit).** If you wish to acquire data in real time at intervals faster than 50 milliseconds, use the Logger Utility, a dedicated software application.
- Although all reasonable care has been taken in the production of this manual, should you find any points which are unclear or in error, please contact your local distributor or the HIOKI International Sales & Marketing Division.
- In the interest of product development, the contents of this manual may be subject to revision without notice.
- Unauthorized reproduction or copying of this manual is prohibited.
- Microsoft, Windows are registered trademarks or trademarks of Microsoft Corporation in the U.S.A. and other countries.
- All other names are registered trademarks or trademarks of their respective companies.

### Contact information

---

---

Please contact us through the "Contact us" link on our website.

<https://www.hioki.com/us-en/contact>

### Revision History

---

---

- First Edition, December 2023

# 2

## Outline

### 2.1 Setup Protocols

#### Wired LAN Specifications

---

---

Model LR8100 can be connected to a PC using a LAN cable.

The port number set in "2.8 Configuring and Establishing a LAN Connection" in the Instruction Manual is used.

**If another PC is connected during connection, disconnect the PC that was previously connected, and connect to the PC that was connected later.**

---

Data link	IEEE802.3 Ethernet 1000BASE-T、100BASE-TX
-----------	--

---

Transfer speed	1Gbps
----------------	-------

---

Protocol	TCP/IP
----------	--------

## Wired LAN Interface Setting

---

---

To connect to an existing network, settings may need to be allocated by the network system administrator (IS department) beforehand.  
This is to avoid duplicating certain settings already allocated to other devices.  
Obtain the following setting information from the administrator (IS department), and write it down.

---

Host name (up to 15 characters)	XXXXXXXXXXXXXXXXXX
------------------------------------	--------------------

---

IPaddress	XXX.XXX.XXX.XXX
-----------	-----------------

---

Subnet mask	XXX.XXX.XXX.XXX
-------------	-----------------

---

Default gateway (present or not present)	XXX.XXX.XXX.XXX (when present)
---	-----------------------------------

---

TCP/IP port number	XXXX The default number is 880X. communication commands use the end of the port number is 2, so specify 8802 by default.
--------------------	---

---

If assembling a stand-alone network with no connection to an existing network, you can still obtain the address information from the administrator, or if there is no administrator, you can assign the setup values yourself. An IP address within the following range is recommended when setting up such a stand-alone network:

**192.168.0.1 to 192.168.253.254**

(Note that the rightmost number may not be 0 or 255.)

We recommend setting the controller device (typically a PC) to 192.168.1.1, and setting this device to 192.168.1.2.

If several Memory HiLoggers are to be connected, the last number should be incremented by one for each device (e.g., the second HiLogger would be set to 192.168.1.3).

The host name must be unique for each device (no two devices may have the same name).

Otherwise, the following settings should be satisfactory:

[Example]

Setting	LR8100	PC
Host name	LOGGER	PC
IP name	192.168.1.2	192.168.1.1
Subnet mask	255.255.255.0	255.255.255.0
Default gateway	OFF	OFF
TCP/IP Port number	880X	※1

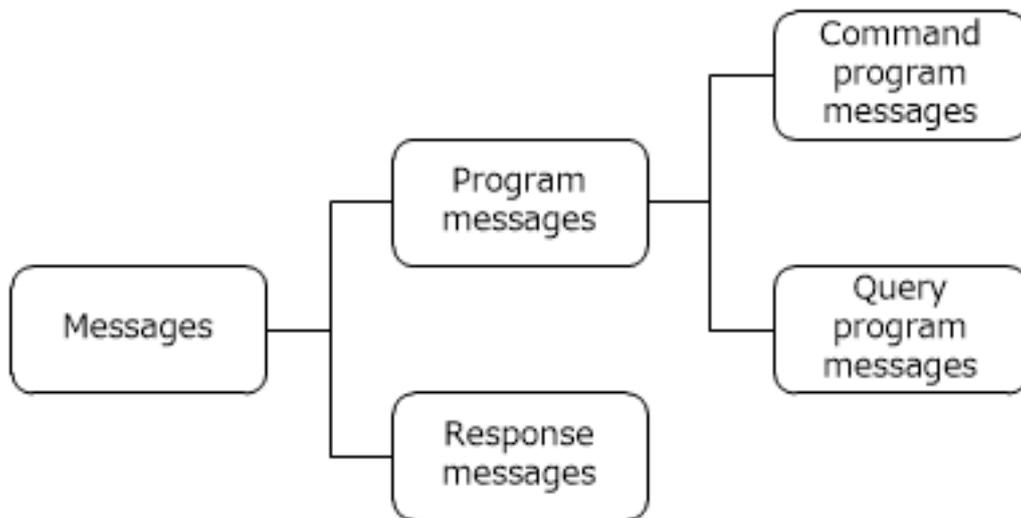
※1: If this unit is set to [880X], set the connection destination port of the computer to [8802].

## 2.2 Whats Command?

### Command Messages

---

Commands, queries and responses sent and received through the interface are called messages. Message types are as follows:



---

Program messages

This term refers to both command messages and query messages.

---

Response messages

These are returned from the instrument to the PC in response to query messages.

The instrument produces a response message as soon as it receives a query message and checks its syntax.

---

Command program messages

These are instrument control instructions sent from the PC, such as to make or reset settings.

---

Query program messages

These are sent from the PC to the instrument to interrogate the state of a particular setting or the result of an operation.

The instrument produces a response message as soon as it receives a query message and checks its syntax.

---

In the following descriptions the word "command" refers to both command and query program messages, except where an ambiguity may result.

(Control commands: HIOKI unique SCPI)

## Command Syntax

---

Commands sent to the instrument are not case sensitive.

Command names are intended to be mnemonic, and all have **a long form** and **an abbreviated short form**.

In the command descriptions in this manual, the short form appears in upper-case letters, and is continued in lower-case letters to show the long form as well.

While either form is valid during operation, intermediate forms are invalid. Both lower- and upper-case letters are accepted without distinction during operation.

**The instrument generates response messages in the long form (when headers are enabled), in all upper-case letters.**

(Example)

Command descriptions in this manual	Short form	Long form
DISPlay	DISP	DISPLAY

For "DISPlay" commands, either "DISPLAY" (the long form) or "DISP" (the short form) is accepted.

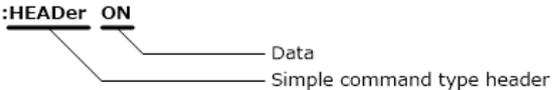
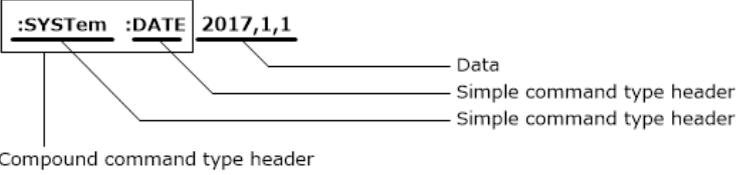
However, all of "DISPLA", "DISPL" and "DIS" are invalid and will generate an error.

## Command headers

---

All commands must include a header for identification.

There are three kinds of headers (command types): **simple**, **compound**, and **standard**.

Command Type (Header)	Description
Simple	<p>The first word constitutes the header.</p> <p>(Example)</p> <p><b>:HEADer ON</b></p> 
Compound	<p>A header composed of multiple simple command type headers separated by colons.</p> <p>(Example)</p> <p><b>:SYSTem :DATE 2017,1,1</b></p> 
Standard	<p>A command beginning with an asterisk and stipulated by IEEE 488.2.</p> <p>(Example)</p> <p style="text-align: center;"><b>*RST</b></p>

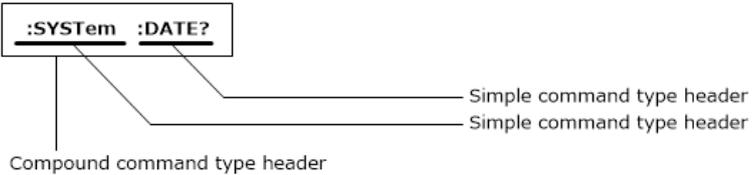
## Query headers

---

Queries interrogate the instrument regarding a setting state or the result of an operation.

Query headers are distinguished by a question mark at the end of the header.

The structure of a query header is identical to that of its corresponding command header, with "?" always appended as the last character.

Command Type	Description
Simple	<p>The first word constitutes the header.</p> <p>(Example)</p> <p style="text-align: center;"><b><u>:HEADer?</u></b></p> 
Compound	<p>A header composed of multiple simple command type headers separated by colons.</p> <p>(Example)</p> <div style="border: 1px solid black; padding: 5px; display: inline-block; margin-bottom: 10px;"> <b><u>:SYSTem</u> <u>:DATE?</u></b> </div> 
Standard	<p>A command beginning with an asterisk and stipulated by IEEE 488.2.</p> <p>(Example)</p> <p style="text-align: center;"><b>*IDN?</b></p>

## Response Messages

---

---

Response messages returned by the instrument to the PC in response to received queries are composed of a header portion (omitted when headers are disabled) and a data portion identical to the data portion defined for the command message that corresponds to the received query. As a general rule, the format of returned data is the same as that of the command message.

(Example)

---

Query	:SYSTem:TIME?
Response	:SYSTEM:TIME 12,34,56 (when headers are enabled) 12,34,56 (when headers are disabled)

---

# Terminators and Separators

---

---

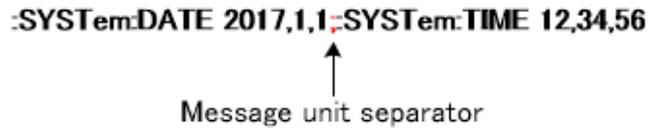
## (1) Message terminators

The terminator character signifies the end of a message (including compound messages to be sent as one line), and is not considered part of the message itself.

Setting on the instrument and PC	ANSI code (hexadecimal)	Meaning	Description	Delimiter setting
LF	0Ah	New Line	Line Feed	LF is the message delimiter
CR+LF	0Dh 0Ah			CR+LF is the message delimiter

## (2) Message separator

When a compound message is to be sent to the instrument as one line, a semicolon ";" serves as the separator for individual messages.



## (3) Header separator

Messages having both a header and data include a space character (" ") to separate the header from the data.

The space is not part of the actual command.



## (4) Data separator

Messages containing multiple data items include commas to separate the data items from one another.



## The Command Tree

---

When writing multiple messages in compound command form on the same line, as a rule, the header from one message is considered to carry forward to immediately following messages that do not begin with a colon (following the semicolon message separator).

This construction corresponds to the general concept of the current directory in the directory structure of UNIX or MS-DOS,

and the header that is carried forward is called the "**current path**".

【Example 1】

```
:SYSTem:DATE 2017,1,1;:SYSTem:TIME 12,34,56
```

【Example 2】

```
:SYSTem:DATE 2017,1,1;TIME 12,34,56
```

【Example 3】

```
:SYSTem:DATE 2017,1,1
```

(After sending the above data, send the following data)

```
TIME 12,34,56
```

Examples 1, 2 and 3 are commands to set time. In Example 1, because there is a colon directly after the semicolon, the current position is the "**root**".

The subsequent command is therefore referenced relative to the root.

On the other hand, Examples 2 and 3 take advantage of the fact that the first command

(":SYSTem:DATE 2017,1,1;") sets the **current path** to ":SYSTEM",

so that the ":SYSTem:" before "TIME" in the second command is omitted. To reiterate, the colon at the beginning of a command forces command parsing to begin from the **root**.

## Data formats

---

The instrument recognizes the following data formats:  
character data, decimal numeric data and character string data.

### [Character data]

1. The first character must be alphabetic.
2. Subsequent characters may only be alphabetic characters, numerals or underline characters ( ).
3. The instrument accepts both upper- and lower-case character data, but returns only upper-case characters.

【Example】

**:HEADER ON**

### [Decimal data]

1. Decimal data values are represented in what is termed NR formats.
2. The three types of **NRf** formats are called **NR1**, **NR2** and **NR3**, each of which may be signed or unsigned numbers.

NRf format	Data Type	Example
NR1	Integer data	+15,-20,25
NR2	Fixed-point numbers	+1.23,-4.57,7.89
NR3	Floating-point numbers	+10.0E-3,-2.3E+3,5E+3

3. Unsigned numbers are considered positive.
4. Numerical values with accuracy exceeding the range with which the instrument can deal are rounded (digits 5 and above are rounded up, and 4 and below are rounded down).
5. The term "NRf format" includes all three formats. Though the instrument can accept any NRf format, it only returns data in the format (NR1 to NR3) defined for the corresponding command.

(Example 1) NR1 format

**:SYSTem:THINOut 10**

(Example 2) NR2 format

**:CONFigure:SAMPlE 0.1**

(Example 3) NR3 format

**:ALARm:ANALog:LEVEl ALM1,CH1\_1,+1.0E-3**

### [Character string data]

1. Character string data is enclosed within quotation marks.
2. The data consists of 8-bit ASCII characters.
3. Characters which cannot be handled by the instrument are replaced by underscores.
4. Although the instrument sends only double quotation marks (") to the PC, it accepts both double and single quotation marks (').

【Example】

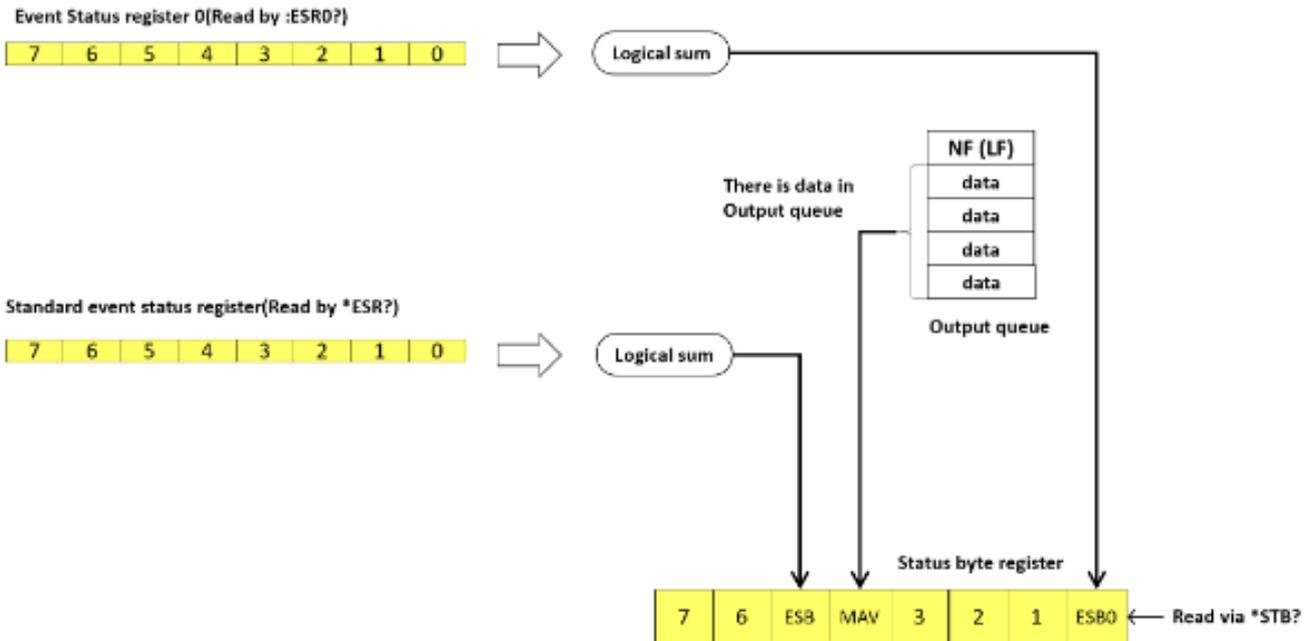
:COMMeNt:TITLe 'HIOKI'

:COMMeNt:TITLe "HIOKI"

## 2.3 Registers

### Status Byte and Event Registers

The instrument includes a number of standard registers:  
the "Standard Event Status Register", "Status Byte Register", "Event Status Register 0".



## The status byte

---

Each bit of the status byte is a summary (logical OR) of the event register corresponding to that bit.

Status byte	Contents
bit7	Unused:0
bit6	Unused:0
bit5 (ESB)	Event summary bit Shows a summary of the Standard Event Status Register.
bit4 (MAV)	Message available Shows that a message is present in the output queue.
bit3	Unused:0
bit2	Unused:0
bit1	Unused:0
bit0 (ESB0)	Event summary bit 0 Shows a summary of Event Status Register 0.

The following command read the status byte.

**\*STB?**

## Standard Event Status Register (SESR)

Bit 5 of the status byte indicates the summary of this register.

The following events clear the contents of the Standard Event Status Register:

1. When the \*CLS command is received.
2. When the contents have been read by an \*ESR? query.
3. When power is turned off and on again.

Standard Event Status Register (SESR)	Contents
bit7 (PON)	Power has been turned on again. Since this register was last read, the instrument has been powered off and on. Initialized to one at power on.
bit6 (URQ)	User request Unused:0
bit5 (CME)	Command error. There is an error in a command that has been received: either an error in syntax, or an error in meaning.
bit4 (EXE)	Execution error, Range or Mode error. An error has occurred while executing a command.
bit3 (DDE)	Device-dependent error. Unused:0
bit2 (QYE)	Query error. The queue is empty, or data loss has occurred (queue overflow).
bit1 (RQC)	Request for controller rights. Unused: 0
bit0 (OPC)	Operation finished. *Only set for the *OPC command.

The following commands are used to read the Standard Event Status Register.

**\*ESR?**

## Event Status Register 0 (ESR0)

---

The summary of this register is set in bit 0 of the status byte.

The following events clear the contents of Event Status Register 0:

1. When the \*CLS command is received.
2. When the contents have been read by an :ESR0? query.
3. When power is turned off and on again.

Event Status Register 0 (ESR0)	Contents
bit7	Unused:0
bit6	Unused:0
bit5	Unused:0
bit4	Unused:0
bit3	Unused:0
bit2	Trigger wait finished (set when a trigger event occurs).
bit1	Measurement operation concluded (set by STOP).
bit0	Occurrence of error/warning.

The following commands read the Event Status Register 0.

**:ESR0?**

## 2.4 The input buffer and the output queue

### Input buffer

---

---

The instrument has a 200KB input buffer.  
Received messages are put into this buffer and executed in order.  
However, "**ABORT**" commands are executed immediately when received.

### Output queue

---

---

The instrument has a 200KB output queue.  
Response messages accumulate in this queue until read out by the controller (PC).  
A query error occurs if the length of a response message exceeds 200KB.  
The following events clear the output queue:

1. When the controller has read out the entire contents.
2. When power is turned off and on again.
3. Upon receipt of the next message.

# 3

# Command

## 3.1 Argument

Notation	Meaning	Example
A\$	Character string data	OFF,ON
module\$	Module string data	MODULE1 to MODULE10
ch\$	Channel string data	CH1_1 to CH10_30/PLS1/ALM1 to ALM4/W1 to W30
pls\$	Pulse Channel string data	PLS1
alm\$	Alarm Channel string data	ALM1 to ALM4
w\$	Wave Calculate Channel string data	W1 to W30
A,B,C	Numerical value	10, -20, 1.5E+05, 0.1
A<NR1>	Integer data	+15, -20, 25
A<NR2>	Fixed point numbers	+1.23, -4.57, 7.89
A<NR3>	Floating point numbers	+10.0E-03, -2.3E+03, 5E+03

## 3.2 Standard

Clear the status byte and associated queues

\*CLS

### Syntax

---

#### Command

\*CLS

### Details

---

This command clears the event register associated with each bit of the status byte register. It also clears the status byte register.

### Example

---

\*CLS

### Note

---

Because it does not clear the output queue, it has no effect on bit 4 (MAV) of the status byte.

### Usage Conditions

---

## Query and clear SESR

\*ESR?

### Syntax

---

#### Query

\*ESR?

#### Response

A<NR1>

A = 0 to 255

### Details

---

Reads and clears the contents of the standard event status register (SESR).  
The contents of the SESR are returned.

### Example

---

\*ESR?

(Response) \*ESR 0 (when headers are enabled)

### Note

---

### Usage Conditions

---

## Query device ID

\*IDN?

### Syntax

---

#### Query

\*IDN?

#### Response

A\$,B\$,C\$,D\$

A\$ = Manufacturer's name

B\$ = Model name

C\$ = Serial number

D\$ = Software version

### Details

---

Query device ID.

### Example

---

\*IDN?

(Response) \*IDN HIOKI,LR8102,221215007,V1.00 (when headers are enabled)

### Note

---

### Usage Conditions

---

## Set the LSB of SESR when finished processing

**\*OPC**

### Syntax

---

#### Command

\*OPC

### Details

---

OPC Sets the LSB (bit 0) of SESR (the standard event status register) after all commands finish executing.

### Example

---

A\$;\*OPC

(After the execution of the commands A\$ is completed, the LSB of SESR is set.)

### Note

---

The command to wait is as follows:

- STOP waveform sampling(:STOP)
  - Captures real time data(:MEMory:GETReal)
  - Initialize instrument settings(\*RST)
- (If you wait for the measurement stop, you need to send the STOP command twice.)

### Usage Conditions

---

Return an ASCII 1 when finished processing

**\*OPC?**

## Syntax

---

### Query

\*OPC?

### Response

A<NR1>

A = 1

## Details

---

When the command preceding the \*OPC command completes execution, the response of ASCII [1] is made.

## Example

---

A\$;\*OPC?

(After the execution of the commands A\$ is completed, Returns ASCII [1]) (Response) \*OPC 1 (when headers are enabled)

## Note

---

The command to wait is as follows:

- STOP waveform sampling(:STOP)
- Captures real time data(:MEMory:GETReal)
- Initialize instrument settings(\*RST)

(If you wait for the measurement stop, you need to send the STOP command twice.)

## Usage Conditions

---

Queries device option provision.

**\*OPT?**

## Syntax

---

### Query

\*OPT?

### Response

A1,A2,A3,...,A10<NR1>

Ax = 0 to 3

## Details

---

Returns device option provision.

0: No Module

1: M7100 15ch Voltage and temperature module

3: M7102 30ch Voltage and temperature module

## Example

---

\*OPT?

(Response) \*OPT 1,1,3,1,1,3,1,1,3,1 (when headers are enabled)

## Note

---

## Usage Conditions

---

Device initial setting.

**\*RST**

## Syntax

---

### Command

\*RST

## Details

---

Initializes the system.

## Example

---

\*RST

## Note

---

Interface settings are not initialized.  
Initialization takes a little time.

## Usage Conditions

---

Reads the status byte and MSS bit.

**\*STB?**

## Syntax

---

### Query

\*STB?

### Response

A<NR1>

A = 0 to 255

## Details

---

This is the same as reading out the status byte.

## Example

---

\*STB?

(Response) \*STB? 128 (when headers are enabled)

## Note

---

## Usage Conditions

---

Queries the result of the ROM/RAM check.

**\*TST?**

## Syntax

---

### Query

\*TST?

### Response

A<NR1>

A = 0, 1

## Details

---

The result of the ROM/RAM check of the module is returned as an NR1 numerical value.

0: Pass

1: Fail

## Example

---

\*TST?

(Response) \*TST 0 (when headers are enabled)

## Note

---

## Usage Conditions

---

Execute the following command when finished processing

**\*WAI**

## Syntax

---

### Command

\*WAI

## Details

---

After the execution of the command is completed, subsequently performs the following command.

## Example

---

A\$;\*WAI;\*IDN?

## Note

---

The command to wait is as follows:

- STOP waveform sampling(:STOP)
- Captures real time data(:MEMory:GETReal)
- Initialize instrument settings(\*RST)  
(If you wait for the measurement stop, you need to send the STOP command twice.)

## Usage Conditions

---

Reads and clear event status register 0 (ESR0).

:ESR0?

## Syntax

---

### Query

:ESR0?

### Response

A<NR1>

A = 0 to 255

## Details

---

The contents of ESR0 are returned as an NR1 numerical value, and ESR0 is cleared.

## Example

---

:ESR0?

(Response) :ESR0 0 (when headers are enabled)

## Note

---

## Usage Conditions

---

## 3.3 Execution control

Force STOP waveform sampling.

:ABORT

### Syntax

---

#### Command

:ABORT

### Details

---

Force STOP waveform sampling.

### Example

---

:ABORT

### Note

---

Terminates even if waveform sampling operation is not yet completed.

:ABORT command cannot be used in combination with other commands.(e.g.:ABORT;\*OPC?)

### Usage Conditions

---

Queries the module error number

:ERRor?

## Syntax

---

### Query

:ERRor?

### Response

A\$

A\$ = error number

## Details

---

The number of error or warning that has occurred on the module is returned.  
For errors or warnings, refer to the instruction manual included with the module.

## Example

---

:ERRor?

(Response) :ERROR ERR\_SY01 (when headers are enabled)

## Note

---

## Usage Conditions

---

Enable/disable header prefixes on query responses. Query the current setting of header prefixes

**:HEADer A\$**

## Syntax

---

### Command

:HEADer A\$

### Query

:HEADer?

### Response

A\$

A\$ = OFF, ON

## Details

---

Sets header enablement.

When headers are enabled, responses to queries are prefixed by headers; when headers are disabled, responses are not so prefixed.

Returns whether or not headers are prefixed to responses to queries.

OFF: headers are disabled.

ON: headers are enabled.

## Example

---

:HEADer ON

:HEADer?

(Response) :HEADER ON (when headers are enabled)

## Note

---

## Usage Conditions

---

## START waveform sampling

:STARTt

### Syntax

---

#### Command

:STARTt

### Details

---

START waveform sampling.

### Example

---

:STARTt

### Note

---

### Usage Conditions

---

Measurement cannot be started when synchronous operation is set to Secondary.

## Query storage status

:STATUS?

### Syntax

---

#### Query

:STATUS?

#### Response

A<NR1>

A = 0 to 63

### Details

---

Returns the current storage status of the instrument.

For example, the value 3 is returned when starting (bit 0) and storing (bit 1).

Bit0: starting

Bit1: storing

Bit2: awaiting trigger

Bit3: pre-trigger wait period

Bit4: (unused)

Bit5: saving

### Example

---

:STATUS?

(Response) :STATUS 3 (when headers are enabled)

### Note

---

### Usage Conditions

---

## STOP waveform sampling

**:STOP**

### Syntax

---

#### Command

**:STOP**

### Details

---

STOP waveform sampling.

Terminates at the instant that waveform sampling operation is completed.

### Example

---

**:STOP**

### Note

---

The operation differs depending on the recording time setting.

First STOP command

When recording time is Continuous → Not stop

When recording time is Designated time → Stops after measuring for the recording time.

Second STOP command

When recording time is Continuous → Stop waveform sampling.

When Recording time is Designated time → Stop waveform sampling.

### Usage Conditions

---

Measurement cannot be stopped when synchronous operation is set to Secondary.

## Normalization control

:NRMFlag?

### Syntax

#### Query

:NRMFlag?

#### Response

A<NR1>

A = 0x00000000 to 0xffffffff

### Details

Reads out the normalized bits.

Returns the value in hexadecimal to NR1 and clears the bit.

Bit	Object	Content
1	Others	Normalization other than the following classifications.
2	Recording Interval	Normalization of recording intervals.
3	External input terminal (I/O 3)	When the external trigger setting is ON, the setting of the external input terminal (I/O 3) is changed to trigger input.
4	Recording time	Change in recording time based on the number of channels used and recording interval.
5	Waveform data split time for automatic saving	Change of waveform data division time by recording interval.
6	Numerical calculation results split time of automatic saving	Change of numerical calculation result division time by recording interval.
7	Pretrigger time	Change of pre-trigger time by recording interval.
8	Module data update interval	Change of data update interval of the module by recording interval.
9	Waveform data saving format for automatic saving	Change of waveform data saving format by the number of channels used and recording interval.
10	Repeat recording	When the interval trigger setting is turned ON, the repeat recording setting is changed to ON.
11	Interval trigger	When the repeat recording setting is turned OFF, the interval trigger setting is changed to OFF.
12	Auto save text format date format	Change text format date format setting to localization format
13	Measurement start and stop times	Change measurement start and stop times.

### Example

:NRMFlag?

(Response) :NRMFLAG 4 (when headers are enabled)

**Note**

---

**Usage Conditions**

---

Queries for held data number.

`:WAITNextsmpl?`

## Syntax

---

### Query

`:WAITNextsmpl?`

### Response

A<NR1>

A = 0 to Latest storage number

A = -1 (When not in measurement)

## Details

---

Combined with the `:MEMory:xxxxFETch?` series of commands, you can retrieve the data immediately after it has been updated.

The hold data acquisition commands that can be combined are as follows.

`:MEMory:VFETch?`

`:MEMory:AFETch?`

`:MEMory:BFETch?`

`:MEMory:TVFETch?`

`:MEMory:TAFETch?`

## Example

---

`:WAITNextsmpl?`

(Response) `:WAITNEXTSMPL 1000` (when headers are enabled)

`:MEMory:TVFETch? MODULE1`

(Response) `:MEMORY:TVFETCH +1.00000E-2, +2.00000E-2, +3.00000E-2` (when headers are enabled)

## Note

---

If the recording interval is slow (from 10 s), an execution error occurs.

There is no need to send `:MEMory:GETReal` command.

## Usage Conditions

---

## 3.4 Configuration(CONFigure)

Sets and queries the title comment addition for auto save file name.

`:CONFigure:ADDCOMMENT A$`

### Syntax

---

#### Command

`:CONFigure:ADDCOMMENT A$`

#### Query

`:CONFigure:ADDCOMMENT?`

#### Response

A\$

A\$ = OFF, ON

### Details

---

Sets the title comment addition for auto save file name.

Returns the current setting of the title comment addition for auto save file name.

OFF: Do not add title comment(Automatically add numbers)

ON: Add title comment

### Example

---

`:CONFigure:ADDCOMMENT ON`

`:CONFigure:ADDCOMMENT?`

(Response) `:CONFigure:ADDCOMMENT ON` (when headers are enabled)

### Note

---

### Usage Conditions

---

Sets and queries the trigger date addition for auto save file name.

**:CONFigure:ADDDate A\$**

## Syntax

---

### Command

:CONFigure:ADDDate A\$

### Query

:CONFigure:ADDDate?

### Response

A\$

A\$ = OFF, ON

## Details

---

Sets the trigger date addition for auto save file name.

Returns the current setting of the trigger date addition for auto save file name.

OFF: Do not add trigger date (Automatically add numbers)

ON: Add trigger date

## Example

---

:CONFigure:ADDDate ON

:CONFigure:ADDDate?

(Response) :CONFIGURE:ADDDATE ON (when headers are enabled)

## Note

---

## Usage Conditions

---

Sets and queries the auto save function.

**:CONFigure:ATSAve A\$,"B\$")**

## Syntax

---

### Command

:CONFigure:ATSAve A\$,"B\$")

### Query

:CONFigure:ATSAve?

### Response

A\$,"B\$")

A\$ = OFF, BIN, CSV, MF4, MEAS, BIN\_MEAS, CSV\_MEAS, MF4\_MEAS

B\$ = file name (up to 8 characters)(can be omitted)

## Details

---

Sets the auto save function. Omit B\$ only when A\$ = OFF.

Returns the auto save function as character data.

OFF: (Waveform Data) No Save (Numerical Calculation Data) No Save

BIN: (Waveform Data) Save binary format (Numerical Calculation Data) No Save

CSV: (Waveform Data) Save text format (Numerical Calculation Data) No Save

MF4: (Waveform Data) Save MF4 format (Numerical Calculation Data) No Save

MEAS: (Waveform Data) No Save (Numerical Calculation Data) Save text format

BIN\_MEAS: (Waveform Data) Save binary format (Numerical Calculation Data) Save text format

CSV\_MEAS: (Waveform Data) Save text format (Numerical Calculation Data) Save text format

MF4\_MEAS: (Waveform Data) Save MF4 format (Numerical Calculation Data) Save text format

## Example

---

:CONFigure:ATSAve BIN,"AUTO"

:CONFigure:ATSAve?

(Response) :CONFIGURE:ATSAVE BIN,"AUTO" (when headers are enabled)

## Note

---

Waveform data save format can be set by :CONFigure:SAVEWave command.

Numerical calculation data save format can be set by :CONFigure:SAVECalc command.

File name can be set by :CONFigure:FILENAME command.

## Usage Conditions

---

Sets and queries the folder Splitting for auto save.

:CONFigure:AUTOFolder A\$

## Syntax

---

### Command

:CONFigure:AUTOFolder A\$

### Query

:CONFigure:AUTOFolder?

### Response

A\$

A\$ = OFF, DAY, WEEK, MONTH

## Details

---

Sets folder Splitting for auto save.

Returns the current setting of the folder Splitting for auto save.

OFF: No Splitting

DAY: 1 Day

WEEK: 1 Week

MONTH: 1 Month

## Example

---

:CONFigure:AUTOFolder DAY

:CONFigure:AUTOFolder?

(Response) :CONFIGURE:AUTOFOLDER DAY (when headers are enabled)

## Note

---

## Usage Conditions

---

Sets and queries the number of samples recorded for external sampling.

**:CONFigure:EXTRECSamp A**

## Syntax

---

### Command

:CONFigure:EXTRECSamp A

### Query

:CONFigure:EXTRECSamp?

### Response

A<NR1>

A = 1 to 1000000000

## Details

---

Sets the number of samples recorded for external sampling.

Returns the current number of recorded samples for external sampling as an NR1 number.

## Example

---

:CONFigure:EXTRECSamp 10

:CONFigure:EXTRECSamp?

(Response) :CONFIGURE:EXTRECSAMP 10 (when headers are enabled)

## Note

---

## Usage Conditions

---

Sets and queries the file name for auto save.

`:CONFigure:FILEName "A$"`

## Syntax

---

### Command

`:CONFigure:FILEName "A$"`

### Query

`:CONFigure:FILEName?`

### Response

"A\$"

A\$ = file name (up to 8 characters)

## Details

---

Sets file name for auto save.

Returns a string of file name for auto save.

## Example

---

`:CONFigure:FILEName "ABC"`

`:CONFigure:FILEName?`

(Response) `:CONFIGURE:FILENAME "ABC"` (when headers are enabled)

## Note

---

File name can be set by `:CONFigure:ATSAve` command.

Characters exceeding the maximum number of characters are not set.

## Usage Conditions

---

Sets and queries the recording time.

`:CONFigure:RECTime day,hour,min,sec`

## Syntax

---

### Command

`:CONFigure:RECTime day,hour,min,sec`

### Query

`:CONFigure:RECTime?`

### Response

`day<NR1>,hour<NR1>,min<NR1>,sec<NR1>`

day = 0 to 500 (day)

hour = 0 to 23 (hour)

min = 0 to 59 (min)

sec = 0 to 59 (sec)

## Details

---

Sets the recording time to a numerical value.

Returns the currently set value of the recording time as an NR1 numerical value.

If all parameters are 0, the recording time is continuous.

## Example

---

`:CONFigure:RECTime 0,0,0,10`

`:CONFigure:RECTime?`

(Response) `:CONFIGURE:RECTIME 0,0,0,10` (when headers are enabled)

## Note

---

## Usage Conditions

---

Sets and queries the recording interval.

**:CONFigure:SAMPle A**

## Syntax

---

### Command

:CONFigure:SAMPle A

### Query

:CONFigure:SAMPle?

### Response

A<NR3> (1 decimal places)

A = 5.0E-3 to 3.6E+3 (sec)

## Details

---

Sets the recording interval to a numerical value (unit seconds).

Returns the currently set value of the recording interval as an NR3 numerical value.

## Example

---

:CONFigure:SAMPle 1E-2

:CONFigure:SAMPle?

(Response) :CONFIGURE:SAMPLE 1.0E-02 (when headers are enabled)

## Note

---

If an attempt is made to set the horizontal axis to a non-permitted value, and there is a range above that value, that range will be selected.

## Usage Conditions

---

Sets and queries the recording mode.

**:CONFigure:SAMPKind A\$**

## Syntax

---

### Command

:CONFigure:SAMPKind A\$

### Query

:CONFigure:SAMPKind?

### Response

A\$

A\$ = NORMAl, EXT

## Details

---

Sets the recording mode(function).

Returns the currently the recording mode.

NORMAl: Records data in synchronization with the internal clock.

EXT: Records data in synchronization with an external clock.

## Example

---

:CONFigure:SAMPKind NORMAl

:CONFigure:SAMPKind?

(Response) :CONFIGURE:SAMPKIND NORMAL (when headers are enabled)

## Note

---

## Usage Conditions

---

Sets and queries the format for saving mathematical results.

`:CONFigure:SAVECalc A$`

## Syntax

---

### Command

`:CONFigure:SAVECalc A$`

### Query

`:CONFigure:SAVECalc?`

### Response

A\$

A\$ = OFF, CSV

## Details

---

Sets the format for saving mathematical results by the auto save function.

Returns the current setting of the format for saving mathematical results.

OFF: No Save

CSV: save text format

## Example

---

`:CONFigure:SAVECalc CSV`

`:CONFigure:SAVECalc?`

(Response) `:CONFIGURE:SAVECALC CSV` (when headers are enabled)

## Note

---

Numerical calculation data save format can be set by `:CONFigure:ATSAve` command.

## Usage Conditions

---

Sets and queries the text save decimal.

**:CONFigure:SAVEDeci A\$**

## Syntax

---

### Command

:CONFigure:SAVEDeci A\$

### Query

:CONFigure:SAVEDeci?

### Response

A\$

A\$ = PERIOD, COMMA

## Details

---

Sets the text save decimal when data is stored in text format by the auto save function.

Returns the current setting of the text save decimal as character data.

PERIOD: Use Period

COMMA: Use Comma

## Example

---

:CONFigure:SAVEDeci PERIOD

:CONFigure:SAVEDeci?

(Response) :CONFIGURE:SAVEDECI PERIOD (when headers are enabled)

## Note

---

When save separator is COMMA, save decimal cannot be set to COMMA.

## Usage Conditions

---

Sets and queries the text save date format.

**:CONFigure:SAVEFormat A\$**

## Syntax

---

### Command

:CONFigure:SAVEFormat A\$

### Query

:CONFigure:SAVEFormat?

### Response

A\$

A\$ = COMMENT, SPLITMS

## Details

---

Sets the text save date format when data is stored in text format by the auto save function.

Returns the current setting of the text save date format as character data.

COMMENT: yy-MM-dd hh:mm:ss.0

SPLITMS: yyyy-MM-dd hh:mm:ss + ms

## Example

---

:CONFigure:SAVEFormat COMMENT

:CONFigure:SAVEFormat?

(Response) :CONFIGURE:SAVEFORMAT COMMENT (when headers are enabled)

## Note

---

## Usage Conditions

---

Sets and queries the auto save division.

`:CONFigure:SAVEKind A$`

## Syntax

---

### Command

`:CONFigure:SAVEKind A$`

### Query

`:CONFigure:SAVEKind?`

### Response

A\$

A\$ = NORMAl, DIVide, REGUlarly

## Details

---

Sets the auto save division when data is stored by the auto save function.

Returns the current setting of the realtime save division as character data.

NORMAl: Disable

DIVide: Enable

REGUlarly: Fix Time

## Example

---

`:CONFigure:SAVEKind NORMAl`

`:CONFigure:SAVEKind?`

(Response) `:CONFIGURE:SAVEKIND NORMAL` (when headers are enabled)

## Note

---

## Usage Conditions

---

Sets and queries the realtime save division length.

`:CONFigure:SAVELen day,hour,min`

## Syntax

---

### Command

`:CONFigure:SAVELen day,hour,min`

### Query

`:CONFigure:SAVELen?`

### Response

`day<NR1>,hour<NR1>,min<NR1>`

day = 0 to 30 (day)

hour = 0 to 23 (hour)

min = 0 to 59 (min)

## Details

---

Sets the auto save division length to a numerical value.

Returns the currently set value of the realtime save division length as an NR1 numerical value.

## Example

---

`:CONFigure:SAVELen 0,0,10`

`:CONFigure:SAVELen?`

(Response) `:CONFIGURE:SAVELEN 0,0,10` (when headers are enabled)

## Note

---

## Usage Conditions

---

Sets and queries the realtime save mode.

**:CONFigure:SAVEMode A\$**

## Syntax

---

### Command

:CONFigure:SAVEMode A\$

### Query

:CONFigure:SAVEMode?

### Response

A\$

A\$ = FILEfull, REMove

## Details

---

Sets the realtime save mode when data is stored by the auto save function.

Returns the current setting of the auto save mode as character data.

FILEfull: OFF:If a media becomes full auto save is ended.

REMOve: ON:If a media becomes full remove oldest data file before saving.

## Example

---

:CONFigure:SAVEMode FILEfull

:CONFigure:SAVEMode?

(Response) :CONFIGURE:SAVEMODE FILEFULL (when headers are enabled)

## Note

---

## Usage Conditions

---

Sets and queries the realtime save priority.

`:CONFigure:SAVEPri A$`

## Syntax

---

### Command

`:CONFigure:SAVEPri A$`

### Query

`:CONFigure:SAVEPri?`

### Response

A\$

A\$ = SD, USB

## Details

---

Sets the realtime save priority when data is stored by the auto save function.

Returns the current setting of the realtime save priority as character data.

SD: SD Memory Card

USB: USB flash drive

## Example

---

`:CONFigure:SAVEPri SD`

`:CONFigure:SAVEPri?`

(Response) `:CONFIGURE:SAVEPRI SD` (when headers are enabled)

## Note

---

## Usage Conditions

---

## Sets and queries the reference time for autosave file splitting

`:CONFigure:SAVEReg hour,min`

### Syntax

---

#### Command

`:CONFigure:SAVEReg hour,min`

#### Query

`:CONFigure:SAVEReg?`

#### Response

hour<NR1>, min<NR1>

hour = 0 to 23 (hour)

min = 0 to 59 (min)

### Details

---

Set the reference time of the autosave file splitting function.

Returns the setting for the current autosave file split as a NR1 number.

### Example

---

`:CONFigure:SAVEReg 0,0`

`:CONFigure:SAVEReg?`

(Response) `:CONFIGURE:SAVEREG 0,0` (when headers are enabled)

### Note

---

### Usage Conditions

---

Sets and queries the text save separator.

`:CONFigure:SAVESep A$`

## Syntax

---

### Command

`:CONFigure:SAVESep A$`

### Query

`:CONFigure:SAVESep?`

### Response

A\$

A\$ = COMMA, SPACE, TAB, SEMI

## Details

---

Sets the text save separator when data is stored in text format by the auto save function.

Returns the current setting of the text save separator as character data.

COMMA: Use Comma

SPACE: Use Space

TAB: Use Tab

SEMI: Use Semicolon

## Example

---

`:CONFigure:SAVESep COMMA`

`:CONFigure:SAVESep?`

(Response) `:CONFIGURE:SAVESEP COMMA` (when headers are enabled)

## Note

---

When save decimal is COMMA, save separator cannot be set to COMMA.

## Usage Conditions

---

## Sets and queries the split time for autosave file splits

`:CONFigure:SAVETime A`

### Syntax

---

#### Command

`:CONFigure:SAVETime A`

#### Query

`:CONFigure:SAVETime?`

#### Response

A<NR1>

A =

1,2,5,10,15,20,30,60(1hour),120(2hour),180(3hour),240(4hour),360(6hour),480(8hour),720(12hour),1440(1day) (unit min)

### Details

---

Sets the split time for autosave file splits as a numeric value.

Returns the current split time of the autosave file split as an NR1 number.

### Example

---

`:CONFigure:SAVETime 1`

`:CONFigure:SAVETime?`

(Response) `:CONFIGURE:SAVETIME 1` (when headers are enabled)

### Note

---

If you specify a value that is not in the setting, it will be set to the nearest split time if there is a longer split time than the value you tried to set.

### Usage Conditions

---

Sets and queries the format for saving waveform Data.

**:CONFigure:SAVEWave A\$**

## Syntax

---

### Command

:CONFigure:SAVEWave A\$

### Query

:CONFigure:SAVEWave?

### Response

A\$

A\$ = OFF, BIN, CSV, MF4

## Details

---

Sets the format for saving waveform data by the auto save function.

Returns the current setting of the format for saving waveform data.

OFF: No Save

BIN: Save binary format

CSV: Save text format

MF4: MF4 format

## Example

---

:CONFigure:SAVEWave BIN

:CONFigure:SAVEWave?

(Response) :CONFigure:SAVEWave BIN (when headers are enabled)

## Note

---

Waveform data save format can be set by :CONFigure:ATSAve command.

## Usage Conditions

---

Sets and queries the measurement start.

**:CONFigure:START A\$**

## Syntax

---

### Command

:CONFigure:START A\$

### Query

:CONFigure:START?

### Response

A\$

A\$ = MANUAL, TIME

## Details

---

Sets the measurement start.

Returns the current setting of measurement start.

MANUAL: Unreserved

TIME: Reservation time

## Example

---

:CONFigure:START MANUAL

:CONFigure:START?

(Response) :CONFIGURE:START MANUAL (when headers are enabled)

## Note

---

The measurement start cannot be setted during measurement.

## Usage Conditions

---

Sets and queries the measurement start time.

**:CONFigure:STARTTime year,month,day,hour,minute**

## Syntax

---

### Command

**:CONFigure:STARTTime year,month,day,hour,minute**

### Query

**:CONFigure:STARTTime?**

### Response

year<NR1>,month<NR1>,day<NR1>,hour<NR1>,minute<NR1>

year = 21 to 37 (year)

month = 1 to 12 (month)

day = 1 to 31 (day)

hour = 0 to 23 (hour)

minute = 0 to 59 (minute)

## Details

---

Sets the measurement start.

Returns the current setting of measurement start time.

## Example

---

**:CONFigure:STARTTime 21,1,2,12,34**

**:CONFigure:STARTTime?**

(Response) **:CONFIGURE:STARTTIME 21,1,2,12,34** (when headers are enabled)

## Note

---

The measurement start time cannot be set during measurement.

The measurement start time cannot be set to a time later than the measurement stop time.

## Usage Conditions

---

Sets and queries the measurement stop.

**:CONFigure:STOP A\$**

## Syntax

---

### Command

:CONFigure:STOP A\$

### Query

:CONFigure:STOP?

### Response

A\$

A\$ = MANUAL, TIME

## Details

---

Sets the measurement stop.

Returns the current setting of measurement stop.

MANUAL: Unreserved

TIME: Reservation time

## Example

---

:CONFigure:STOP MANUAL

:CONFigure:STOP?

(Response) :CONFigure:STOP MANUAL (when headers are enabled)

## Note

---

The measurement stop cannot be setted during measurement.

## Usage Conditions

---

Sets and queries the measurement stop time.

**:CONFigure:STOPTime year,month,day,hour,minute**

## Syntax

---

### Command

:CONFigure:STOPTime year,month,day,hour,minute

### Query

:CONFigure:STOPTime?

### Response

year<NR1>,month<NR1>,day<NR1>,hour<NR1>,minute<NR1>

year = 21 to 37 (year)

month = 1 to 12 (month)

day = 1 to 31 (day)

hour = 0 to 23 (hour)

minute = 0 to 59 (minute)

## Details

---

Sets the measurement stop.

Returns the current setting of measurement stop time.

## Example

---

:CONFigure:STOPTime 21,1,2,12,34

:CONFigure:STOPTime?

(Response) :CONFIGURE:STOPTIME 21,1,2,12,34 (when headers are enabled)

## Note

---

The measurement stop time cannot be set during measurement.

The measurement stop time cannot be set to a time earlier than the measurement start time.

## Usage Conditions

---

Sets and queries the synchronous operation.

**:CONFigure:SYNC:SET A\$**

## Syntax

---

### Command

:CONFigure:SYNC:SET A\$

### Query

:CONFigure:SYNC:SET?

### Response

A\$

A\$ = OFF, PRIMary, SECondary

## Details

---

Sets the Primary, Secondary, and off for synchronous operation.

Returns the current setting of the synchronous operation.

OFF: Do not perform synchronous operation

PRIMary: Primary

SECondary: Secondary

## Example

---

:CONFigure:SYNC:SET PRIMary

:CONFigure:SYNC:SET?

(Response) :CONFIGURE:SYNC:SET PRIMARY (when headers are enabled)

## Note

---

Synchronous operation cannot be set during measurement.

## Usage Conditions

---

Sets and queries the wiring of optical connection cables.

`:CONFigure:SYNC:CHECK?`

## Syntax

---

### Query

`:CONFigure:SYNC:CHECK?`

### Response

A<NR1>

A = 0 to 255

## Details

---

Returns the result of the following wiring checks as NR1 numbers.

For example, if 1 is returned, it means that the instrument's setting is not PRIMARY.

bit0: The bit stands when the setting of synchronous operation is not PRIMARY.

bit4: The bit is set when the number of secondaries may exceed 9.

bit7: The bit is set if the optical connection cable may be disconnected.

Other bits are fixed at 0.

## Example

---

`:CONFigure:SYNC:CHECK?`

(Response) `:CONFIGURE:SYNC:CHECK 1` (when headers are enabled)

## Note

---

## Usage Conditions

---

## Sets and queries the downsampling type for auto save

`:CONFigure:THINData A$`

### Syntax

---

#### Command

`:CONFigure:THINData A$`

#### Query

`:CONFigure:THINData?`

#### Response

A\$

A\$ = INSTANT, STATISTICS

### Details

---

Sets the downsampling type for auto save

Returns the downsampling type for auto save

INSTANT: Saves the data at the beginning.

STATISTICS: Saves statistical data (maximum value, minimum value, average value, and data at the beginning).

### Example

---

`:CONFigure:THINData INSTANT`

`:CONFigure:THINData?`

(Response) `:CONFIGURE:THINDATA INSTANT` (when headers are enabled)

### Note

---

Sets the save format to CSV and sets 2 or more in the parameter of `:CONFigure:THINOut` command.

### Usage Conditions

---

Sets and queries the decimation factor for auto save.

**:CONFigure:THINOut A**

## Syntax

---

### Command

:CONFigure:THINOut A

### Query

:CONFigure:THINOut?

### Response

A<NR1>

A = 1(OFF) to 100000

## Details

---

Sets the decimation factor for auto save.

Returns the currently set value of the decimation factor as an NR1 numerical value.

If A=1, downsampling is off.

## Example

---

:CONFigure:THINOut 1000

:CONFigure:THINOut?

(Response) :CONFIGURE:THINOUT 1000 (when headers are enabled)

## Note

---

Sets the save format to CSV.

## Usage Conditions

---

## 3.5 Channel(MODule)

Execute zero adjustment of input modules and queries the result.

:MODule:ADJUST?

### Syntax

---

#### Query

:MODule:ADJUST?

#### Response

A<NR1>

A = 0, 1

### Details

---

Performs zero adjustment of input modules.

Returns the result of zero adjustment of input modules.

A = 0: Pass

A = 1: Fail

### Example

---

:MODule:ADJUST?

(Response) :MODULE:ADJUST 1 (when headers are enabled)

### Note

---

### Usage Conditions

---

Sets and queries the data update interval for module.

**:MODule:DATARate module\$,A**

## Syntax

---

### Command

:MODule:DATARate module\$,A

### Query

:MODule:DATARate? module\$

### Response

module\$,A<NR3> (1 decimal places)  
module\$ = MODULE1 to MODULE10  
A = 0(Auto), 5.0E-3 to 1.0E+1 (sec)

## Details

---

Sets the data update interval to a numerical value (unit seconds) for the module designated by module\$.

Returns the value of the data update interval for the module designated by module\$ as an NR3 numerical value.

if A=0, the data update interval is automatically.

## Example

---

:MODule:DATARate MODULE1,1E-1  
:MODule:DATARate? MODULE1  
(Response) :MODULE:DATARATE MODULE1,1.0E-01 (when headers are enabled)

## Note

---

A value greater than or equal to the recording interval can be set. If the recording interval is 10s or longer, it is fixed at 10s.

If a value not in the settings is specified, and there exists an update interval higher than the value attempted to be set, it will be set to the closest update interval.

## Usage Conditions

---

Queries the digital filter values for the specified module.

`:MODule:DFILter? module$`

## Syntax

---

### Query

`:MODule:DFILter? module$`

### Response

`module$,A<NR3>`

`module$ = MODULE1 to MODULE10`

## Details

---

Returns the digital filter value for the specified module.

## Example

---

`:MODule:DFILter? MODULE1`

(Response) `:MODULE:DFILTER MODULE1 2.4E+03` (when headers are enabled)

## Note

---

## Usage Conditions

---

Sets and queries the format for saving mathematical results.

**:MODule:FILTer A\$**

## Syntax

---

### Command

:MODule:FILTer A\$

### Query

:MODule:FILTer?

### Response

A\$

A\$ = 50HZ, 60HZ

## Details

---

Sets the power frequency filter.

Returns the power frequency filter as character data.

## Example

---

:MODule:FILTer 50HZ

:MODule:FILTer?

(Response) :MODULE:FILTER 50HZ (when headers are enabled)

## Note

---

## Usage Conditions

---

Queries the module ID.

`:MODule:IDN? module$`

## Syntax

---

### Query

`:MODule:IDN? module$`

### Response

`module$,A$,B$,C$,D$`

`module$ = MODULE1 to MODULE10`

`A$ = model number`

`B$ = serial number`

`C$ = software version`

`D$ = FPGA version`

## Details

---

Returns the module ID for the module designated by `module$`.

## Example

---

`:MODule:IDN? MODULE1`

(Response) `:MODULE:IDN MODULE1,M7100,100000000,V 100,V 100` (when headers are enabled)

## Note

---

Returns "UNKNOWN" if the module is not connected.

## Usage Conditions

---

Sets and queries the measurement mode of an input channel.

**:MODule:INMOde ch\$,A\$**

## Syntax

### Command

:MODule:INMOde ch\$,A\$

### Query

:MODule:INMOde? ch\$

### Response

ch\$,A\$

ch\$ = CH1\_1 to CH10\_30

A\$ = VOLTAGE, TC

## Details

Sets the measurement mode for the channel designated by ch\$.

Returns the current measurement mode for the channel designated by ch\$ as character data.

VOLTAGE: Voltage

TC: Thermocouple

## Example

:MODule:INMOde CH1\_1,VOLTAGE

:MODule:INMOde? CH1\_1

(Response) :MODULE:INMODE CH1\_1,VOLTAGE (when headers are enabled)

## Note

Module		Input Type
M7100	Voltage/Temp Module(15Ch)	VOLTAGE TC
M7102	Voltage/Temp Module(30Ch)	VOLTAGE TC

## Usage Conditions

---

Sets and queries the count mode of an input pulse channel.

`:MODULE:PCOMode pls$,A$`

## Syntax

---

### Command

`:MODULE:PCOMode pls$,A$`

### Query

`:MODULE:PCOMode? pls$`

### Response

pls\$,A\$  
pls\$ = PLS1  
A\$ = ADD, INST

## Details

---

Sets the count mode for the pulse channel designated by pls\$.  
Returns the current count mode for the pulse channel designated by pls\$ as character data.  
ADD: Addition  
INST: Instant

## Example

---

`:MODULE:PCOMode PLS1,ADD`  
`:MODULE:PCOMode? PLS1`  
(Response) `:MODULE:PCOMODE PLS1,ADD` (when headers are enabled)

## Note

---

## Usage Conditions

---

Sets and queries the counting start timing of an input pulse channel.

**:MODule:PCOSTart pls\$,A\$**

## Syntax

---

### Command

:MODule:PCOSTart pls\$,A\$

### Query

:MODule:PCOSTart? pls\$

### Response

pls\$,A\$  
pls\$ = PLS1  
A\$ = START, TRIGger

## Details

---

Sets the counting start timing for the pulse channel designated by pls\$.  
Returns the current counting start timing for the pulse channel designated by pls\$ as character data.  
START: Start  
TRIGger: Trigger

## Example

---

:MODule:PCOSTart PLS1,START  
:MODule:PCOSTart? PLS1  
(Response) :MODULE:PCOSTART PLS1,START (when headers are enabled)

## Note

---

## Usage Conditions

---

Sets and queries the pulse num per revolve.

:MODULE:PCOunt pls\$,A

## Syntax

---

### Command

:MODULE:PCOunt pls\$,A

### Query

:MODULE:PCOunt? pls\$

### Response

pls\$,A<NR1>  
pls\$ = PLS1  
A = 1 to 1000

## Details

---

Sets the pulse num per revolve for the pulse channel designated by pls\$ in the range to a numerical value.

Returns the current pulse num per revolve for the pulse channel designated by pls\$ as an NR1 numerical value.

## Example

---

:MODULE:PCOunt PLS1,1  
:MODULE:PCOunt? PLS1  
(Response) :MODULE:PCOUNT PLS1,1 (when headers are enabled)

## Note

---

## Usage Conditions

---

Sets and queries the input pulse channel filter.

```
:MODule:PFILTer pls$,A$
```

## Syntax

---

### Command

```
:MODule:PFILTer pls$,A$
```

### Query

```
:MODule:PFILTer? pls$
```

### Response

```
pls$,A$  
pls$ = PLS1  
A$ = OFF, ON
```

## Details

---

Sets the filter for the pulse channel designated by pls\$.

Returns the current filter for the pulse channel designated by pls\$ as character data.

## Example

---

```
:MODule:PFILTer PLS1,ON  
:MODule:PFILTer? PLS1  
(Response) :MODULE:PFILTER PLS1,ON (when headers are enabled)
```

## Note

---

## Usage Conditions

---

Sets and queries the measurement mode of an input pulse channel.

`:MODule:PINMOde pls$,A$`

## Syntax

---

### Command

`:MODule:PINMOde pls$,A$`

### Query

`:MODule:PINMOde? pls$`

### Response

`pls$,A$`

`pls$ = PLS1`

`A$ = COUNT, REVOLVE, LOGIC`

## Details

---

Sets the measurement mode for the pulse channel designated by `pls$`.

Returns the current measurement mode for the pulse channel designated by `pls$` as character data.

COUNT: Integration

REVOLVE: Rotational speed

LOGIC: Logic

## Example

---

`:MODule:PINMOde PLS1,COUNT`

`:MODule:PINMOde? PLS1`

(Response) `:MODULE:PINMODE PLS1,COUNT` (when headers are enabled)

## Note

---

## Usage Conditions

---

Sets and queries the input pulse channel revolve range.

`:MODule:PRANGe pls$,A$`

## Syntax

---

### Command

`:MODule:PRANGe pls$,A$`

### Query

`:MODule:PRANGe? pls$`

### Response

pls\$,A\$  
pls\$ = PLS1  
A\$ = RPS, RPM

## Details

---

Sets the revolve range of the pulse channel specified by pls\$.  
Returns the revolve range of the pulse channel specified by pls\$ as character data.  
RPS: 5,000 r/s  
RPM: 300,000 r/min

## Example

---

`:MODule:PRANGe PLS1,RPM`  
`:MODule:PRANGe? PLS1`  
(Response) `:MODULE:PRANGE PLS1,RPM` (when headers are enabled)

## Note

---

## Usage Conditions

---

If the pulse input type is other than rotational speed, a command error occurs.

Sets and queries the input pulse channel reset.

`:MODule:PRESet plS$,A$`

## Syntax

---

### Command

`:MODule:PRESet plS$,A$`

### Query

`:MODule:PRESet? plS$`

### Response

plS\$,A\$  
plS\$ = PLS1  
A\$ = OFF, ON

## Details

---

Sets the reset for the pulse channel designated by plS\$.

Returns the current reset for the pulse channel designated by plS\$ as character data.

## Example

---

`:MODule:PRESet PLS1,ON`  
`:MODule:PRESet? PLS1`  
(Response) `:MODULE:PRESET PLS1,ON` (when headers are enabled)

## Note

---

## Usage Conditions

---

Sets and queries the pulse channel count slope.

`:MODule:PSLOPe pls$,A$`

## Syntax

---

### Command

`:MODule:PSLOPe pls$,A$`

### Query

`:MODule:PSLOPe? pls$`

### Response

pls\$,A\$  
pls\$ = PLS1  
A\$ = UP, DOWN

## Details

---

Sets the count slope for the pulse channel designated by pls\$.  
Returns the current count slope for the pulse channel designated by pls\$ as character data.

## Example

---

`:MODule:PSLOPe PLS1,UP`  
`:MODule:PSLOPe? PLS1`  
(Response) `:MODULE:PSLOPE PLS1,UP` (when headers are enabled)

## Note

---

## Usage Conditions

---

Sets and queries the smoothing for revolve.

`:MODULE:PSmooth pls$,A`

## Syntax

---

### Command

`:MODULE:PSmooth pls$,A`

### Query

`:MODULE:PSmooth? pls$`

### Response

`pls$,A<NR1>`  
`pls$ = PLS1`  
`A = 1(OFF) to 60`

## Details

---

Sets the smoothing for revolve for the pulse channel designated by pls\$.  
Returns the current smoothing for revolve for the pulse channel designated by pls\$ as an NR1 numerical value.  
If A=1, smoothing is off.

## Example

---

`:MODULE:PSmooth PLS1,1`  
`:MODULE:PSmooth? PLS1`  
(Response) `:MODULE:PSMOOTH PLS1,1` (when headers are enabled)

## Note

---

## Usage Conditions

---

Sets and queries the pulse threshold level.

`:MODule:PTHRe pls$,A$`

## Syntax

---

### Command

`:MODule:PTHRe pls$,A$`

### Query

`:MODule:PTHRe? pls$`

### Response

pls\$,A\$  
pls\$ = PLS1  
A\$ = 1V, 4V

## Details

---

Sets the pulse threshold level for the pulse channel designated by pls\$.  
Returns the current pulse threshold level for the pulse channel designated by pls\$ as character data.

## Example

---

`:MODule:PTHRe PLS1,1V`  
`:MODule:PTHRe? PLS1`  
(Response) `:MODULE:PTHRE PLS1,1V` (when headers are enabled)

## Note

---

## Usage Conditions

---

Sets and queries the measurement range of an input channel.

**:MODule:RANGe ch\$,A**

## Syntax

---

### Command

:MODule:RANGe ch\$,A

### Query

:MODule:RANGe? ch\$

### Response

ch\$,A<NR3> (1 decimal places)

ch\$ = CH1\_1 to CH10\_30

A = measurement range

## Details

---

Sets the measurement range for the channel designated by ch\$ to a numerical value.

Returns the current measurement range for the channel designated by ch\$ as an NR3 numerical value.

## Example

---

:MODule:RANGe CH1\_1,1E-1

:MODule:RANGe? CH1\_1

(Response) :MODULE:RANGE CH1\_1,+1.0E-01 (when headers are enabled)

## Note

---

If an attempt is made to set the measurement range to a non-permitted value, and there is a range above that value, that range will be selected.

Please set up A=15, when it makes the range of 1-5V with voltage.

## Usage Conditions

---

Temperature Sensor cannot be set B in 100°C or 500°C range.

Sets and queries the point of contact compensation for tc mode.

**:MODule:RJC ch\$,A**

## Syntax

---

### Command

:MODule:RJC ch\$,A

### Query

:MODule:RJC? ch\$

### Response

ch\$,A\$

ch\$ = CH1\_1 to CH10\_30

A = INT, EXT

## Details

---

Sets the point of contact compensation for the channel designated by ch\$.

Returns the current point of contact compensation for the channel designated by ch\$ as character data.

INT: Internal

EXT: External

## Example

---

:MODule:RJC CH1\_1,INT

:MODule:RJC? CH1\_1

(Response) :MODULE:RJC CH1\_1,INT (when headers are enabled)

## Note

---

## Usage Conditions

---

Sets and queries the sensor kind for tc mode.

**:MODule:SENSor ch\$,A\$**

## Syntax

---

### Command

:MODule:SENSor ch\$,A\$

### Query

:MODule:SENSor? ch\$

### Response

ch\$,A\$

ch\$ = CH1\_1 to CH10\_30

A\$ = K, J, E, T, N, R, S, B, C

## Details

---

Sets the sensor kind for the channel designated by ch\$.

Returns the current sensor kind for the channel designated by ch\$ as character data.

## Example

---

:MODule:SENSor CH1\_1,K

:MODule:SENSor? CH1\_1

(Response) :MODULE:SENSOR CH1\_1,K (when headers are enabled)

## Note

---

Sensor cannot be set to B at temperature range is 100 or 500.

## Usage Conditions

---

Sets and queries the store enable or disable for channel data record.

`:MODule:STORe ch$,A$`

## Syntax

---

### Command

`:MODule:STORe ch$,A$`

### Query

`:MODule:STORe? ch$`

### Response

`ch$,A$`

`ch$ = CH1_1 to CH10_30, PLS1, LOG, ALARM, W1 to W30`

`A$ = OFF, ON`

## Details

---

Sets the store enable or disable for the channel designated by `ch$`.

Returns the current store enable or disable for the channel designated by `ch$` as character data.

## Example

---

`:MODule:STORe CH1_1,ON`

`:MODule:STORe? CH1_1`

(Response) `:MODULE:STORE CH1_1,ON` (when headers are enabled)

## Note

---

Pulse channels with measurement mode set to LOGIC cannot be set.

If there is no pulse channel with measurement mode set to LOGIC, it cannot be set with `ch$=LOG`.

## Usage Conditions

---

Sets and queries the disconnection detection for tc mode.

`:MODule:WIRE module$,A$`

## Syntax

---

### Command

`:MODule:WIRE module$,A$`

### Query

`:MODule:WIRE? module$`

### Response

module\$,A\$  
module\$ = MODULE1 to MODULE10  
A\$ = OFF, ON

## Details

---

Sets the disconnection detection for the module designated by module\$.  
Returns the current disconnection detection for the module designated by module\$ as character data.

## Example

---

`:MODule:WIRE MODULE1,ON`  
`:MODule:WIRE? MODULE1`  
(Response) `:MODULE:WIRE MODULE1,ON` (when headers are enabled)

## Note

---

## Usage Conditions

---

## 3.6 Scaling(SCALing)

Sets and queries the type of scaling.

```
:SCALing:KIND ch$,A$
```

### Syntax

---

#### Command

```
:SCALing:KIND ch$,A$
```

#### Query

```
:SCALing:KIND? ch$
```

#### Response

```
ch$,A$  
ch$ = CH1_1 to CH10_30, PLS1  
A$ = RATIO, POINT, SENS
```

### Details

---

Sets the type of scaling designated by ch\$.  
Returns the current type of scaling designated by ch\$ as a character string.  
RATIO: Scale by specifying a conversion ratio  
POINT: Scale by specifying two data points  
SENS: Scale by specifying a sensitivity

### Example

---

```
:SCALing:KIND CH1_1,POINT  
:SCALing:KIND? CH1_1  
(Response) :SCALING:KIND CH1_1,POINT (when headers are enabled)
```

### Note

---

The setting method depends on the module type.

### Usage Conditions

---

Sets and queries the scaling offset.

**:SCALing:OFFSet ch\$,A**

## Syntax

---

### Command

:SCALing:OFFSet ch\$,A

### Query

:SCALing:OFFSet? ch\$

### Response

ch\$,A<NR3> (4 decimal places)  
ch\$ = CH1\_1 to CH10\_30, PLS1  
A\$ = -9.9999E+09 to 9.9999E+09

## Details

---

Sets the scaling offset for the channel designated by ch\$.  
Returns the current scaling offset for the channel designated by ch\$ as an NR3 numerical value.

## Example

---

:SCALing:OFFSet CH1\_1,0  
:SCALing:OFFSet? CH1\_1  
(Response) :SCALING:OFFSET CH1\_1,+0.0000E+00 (when headers are enabled)

## Note

---

The possible values for A are the same as for the main module.  
This setting may change the following settings

- Scaling SCALE UP and LOW
- Thresholds for Numerical Calculation

## Usage Conditions

---

Sets and queries the scaling SCALE UP and LOW.

`:SCALing:SCUPLOW ch$,A,B`

## Syntax

---

### Command

`:SCALing:SCUPLOW ch$,A,B`

### Query

`:SCALing:SCUPLOW? ch$`

### Response

`ch$,A<NR3>,B<NR3>` (4 decimal places)

`ch$ = CH1_1 to CH10_30, PLS1`

`A, B = -9.9999E+29 to +9.9999E+29`

## Details

---

Sets the scaling SC UP and SC LOW values for the channel designated by ch\$.

Returns the current scaling SC UP and SC LOW values for the channel designated by ch\$ as an NR3 numerical value.

## Example

---

`:SCALing:SCUPLOW CH1_1,0.5,-0.5`

`:SCALing:SCUPLOW? CH1_1`

(Response) `:SCALING:SCUPLOW CH1_1,+5.0000E-01,-5.0000E-01` (when headers are enabled)

## Note

---

Same value in A and B can not be set.

This setting may change the following settings

- Scaling offset
- Scaling rated capacity
- Scaling rated output
- Scaling sense value
- Scaling conversion value

## Usage Conditions

---

Sets and queries the scaling sense value.

`:SCALing:SENSE ch$,A`

## Syntax

---

### Command

`:SCALing:SENSE ch$,A`

### Query

`:SCALing:SENSE? ch$`

### Response

ch\$,A<NR3> (4 decimal places)  
ch\$ = CH1\_1 to CH10\_30, PLS1  
A = -1.0000E+09 to +1.0000E+09

## Details

---

Sets the scaling sense value for the channel designated by ch\$.  
Returns the current scaling sense value setting for the channel designated by ch\$ as an NR3 numerical value.

## Example

---

`:SCALing:SENSE CH1_1,1`  
`:SCALing:SENSE? CH1_1`  
(Response) `:SCALING:SENSE CH1_1,+1.0000E+00` (when headers are enabled)

## Note

---

The possible values for A are the same as for the main module.  
This setting may change the following settings

- Scaling rated capacity
- Scaling rated output
- Scaling SCALE UP and LOW
- Scaling conversion value

## Usage Conditions

---

Sets and queries the scaling kind.

**:SCALing:SET ch\$,A\$**

## Syntax

---

### Command

**:SCALing:SET ch\$,A\$**

### Query

**:SCALing:SET? ch\$**

### Response

ch\$,A  
ch\$ = CH1\_1 to CH10\_30, PLS1  
A\$ = OFF, ENG, SCI

## Details

---

Sets the scaling kind designated by ch\$.

Returns the current scaling kind designated by ch\$ as a character string.

OFF: Scaling is disabled

ENG: Decimal: Scaling is enabled and values are displayed using engineering notation

SCI: Exponential: Scaling is enabled and values are displayed using scientific notation

## Example

---

**:SCALing:SET CH1\_1,ENG**

**:SCALing:SET? CH1\_1**

(Response) **:SCALING:SET CH1\_1,ENG** (when headers are enabled)

## Note

---

## Usage Conditions

---

Sets and queries the scaling unit.

**:SCALing:UNIT ch\$,"A\$"**

## Syntax

---

### Command

:SCALing:UNIT ch\$,"A\$"

### Query

:SCALing:UNIT? ch\$

### Response

ch\$,"A\$"

ch\$ = CH1\_1 to CH10\_30, PLS1

A\$ = scaling unit(up to 7 characters)

## Details

---

Sets the scaling unit for the channel designated by ch\$.

Returns the current scaling unit for the channel designated by ch\$ as character data.

Single quotation marks (') can be used instead of double quotation marks("").

Entry of the special characters is as follows.

PC	^2	^3	~u	~o	~e	~c	~+	~,	~;	^^	~~
LR8101 LR8102	2	3	μ	Ω	ε	o	±	'	"	^	~

## Example

---

:SCALing:UNIT CH1\_1,"mA"

:SCALing:UNIT? CH1\_1

(Response) :SCALING:UNIT CH1\_1,"mA" (when headers are enabled)

## Note

---

Characters exceeding the maximum number of characters are not set.

## Usage Conditions

---

Sets and queries the scaling conversion value.(RATIO)

:SCALing:VOLT ch\$,A

## Syntax

---

### Command

:SCALing:VOLT ch\$,A

### Query

:SCALing:VOLT? ch\$

### Response

ch\$,A<NR3> (4 decimal places)

ch\$ = CH1\_1 to CH10\_30, PLS1

A = -9.9999EE+09 to +9.9999E+09(pulse channel mode = COUNT:+1.0000E-09 to +9.9999E+09)

## Details

---

Sets the scaling conversion value for the channel designated by ch\$.

Returns the current scaling conversion value setting for the channel designated by ch\$ as an NR3 numerical value.

## Example

---

:SCALing:VOLT CH1\_1,1

:SCALing:VOLT? CH1\_1

(Response) :SCALING:VOLT CH1\_1,+1.0000E+00 (when headers are enabled)

## Note

---

The possible values for A are the same as for the main module.

This setting may change the following settings

- Scaling rated capacity
- Scaling rated output
- Scaling SCALE UP and LOW
- Scaling sense value

## Usage Conditions

---

Sets and queries the scaling VOLT UP and LOW.

`:SCALing:VOUPLOW ch$,A,B`

## Syntax

---

### Command

`:SCALing:VOUPLOW ch$,A,B`

### Query

`:SCALing:VOUPLOW? A$`

### Response

`ch$,A<NR3>,B<NR3>` (4 decimal places)

`ch$ = CH1_1 to CH10_30, PLS1`

`A, B = -9.9999E+29 to +9.9999E+29`

## Details

---

Sets the scaling VOLT UP and VOLT LOW values for the channel designated by ch\$.

Returns the current scaling VOLT UP and VOLT LOW values for the channel designated by ch\$ as an NR3 numerical value.

## Example

---

`:SCALing:VOUPLOW CH1_1,0.05,-0.05`

`:SCALing:VOUPLOW? CH1_1`

(Response) `:SCALING:VOUPLOW CH1_1,+5.0000E-02,-5.0000E-02` (when headers are enabled)

## Note

---

Same value in A and B can not be set.

This setting may change the following settings

- Scaling offset
- Scaling rated capacity
- Scaling rated output
- Scaling sense value
- Scaling conversion value

## Usage Conditions

---

## 3.7 Comments(COMMent)

Sets and queries the comment characters for alarm channel.

```
:COMMent:ALMCH alm$,"A$"
```

### Syntax

#### Command

```
:COMMent:ALMCH alm$,"A$"
```

#### Query

```
:COMMent:ALMCH? alm$
```

#### Response

```
alm$,"A$"
```

alm\$ = ALM1 to ALM4

A\$ = comment characters (up to 40 characters)

### Details

Sets a string of comment characters for the alarm channel specified by alm\$.

Returns a string of comment characters for the alarm channel specified by alm\$ as character data.

Single quotation marks (') can be used instead of double quotation marks("").

Entry of the special characters is as follows.

PC	^2	^3	~u	~o	~e	~c	~+	~,	~;	^^	~~
LR8101 LR8102	2	3	μ	Ω	ε	o	±	'	"	^	~

### Example

```
:COMMent:ALMCH ALM1,"ABCDEFGH"
```

```
:COMMent:ALMCH? ALM1
```

(Response) :COMMENT:ALMCH ALM1,"ABCDEFGH" (when headers are enabled)

### Note

Characters exceeding the maximum number of characters are not set.

The alarm hannel comment settings cannot be changed during measurement.

## Usage Conditions

---

Sets and queries the comment characters for each channel.

```
:COMMeNt:CH ch$,"A$"
```

## Syntax

---

### Command

```
:COMMeNt:CH ch$,"A$"
```

### Query

```
:COMMeNt:CH? ch$
```

### Response

```
ch$,"A$"  
ch$ = CH1_1 to CH10_30, PLS1, W1 to W30  
A$ = comment characters (up to 40 characters)
```

## Details

---

Sets a string of comment characters for the channel specified by ch\$.

Returns a string of comment characters for the channel specified by ch\$ as character data.

Single quotation marks (') can be used instead of double quotation marks("").

Entry of the special characters is as follows.

PC	^2	^3	~u	~o	~e	~c	~+	~,	~;	^^	~~
LR8101 LR8102	2	3	μ	Ω	ε	o	±	'	"	^	~

## Example

---

```
:COMMeNt:CH CH1_1,"ABCDEFGG"  
:COMMeNt:CH? CH1_1  
(Response) :COMMENT:CH CH1_1,"ABCDEFGG" (when headers are enabled)
```

## Note

---

Characters exceeding the maximum number of characters are not set.

The channel comment settings cannot be changed during measurement.

## Usage Conditions

---

Sets and queries the title comments, and inputs comment characters.

`:COMMeNt:TITLe "A$"`

## Syntax

---

### Command

`:COMMeNt:TITLe "A$"`

### Query

`:COMMeNt:TITLe?`

### Response

"A\$"

A\$ = comment characters (up to 40 characters)

## Details

---

Sets a string of title comment characters.

Returns a string of title comment characters as character data.

Single quotation marks (') can be used instead of double quotation marks("").

Entry of the special characters is as follows.

PC	$\wedge 2$	$\wedge 3$	$\sim u$	$\sim o$	$\sim e$	$\sim c$	$\sim +$	$\sim ,$	$\sim ;$	$\wedge \wedge$	$\sim \sim$
LR8101 LR8102	2	3	$\mu$	$\Omega$	$\epsilon$	o	$\pm$	'	"	$\wedge$	$\sim$

## Example

---

`:COMMeNt:TITLe "HIOKI"`

`:COMMeNt:TITLe?`

(Response) `:COMMeNt:TITLe "HIOKI"` (when headers are enabled)

## Note

---

Characters exceeding the maximum number of characters are not set.

The title comment settings cannot be changed during measurement.

## Usage Conditions

---

Sets and queries the identifier characters for each module.

`:COMMeNt:MODUle module$,"A$"`

## Syntax

---

### Command

`:COMMeNt:MODUle module$,"A$"`

### Query

`:COMMeNt:MODUle? module$`

### Response

module\$,"A\$"

module\$ = MODULE1 to MODULE10

A\$ = identifier characters (up to 16 characters)

## Details

---

Sets a string of identifier characters for the each module specified by module\$.

Returns a string of identifier characters for the each module specified by module\$ as character data.

Single quotation marks (') can be used instead of double quotation marks("").

Entry of the special characters is as follows.

PC	$\wedge 2$	$\wedge 3$	$\sim u$	$\sim o$	$\sim e$	$\sim c$	$\sim +$	$\sim ,$	$\sim ;$	$\wedge \wedge$	$\sim \sim$
LR8101 LR8102	2	3	$\mu$	$\Omega$	$\epsilon$	o	$\pm$	'	"	$\wedge$	$\sim$

## Example

---

`:COMMeNt:MODUle MODULE1,"ABCDEFGH"`

`:COMMeNt:MODUle? MODULE1`

(Response) `:COMMeNt:MODUle MODULE1,"ABCDEFGH"` (when headers are enabled)

## Note

---

Characters exceeding the maximum number of characters are not set.

The identifier characters for each module cannot be changed during measurement.

## Usage Conditions

---

## 3.8 Triggering(TRIGger)

Queries the date for trigger detection.

`:TRIGger:DETECTDate?`

### Syntax

---

#### Query

`:TRIGger:DETECTDate?`

#### Response

`year<NR1>,month<NR1>,day<NR1>`

year = 00 to 99 (year)

month = 01 to 12 (month)

day = 01 to 31 (day)

### Details

---

Returns the setting for the date for trigger detection as a numerical value in NR1 format.

### Example

---

`:TRIGger:DETECTDate?`

(Response) `:TRIGGER:DETECTDATE 19,12,26` (when headers are enabled)

### Note

---

Returns 00,00,00 if no storage data exists.

Returns the measurement start date if the start trigger is OFF.

### Usage Conditions

---

Queries the time point for trigger detection.

:TRIGger:DETECTTime?

## Syntax

---

### Query

:TRIGger:DETECTTime?

### Response

hour<NR1>,min<NR1>,sec<NR1>,ms<NR1>

hour = 00 to 23 (hour)

min = 01 to 59 (min)

sec = 01 to 59 (sec)

ms = 000 to 999(ms)

## Details

---

Returns the setting for the time point for trigger.

## Example

---

:TRIGger:DETECTTime?

(Response) :TRIGGER:DETECTTIME 01,02,03,004 (when headers are enabled)

## Note

---

Returns 00,00,00,000 if no storage data exists.

Returns the measurement start date if the start trigger is OFF.

## Usage Conditions

---

## Manually trigger measurement

`:TRIGger:MANUal`

### Syntax

---

#### Command

`:TRIGger:MANUal`

### Details

---

Sends this command to trigger the instrument while it is awaiting a trigger.

### Example

---

`:TRIGger:MANUal`

### Note

---

### Usage Conditions

---

Waiting for trigger or pre-trigger.

Sets and queries the Repetitive recording.

`:TRIGger:MODE A$`

## Syntax

---

### Command

`:TRIGger:MODE A$`

### Query

`:TRIGger:MODE?`

### Response

A\$

A\$ = SINGle, REPEat

## Details

---

Sets the repetitive recording.

Returns the current repetitive recording as character data.

SINGle: repeat off

REPEat: repeat on

## Example

---

`:TRIGger:MODE REPEat`

`:TRIGger:MODE?`

(Response) `:TRIGGER:MODE REPEAT` (when headers are enabled)

## Note

---

Changing the setting may change the interval trigger setting.

## Usage Conditions

---

Sets and queries the pre-trigger.

`:TRIGger:PRETrig day,hour,min,sec`

## Syntax

---

### Command

`:TRIGger:PRETrig day,hour,min,sec`

### Query

`:TRIGger:PRETrig?`

### Response

`day<NR1>,hour<NR1>,min<NR1>,sec<NR1>`

day = 0 to 99(day)

hour = 0 to 23(hour)

min = 0 to 59(min)

sec = 0 to 59(sec)

## Details

---

Sets pre-trigger value to a numerical value.

Returns the current pre-trigger value as an NR1 numerical value.

## Example

---

`:TRIGger:PRETrig 0,0,0,10`

`:TRIGger:PRETrig?`

(Response) `:TRIGGER:PRETRIG 0,0,0,10` (when headers are enabled)

## Note

---

The setting of pre-trigger be limited by a set value of recording interval.

## Usage Conditions

---

When the trigger timing is other than STOP.

Sets and queries the trigger usage.

`:TRIGger:SET A$`

## Syntax

---

### Command

`:TRIGger:SET A$`

### Query

`:TRIGger:SET?`

### Response

A\$

A\$ = OFF, ON

## Details

---

Sets the trigger usage.

Returns the trigger usage as character data.

OFF: Disabled

ON: Enabled

## Example

---

`:TRIGger:SET ON`

`:TRIGger:SET?`

(Response) `:TRIGGER:SET ON` (when headers are enabled)

## Note

---

Depending on this setting, the input type of external input terminal 3 may be changed.

## Usage Conditions

---

Sets and queries the start trigger logical operator (AND/OR).

**:TRIGger:SOURce A\$**

## Syntax

---

### Command

:TRIGger:SOURce A\$

### Query

:TRIGger:SOURce?

### Response

A\$

A\$ = OR, AND

## Details

---

Sets the AND/OR logical operator for combining start trigger sources.

Returns the current setting of the start trigger AND/OR logical operator as character data.

AND: Logical product

OR: Logical sum

## Example

---

:TRIGger:SOURce AND

:TRIGger:SOURce?

(Response) :TRIGGER:SOURCE AND (when headers are enabled)

## Note

---

## Usage Conditions

---

Sets and queries the stop trigger logical operator (AND/OR).

`:TRIGger:SSOURce A$`

## Syntax

---

### Command

`:TRIGger:SSOURce A$`

### Query

`:TRIGger:SSOURce?`

### Response

A\$

A\$ = OR, AND

## Details

---

Sets the AND/OR logical operator for combining stop trigger sources.

Returns the current setting of the stop trigger AND/OR logical operator as character data.

AND: Logical product

OR: Logical sum

## Example

---

`:TRIGger:SSOURce AND`

`:TRIGger:SSOURce?`

(Response) `:TRIGGER:SSOURCE AND` (when headers are enabled)

## Note

---

## Usage Conditions

---

Sets and queries the interval trigger.

**:TRIGger:TIMEr A\$**

## Syntax

---

### Command

:TRIGger:TIMEr A\$

### Query

:TRIGger:TIMEr?

### Response

A\$

A\$ = OFF, OR, AND

## Details

---

Sets the interval trigger.

Returns the current interval trigger setting as character data.

OFF: Disabled

OR: Logical sum

AND: Logical product

## Example

---

:TRIGger:TIMEr OR

:TRIGger:TIMEr?

(Response) :TRIGGER:TIMER OR (when headers are enabled)

## Note

---

Changing this setting may change the Repetitive recording settings.

## Usage Conditions

---

Sets and queries the trigger timing.

**:TRIGger:TIMIng A\$**

## Syntax

---

### Command

:TRIGger:TIMIng A\$

### Query

:TRIGger:TIMIng?

### Response

A\$

A\$ = START, STOP, S\_S

## Details

---

Sets the trigger timing.

Returns the trigger timing as character data.

START: Start trigger

STOP: Stop trigger

S\_S: Start & Stop trigger

## Example

---

:TRIGger:TIMIng START

:TRIGger:TIMIng?

(Response) :TRIGGER:TIMING START (when headers are enabled)

## Note

---

Depending on this setting, the input type of external input terminal 3 may be changed.

## Usage Conditions

---

Sets and queries the time interval for the interval trigger.

`:TRIGger:TMINTvl day,hour,min,sec`

## Syntax

---

### Command

`:TRIGger:TMINTvl day,hour,min,sec`

### Query

`:TRIGger:TMINTvl?`

### Response

`day<NR1>,hour<NR1>,min<NR1>,sec<NR1>`

day = 0 to 99(day)

hour = 0 to 23(hour)

min = 0 to 59(min)

sec = 0 to 59(sec)

## Details

---

Sets the time interval for the interval trigger.

Returns the current setting for the interval trigger time interval as NR1 numerical values.

## Example

---

`:TRIGger:TMINTvl 1,20,30,00`

`:TRIGger:TMINTvl?`

(Response) `:TRIGGER:TMINTVL 1,20,30,00` (when headers are enabled)

## Note

---

`:TRIGger:TMINTvl 0,0,0,0` can not be set.

## Usage Conditions

---

Sets and queries the type of trigger.

**[Analog Channel Trigger]**

:TRIGger:ANALog:STARt:KIND ch\$,A\$(Start Trigger)

:TRIGger:ANALog:STOP:KIND ch\$,A\$(Stop Trigger)

**[Waveform Calculation Trigger]**

:TRIGger:CALCulate:STARt:KIND ch\$,A\$(Start Trigger)

:TRIGger:CALCulate:STOP:KIND ch\$,A\$(Stop Trigger)

**[Pulse Channel Trigger]**

:TRIGger:PULSe:STARt:KIND ch\$,A\$(Start Trigger)

:TRIGger:PULSe:STOP:KIND ch\$,A\$(Stop Trigger)

## Syntax

---

### Command

**[Analog Channel Trigger]**

:TRIGger:ANALog:STARt:KIND ch\$,A\$(Start Trigger)

:TRIGger:ANALog:STOP:KIND ch\$,A\$(Stop Trigger)

**[Waveform Calculation Trigger]**

:TRIGger:CALCulate:STARt:KIND ch\$,A\$(Start Trigger)

:TRIGger:CALCulate:STOP:KIND ch\$,A\$(Stop Trigger)

**[Pulse Channel Trigger]**

:TRIGger:PULSe:STARt:KIND ch\$,A\$(Start Trigger)

:TRIGger:PULSe:STOP:KIND ch\$,A\$(Stop Trigger)

### Query

**[Analog Channel Trigger]**

:TRIGger:ANALog:STARt:KIND? ch\$(Start Trigger)

:TRIGger:ANALog:STOP:KIND? ch\$(Stop Trigger)

**[Waveform Calculation Trigger]**

:TRIGger:CALCulate:STARt:KIND ch\$,A\$(Start Trigger)

:TRIGger:CALCulate:STOP:KIND ch\$,A\$(Stop Trigger)

**[Pulse Channel Trigger]**

:TRIGger:PULSe:STARt:KIND ch\$,A\$(Start Trigger)

:TRIGger:PULSe:STOP:KIND ch\$,A\$(Stop Trigger)

### Response

**[Analog Channel Trigger]**

ch\$,A\$

[Waveform Calculation Trigger]

w\$,A\$

[Pulse Channel Trigger]

pls\$,A\$

ch\$ = CH1\_1 to CH10\_30

w\$ = W1 to W30

pls\$ = PLS1

A\$ = OFF, LEVEL, WINDOW

## Details

---

Sets the type of trigger for the channel designated by ch\$/w\$/pls\$.

Returns as character data the type of the current trigger for the channel designated by ch\$/w\$/pls\$.

OFF

LEVEL: Level trigger

WINDOW: Window trigger

## Example

---

```
:TRIGger:ANALog:STARt:KIND CH1_1,LEVEL
```

```
:TRIGger:ANALog:STARt:KIND? CH1_1
```

(Response) :TRIGGER:ANALOG:START:KIND CH1\_1,LEVEL (when headers are enabled)

## Note

---

It is also possible to set and query using the following conventional commands.

```
:TRIGger:KIND ch$,A$ / :TRIGger:KIND? ch$
```

```
:TRIGger:SKIND ch$,A$ / :TRIGger:SKIND? ch$
```

```
:TRIGger:WKIND w$,A$ / :TRIGger:WKIND? w$
```

```
:TRIGger:SWKIND w$,A$ / :TRIGger:SWKIND? w$
```

```
:TRIGger:PKIND pls$,A$ / :TRIGger:PKIND? pls$
```

```
:TRIGger:SPKIND pls$,A$ / :TRIGger:SPKIND? pls$
```

## Usage Conditions

---

Sets and queries trigger level of the level trigger.

**[Analog Channel Trigger]**

:TRIGger:ANALog:STARt:LEVEL ch\$,A (Start Trigger)

:TRIGger:ANALog:STOP:LEVEL ch\$,A (Stop Trigger)

**[Waveform Calculation Trigger]**

:TRIGger:CALCulate:STARt:LEVEL w\$,A (Start Trigger)

:TRIGger:CALCulate:STOP:LEVEL w\$,A (Stop Trigger)

**[Pulse Channel Trigger]**

:TRIGger:PULSe:STARt:LEVEL pls\$,A (Start Trigger)

:TRIGger:PULSe:STOP:LEVEL pls\$,A (Stop Trigger)

## Syntax

---

### Command

**[Analog Channel Trigger]**

:TRIGger:ANALog:STARt:LEVEL ch\$,A (Start Trigger)

:TRIGger:ANALog:STOP:LEVEL ch\$,A (Stop Trigger)

**[Waveform Calculation Trigger]**

:TRIGger:CALCulate:STARt:LEVEL w\$,A (Start Trigger)

:TRIGger:CALCulate:STOP:LEVEL w\$,A (Stop Trigger)

**[Pulse Channel Trigger]**

:TRIGger:PULSe:STARt:LEVEL pls\$,A (Start Trigger)

:TRIGger:PULSe:STOP:LEVEL pls\$,A (Stop Trigger)

### Query

**[Analog Channel Trigger]**

:TRIGger:ANALog:STARt:KIND? ch\$(Start Trigger)

:TRIGger:ANALog:STOP:KIND? ch\$(Stop Trigger)

**[Waveform Calculation Trigger]**

:TRIGger:CALCulate:STARt:KIND ch\$,A\$(Start Trigger)

:TRIGger:CALCulate:STOP:KIND ch\$,A\$(Stop Trigger)

**[Pulse Channel Trigger]**

:TRIGger:PULSe:STARt:KIND ch\$,A\$(Start Trigger)

:TRIGger:PULSe:STOP:KIND ch\$,A\$(Stop Trigger)

### Response

**[Analog Channel Trigger]**

ch\$,A<NR3> (3 decimal places)

[Waveform Calculation Trigger]  
w\$,A<NR3> (4 decimal places)

[Pulse Channel Trigger]  
pls\$,A<NR3> (9 decimal places)

ch\$ = CH1\_1 to CH10\_30

w\$ = W1 to W30

pls\$ = PLS1

[Analog Channel Trigger]

A = Configurable Range : (measurement range) $\times$ ( $\pm 1.5$ ), Minimum Resolution : (measurement range) $\times$ (1/1000)

[Waveform Calculation Trigger]

A = -9.9999E+29 to 9.9999E+29

[Pulse Channel Trigger]

A = 0 to 1000000000(count), 0 to 15000(r/s), 0 to 900000(r/min)

## Details

---

Sets the trigger level of the level trigger of the channel designated by ch\$/w\$/pls\$.  
Returns as an NR3 numerical value the current trigger level of the channel designated by ch\$/w\$/pls\$.

## Example

---

```
:TRIGger:ANALog:STARt:LEVEL CH1_1,0.1
```

```
:TRIGger:ANALog:STARt:LEVEL? CH1_1
```

(Response) :TRIGGER:ANALOG:START:LEVEL CH1\_1,+1.000E-01 (when headers are enabled)

## Note

---

If you enter a value greater than the configurable limit, the maximum value is set.

If you enter a value less than the configurable lower limit, the minimum value is set.

It is also possible to set and query using the following conventional commands.

```
:TRIGger:LEVEL ch$,A / :TRIGger:LEVEL? ch$
```

```
:TRIGger:SLEVEL ch$,A / :TRIGger:SLEVEL? ch$
```

```
:TRIGger:WLEVEL w$,A / :TRIGger:WLEVEL? w$
```

```
:TRIGger:SWLEVEL w$,A / :TRIGger:SWLEVEL? w$
```

```
:TRIGger:PLEVEL pls$,A / :TRIGger:PLEVEL? pls$
```

```
:TRIGger:SPLEVEL pls$,A / :TRIGger:SPLEVEL? pls$
```

## Usage Conditions

---

The type of trigger that can be set is Level Trigger.

Sets and queries the lower limit level for a window trigger.

**[Analog Channel Trigger]**

:TRIGger:ANALog:STARt:LOWEr ch\$,A (Start Trigger)

:TRIGger:ANALog:STOP:LOWEr ch\$,A (Stop Trigger)

**[Waveform Calculation Trigger]**

:TRIGger:CALCulate:STARt:LOWEr w\$,A (Start Trigger)

:TRIGger:CALCulate:STOP:LOWEr w\$,A (Stop Trigger)

**[Pulse Channel Trigger]**

:TRIGger:PULSe:STARt:LOWEr pls\$,A (Start Trigger)

:TRIGger:PULSe:STOP:LOWEr pls\$,A (Stop Trigger)

## Syntax

---

### Command

**[Analog Channel Trigger]**

:TRIGger:ANALog:STARt:LOWEr ch\$,A (Start Trigger)

:TRIGger:ANALog:STOP:LOWEr ch\$,A (Stop Trigger)

**[Waveform Calculation Trigger]**

:TRIGger:CALCulate:STARt:LOWEr w\$,A (Start Trigger)

:TRIGger:CALCulate:STOP:LOWEr w\$,A (Stop Trigger)

**[Pulse Channel Trigger]**

:TRIGger:PULSe:STARt:LOWEr pls\$,A (Start Trigger)

:TRIGger:PULSe:STOP:LOWEr pls\$,A (Stop Trigger)

### Query

**[Analog Channel Trigger]**

:TRIGger:ANALog:STARt:LOWEr? ch\$ (Start Trigger)

:TRIGger:ANALog:STOP:LOWEr? ch\$ (Stop Trigger)

**[Waveform Calculation Trigger]**

:TRIGger:CALCulate:STARt:LOWEr? w\$ (Start Trigger)

:TRIGger:CALCulate:STOP:LOWEr? w\$ (Stop Trigger)

**[Pulse Channel Trigger]**

:TRIGger:PULSe:STARt:LOWEr? pls\$ (Start Trigger)

:TRIGger:PULSe:STOP:LOWEr? pls\$ (Stop Trigger)

### Response

**[Analog Channel Trigger]**

ch\$,A<NR3> (3 decimal places)

[Waveform Calculation Trigger]  
w\$,A<NR3> (4 decimal places)

[Pulse Channel Trigger]  
pls\$,A<NR3> (9 decimal places)

ch\$ = CH1\_1 to CH10\_30

w\$ = W1 to W30

pls\$ = PLS1

[Analog Channel Trigger]

A = Configurable Range : (measurement range) $\times$ ( $\pm 1.5$ ), Minimum Resolution : (measurement range) $\times$ (1/1000)

[Waveform Calculation Trigger]

A = -9.9999E+29 to 9.9999E+29

[Pulse Channel Trigger]

A = 0 to 1000000000(count), 0 to 15000(r/s), 0 to 900000(r/min)

## Details

---

Sets the lower limit level of the window trigger of the channel designated by ch\$/w\$/pls\$ as a voltage value.

Returns the current lower limit level of the window trigger as an NR3 numerical value of the channel designated by ch\$/w\$/pls\$.

## Example

---

```
:TRIGger:ANALog:STARt:LOWEr CH1_1,-0.5
```

```
:TRIGger:ANALog:STARt:LOWEr? CH1_1
```

```
(Response) :TRIGGER:ANALOG:STARt:LOWEr CH1_1,-5.000E-01 (when headers are enabled)
```

## Note

---

You cannot enter a value greater than or equal to the upper limit level of the window trigger.

If you enter a value greater than the configurable limit, the maximum value is set.

If you enter a value less than the configurable lower limit, the minimum value is set.

It is also possible to set and query using the following conventional commands.

```
:TRIGger:LOWEr ch$,A / :TRIGger:LOWEr? ch$
```

```
:TRIGger:SLOWEr ch$,A / :TRIGger:SLOWEr? ch$
```

```
:TRIGger:WLOWEr w$,A / :TRIGger:WLOWEr? w$
```

```
:TRIGger:SWLOWEr w$,A / :TRIGger:SWLOWEr? w$
```

```
:TRIGger:PLowEr pls$,A / :TRIGger:PLowEr? pls$
```

```
:TRIGger:SPLOWEr pls$,A / :TRIGger:SPLOWEr? pls$
```

## Usage Conditions

---

The type of trigger that can be set is Window Trigger.

Sets and queries the window trigger direction (side).

**[Analog Channel Trigger]**

:TRIGger:ANALog:STARt:SIDE ch\$,A\$ (Start Trigger)

:TRIGger:ANALog:STOP:SIDE ch\$,A\$ (Stop Trigger)

**[Waveform Calculation Trigger]**

:TRIGger:CALCulate:STARt:SIDE w\$,A\$ (Start Trigger)

:TRIGger:CALCulate:STOP:SIDE w\$,A\$ (Stop Trigger)

**[Pulse Channel Trigger]**

:TRIGger:PULSe:STARt:SIDE pls\$,A\$ (Start Trigger)

:TRIGger:PULSe:STOP:SIDE pls\$,A\$ (Stop Trigger)

## Syntax

---

### Command

**[Analog Channel Trigger]**

:TRIGger:ANALog:STARt:SIDE ch\$,A\$ (Start Trigger)

:TRIGger:ANALog:STOP:SIDE ch\$,A\$ (Stop Trigger)

**[Waveform Calculation Trigger]**

:TRIGger:CALCulate:STARt:SIDE w\$,A\$ (Start Trigger)

:TRIGger:CALCulate:STOP:SIDE w\$,A\$ (Stop Trigger)

**[Pulse Channel Trigger]**

:TRIGger:PULSe:STARt:SIDE pls\$,A\$ (Start Trigger)

:TRIGger:PULSe:STOP:SIDE pls\$,A\$ (Stop Trigger)

### Query

**[Analog Channel Trigger]**

:TRIGger:ANALog:STARt:SIDE? ch\$ (Start Trigger)

:TRIGger:ANALog:STOP:SIDE? ch\$ (Stop Trigger)

**[Waveform Calculation Trigger]**

:TRIGger:CALCulate:STARt:SIDE? w\$ (Start Trigger)

:TRIGger:CALCulate:STOP:SIDE? w\$ (Stop Trigger)

**[Pulse Channel Trigger]**

:TRIGger:PULSe:STARt:SIDE? pls\$ (Start Trigger)

:TRIGger:PULSe:STOP:SIDE? pls\$ (Stop Trigger)

### Response

**[Analog Channel Trigger]**

ch\$,A\$

[Waveform Calculation Trigger]

w\$,A\$

[Pulse Channel Trigger]

pls\$,A\$

ch\$ = CH1\_1 to CH10\_30

w\$ = W1 to W30

pls\$ = PLS1

A\$ = IN, OUT

## Details

---

Sets the trigger direction of the window trigger of the channel designated by ch\$/w\$/pls\$.

Returns as a character value the current window trigger direction of the channel designated by ch\$/w\$/pls\$.

IN: Window in

OUT: Window out

## Example

---

```
:TRIGger:ANALog:STARt:SIDE CH1_1,IN
```

```
:TRIGger:ANALog:STARt:SIDE? CH1_1
```

(Response) :TRIGGER:ANALOG:START:SIDE CH1\_1,IN (when headers are enabled)

## Note

---

It is also possible to set and query using the following conventional commands.

```
:TRIGger:SIDE ch$,A$ / :TRIGger:SIDE? ch$
```

```
:TRIGger:SSIDE ch$,A$ / :TRIGger:SSIDE? ch$
```

```
:TRIGger:WSIDE w$,A$ / :TRIGger:WSIDE? w$
```

```
:TRIGger:SWSIDE w$,A$ / :TRIGger:SWSIDE? w$
```

```
:TRIGger:PSIDE pls$,A$ / :TRIGger:PSIDE? pls$
```

```
:TRIGger:SPSIDE pls$,A$ / :TRIGger:SPSIDE? pls$
```

## Usage Conditions

---

The type of trigger that can be set is Window Trigger.

Sets and queries the type of trigger.

**[Analog Channel Trigger]**

:TRIGger:ANALog:STARt:SLOPe ch\$,A\$ (Start Trigger)

:TRIGger:ANALog:STOP:SLOPe ch\$,A\$ (Stop Trigger)

**[Waveform Calculation Trigger]**

:TRIGger:CALCulate:STARt:SLOPe w\$,A\$ (Start Trigger)

:TRIGger:CALCulate:STOP:SLOPe w\$,A\$ (Stop Trigger)

**[Pulse Channel Trigger]**

:TRIGger:PULSe:STARt:SLOPe pls\$,A\$ (Start Trigger)

:TRIGger:PULSe:STOP:SLOPe pls\$,A\$ (Stop Trigger)

## Syntax

---

### Command

**[Analog Channel Trigger]**

:TRIGger:ANALog:STARt:SLOPe ch\$,A\$ (Start Trigger)

:TRIGger:ANALog:STOP:SLOPe ch\$,A\$ (Stop Trigger)

**[Waveform Calculation Trigger]**

:TRIGger:CALCulate:STARt:SLOPe w\$,A\$ (Start Trigger)

:TRIGger:CALCulate:STOP:SLOPe w\$,A\$ (Stop Trigger)

**[Pulse Channel Trigger]**

:TRIGger:PULSe:STARt:SLOPe pls\$,A\$ (Start Trigger)

:TRIGger:PULSe:STOP:SLOPe pls\$,A\$ (Stop Trigger)

### Query

**[Analog Channel Trigger]**

:TRIGger:ANALog:STARt:SLOPe? ch\$ (Start Trigger)

:TRIGger:ANALog:STOP:SLOPe? ch\$ (Stop Trigger)

**[Waveform Calculation Trigger]**

:TRIGger:CALCulate:STARt:SLOPe? w\$ (Start Trigger)

:TRIGger:CALCulate:STOP:SLOPe? w\$ (Stop Trigger)

**[Pulse Channel Trigger]**

:TRIGger:PULSe:STARt:SLOPe? pls\$ (Start Trigger)

:TRIGger:PULSe:STOP:SLOPe? pls\$ (Stop Trigger)

### Response

**[Analog Channel Trigger]**

ch\$,A\$

[Waveform Calculation Trigger]

w\$,A\$

[Pulse Channel Trigger]

pls\$,A\$

ch\$ = CH1\_1 to CH10\_30

w\$ = W1 to W30

pls\$ = PLS1

A\$ = UP, DOWN

## Details

---

Sets the trigger direction of the level trigger of the channel designated by ch\$/w\$/pls\$.

Returns as a character value the current level trigger direction of the channel designated by ch\$/w\$/pls\$.

UP: Rising

DOWN: Falling

## Example

---

```
:TRIGger:ANALog:STARt:SLOPe CH1_1,UP
```

```
:TRIGger:ANALog:STARt:SLOPe? CH1_1
```

(Response) :TRIGGER:ANALOG:START:SLOPE CH1\_1,UP (when headers are enabled)

## Note

---

It is also possible to set and query using the following conventional commands.

```
:TRIGger:SLOPe ch$,A$ / :TRIGger:SLOPe? ch$
```

```
:TRIGger:SSLOPe ch$,A$ / :TRIGger:SSLOPe? ch$
```

```
:TRIGger:WSLOPe w$,A$ / :TRIGger:WSLOPe? w$
```

```
:TRIGger:SWSLOPe w$,A$ / :TRIGger:SWSLOPe? w$
```

```
:TRIGger:PSLOPe pls$,A$ / :TRIGger:PSLOPe? pls$
```

```
:TRIGger:SPSLOPe pls$,A$ / :TRIGger:SPSLOPe? pls$
```

## Usage Conditions

---

The type of trigger that can be set is Level Trigger.

Sets and queries the upper limit level for a window trigger.

**[Analog Channel Trigger]**

:TRIGger:ANALog:STARt:UPPEr ch\$,A (Start Trigger)

:TRIGger:ANALog:STOP:UPPEr ch\$,A (Stop Trigger)

**[Waveform Calculation Trigger]**

:TRIGger:CALCulate:STARt:UPPEr w\$,A (Start Trigger)

:TRIGger:CALCulate:STOP:UPPEr w\$,A (Stop Trigger)

**[Pulse Channel Trigger]**

:TRIGger:PULSe:STARt:UPPEr pls\$,A (Start Trigger)

:TRIGger:PULSe:STOP:UPPEr pls\$,A (Stop Trigger)

## Syntax

---

### Command

**[Analog Channel Trigger]**

:TRIGger:ANALog:STARt:UPPEr ch\$,A (Start Trigger)

:TRIGger:ANALog:STOP:UPPEr ch\$,A (Stop Trigger)

**[Waveform Calculation Trigger]**

:TRIGger:CALCulate:STARt:UPPEr w\$,A (Start Trigger)

:TRIGger:CALCulate:STOP:UPPEr w\$,A (Stop Trigger)

**[Pulse Channel Trigger]**

:TRIGger:PULSe:STARt:UPPEr pls\$,A (Start Trigger)

:TRIGger:PULSe:STOP:UPPEr pls\$,A (Stop Trigger)

### Query

**[Analog Channel Trigger]**

:TRIGger:ANALog:STARt:UPPEr? ch\$ (Start Trigger)

:TRIGger:ANALog:STOP:UPPEr? ch\$ (Stop Trigger)

**[Waveform Calculation Trigger]**

:TRIGger:CALCulate:STARt:UPPEr? w\$ (Start Trigger)

:TRIGger:CALCulate:STOP:UPPEr? w\$ (Stop Trigger)

**[Pulse Channel Trigger]**

:TRIGger:PULSe:STARt:UPPEr? pls\$ (Start Trigger)

:TRIGger:PULSe:STOP:UPPEr? pls\$ (Stop Trigger)

### Response

**[Analog Channel Trigger]**

ch\$,A<NR3> (3 decimal places)

[Waveform Calculation Trigger]  
w\$,A<NR3> (4 decimal places)

[Pulse Channel Trigger]  
pls\$,A<NR3> (9 decimal places)

ch\$ = CH1\_1 to CH10\_30

w\$ = W1 to W30

pls\$ = PLS1

[Analog Channel Trigger]

A = Configurable Range : (measurement range) $\times$ ( $\pm 1.5$ ), Minimum Resolution : (measurement range) $\times$ (1/1000)

[Waveform Calculation Trigger]

A = -9.9999E+29 to 9.9999E+29

[Pulse Channel Trigger]

A = 0 to 1000000000(count), 0 to 15000(r/s), 0 to 900000(r/min)

## Details

---

Sets the upper limit level of the window trigger of the channel designated by ch\$/w\$/pls\$ as a voltage value.

Returns the current upper limit level of the window trigger as an NR3 numerical value of the channel designated by ch\$/w\$/pls\$.

## Example

---

```
:TRIGger:ANALog:STARt:UPPEr CH1_1,0.5
```

```
:TRIGger:ANALog:STARt:UPPEr? CH1_1
```

(Response) :TRIGGER:ANALOG:STARt:UPPER CH1\_1,+5.000E-01 (when headers are enabled)

## Note

---

You cannot enter a value less than or equal to the lower limit level of the window trigger.

If you enter a value greater than the configurable limit, the maximum value is set.

If you enter a value less than the configurable lower limit, the minimum value is set.

It is also possible to set and query using the following conventional commands.

```
:TRIGger:UPPEr ch$,A / :TRIGger:UPPEr? ch$
```

```
:TRIGger:SUPPEr ch$,A / :TRIGger:SUPPEr? ch$
```

```
:TRIGger:WUPPEr w$,A / :TRIGger:WUPPEr? w$
```

```
:TRIGger:SWUPPEr w$,A / :TRIGger:SWUPPEr? w$
```

```
:TRIGger:PUPPEr pls$,A / :TRIGger:PUPPEr? pls$
```

```
:TRIGger:SPUPPEr pls$,A / :TRIGger:SPUPPEr? pls$
```

## Usage Conditions

---

The type of trigger that can be set is Window Trigger.

Sets and queries the trigger pattern for a logic trigger.

`:TRIGger:LOGic:START:PATtern "A$" (Start Trigger)`  
`:TRIGger:LOGic:STOP:PATtern "A$" (Stop Trigger)`

## Syntax

---

### Command

`:TRIGger:LOGic:START:PATtern "A$" (Start Trigger)`  
`:TRIGger:LOGic:STOP:PATtern "A$" (Stop Trigger)`

### Query

`:TRIGger:LOGic:START:PATtern? (Start Trigger)`  
`:TRIGger:LOGic:STOP:PATtern? (Stop Trigger)`

### Response

"A\$"  
A\$ = X, 0, 1

## Details

---

Sets the trigger pattern for the logic trigger.

Returns the setting for the trigger pattern for the logic trigger as that specified by the given character data.

X: Ignore signal  
0: Trigger at low level  
1: Trigger at high level

## Example

---

`:TRIGger:LOGic:START:PATtern "1"`  
`:TRIGger:LOGic:START:PATtern?`  
(Response) `:TRIGGER:LOGIC:START:PATTERN "1"` (when headers are enabled)

## Note

---

It is also possible to set and query using the following conventional commands.

`:TRIGger:LOGPat "A$" / :TRIGger:LOGPat?`  
`:TRIGger:SLOGPat "A$" / :TRIGger:SLOGPat?`

## Usage Conditions

---

Sets and queries the external trigger.

:TRIGger:EXTernal:STARt:KIND A\$ (Start Trigger)  
:TRIGger:EXTernal:STOP:KIND A\$ (Stop Trigger)

## Syntax

---

### Command

:TRIGger:EXTernal:STARt:KIND A\$ (Start Trigger)  
:TRIGger:EXTernal:STOP:KIND A\$ (Stop Trigger)

### Query

:TRIGger:EXTernal:STARt:KIND? (Start Trigger)  
:TRIGger:EXTernal:STOP:KIND? (Stop Trigger)

### Response

A\$  
A\$ = OFF, ON

## Details

---

Sets the external trigger.  
Returns the current external trigger setting as character data.  
OFF: Disabled  
ON: Enabled

## Example

---

:TRIGger:EXTernal:STARt:KIND ON  
:TRIGger:EXTernal:STARt:KIND?  
(Response) :TRIGGER:EXTERNAL:STARt:KIND ON (when headers are enabled)

## Note

---

Depending on this setting, the input type of external input terminal 3 may be changed.

## Usage Conditions

---

## 3.9 Alarm(ALARm)

Sets and queries the alarm output terminal.

```
:ALARm:ACTive alm$,A$
```

### Syntax

---

#### Command

```
:ALARm:ACTive alm$,A$
```

#### Query

```
:ALARm:ACTive? alm$
```

#### Response

```
alm$,A$
```

alm\$ = ALM1 to ALM4

A\$ = LOW, HIGH

LOW: ActiveLOW

HIGH: ActiveHIGH

### Details

---

Sets the active low/high of the alarm output terminal.

Returns the current active low/high setting of the alarm output terminal as character data.

### Example

---

```
:ALARm:ACTive ALM1,LOW
```

```
:ALARm:ACTive? ALM1
```

(Response) :ALARM:ACTIVE ALM1,LOW (when headers are enabled)

### Note

---

### Usage Conditions

---

Queries the alarm history details.

`:ALARm:ARCD? NO`

## Syntax

---

### Query

`:ALARm:ARCD? NO`

### Response

`NO<NR1>,.ALM$,CH$,ERR$,STR$,ENDS$`

NO = Alarm history num (1 to 999999)

ALM\$ = ALM1 to ALM4

CH\$ = CH1\_1 to CH10\_30, PLS1, LOG, W1 to W30

ERR\$ = -, BURN\_OUT (Thermocouple disconnection)

STR\$ = Alarm start time

END\$ = Alarm stop time

## Details

---

Returns the alarm history details set by NO.

## Example

---

`:ALARm:ARCD? 1`

(Response) `:ALARm:ARCD 1,ALM1,CH1_1,-, 20ms, 60ms` (when headers are enabled)

## Note

---

The format of the alarm start/stop time is based on the time axis display setting.

If the alarm is not stopped, the stop time will be "-".

Channel number for thermocouple burn out is "-".

If the alarm history num is set to the 100 of latest, only 100 records up to the number of alarm history are valid.

## Usage Conditions

---

The alarm history num is greater than or equal to 1.

Queries the alarm history num.

`:ALARm:ARCDNum?`

## Syntax

---

### Query

`:ALARm:ARCDNum?`

### Response

A<NR1>

A = 0 to 999999 (0 = No alarm)

## Details

---

Returns the alarm history num as an NR1 numerical value.

## Example

---

`:ALARm:ARCDNum?`

(Response) `:ALARM:ARCDNUM 10` (when headers are enabled)

## Note

---

## Usage Conditions

---

Sets and queries the alarm beep.

`:ALARm:BEEP A$`

## Syntax

---

### Command

`:ALARm:BEEP A$`

### Query

`:ALARm:BEEP?`

### Response

A\$

A\$ = OFF, ON

## Details

---

Sets the alarm beep.

Returns the current alarm beep as character data.

## Example

---

`:ALARm:BEEP ON`

`:ALARm:BEEP?`

(Response) `:ALARM:BEEP ON` (when headers are enabled)

## Note

---

## Usage Conditions

---

Sets and queries the alarm at burn out.

`:ALARm:BURN alm$,A$`

## Syntax

---

### Command

`:ALARm:BURN alm$,A$`

### Query

`:ALARm:BURN? alm$`

### Response

`alm$,A$`

`alm$ = ALM1 to ALM4`

`A$ = OFF, ON`

## Details

---

Sets the alarm at burn out.

Returns the current alarm at burn out as character data.

## Example

---

`:ALARm:BURN ALM1,ON`

`:ALARm:BURN? ALM1`

(Response) `:ALARm:BURN ALM1,ON` (when headers are enabled)

## Note

---

## Usage Conditions

---

Sets and queries the alarm filter.

```
:ALARm:FILTer alm$,A
```

## Syntax

---

### Command

```
:ALARm:FILTer alm$,A
```

### Query

```
:ALARm:FILTer? alm$
```

### Response

```
alm$,A<NR1>
```

alm\$ = ALM1 to ALM4

A = 0(OFF), 2 to 1000

## Details

---

Sets the alarm filter.

Returns the current alarm filter setting as an NR1 numerical value.

## Example

---

```
:ALARm:FILTer ALM1,2
```

```
:ALARm:FILTer? ALM1
```

(Response) :ALARM:FILTER ALM1,2 (when headers are enabled)

## Note

---

## Usage Conditions

---

sets and queries the alarm history recording

**:ALARm:HISTory A\$**

## Syntax

---

### Command

:ALARm:HISTory A\$

### Query

:ALARm:HISTory?

### Response

A\$

A\$ = OFF, ON

## Details

---

Sets the alarm history recording.

Returns the current alarm history recording settings.

OFF: 100 from start

ON: 100 of latest

## Example

---

:ALARm:HISTory ON

:ALARm:HISTory?

(Response) :ALARM:HISTORY ON (when headers are enabled)

## Note

---

The alarm history recording settings cannot be changed during the measurement.

## Usage Conditions

---

Sets and queries the alarm hold.

**:ALARm:HOLD A\$**

## Syntax

---

### Command

:ALARm:HOLD A\$

### Query

:ALARm:HOLD?

### Response

A\$

A\$ = OFF, ON, CLEAR

## Details

---

Sets the alarm hold.

Returns the current alarm hold as character data.

If you specify CLEAR for A\$, the alarm is cleared without stopping the measurement.

## Example

---

:ALARm:HOLD ON

:ALARm:HOLD?

(Response) :ALARM:HOLD ON (when headers are enabled)

## Note

---

## Usage Conditions

---

A\$ = CLEAR can only be used when alarm hold is ON.

Sets and queries the alarm source AND/OR.

`:ALARm:SOURce alm$,A$`

## Syntax

---

### Command

`:ALARm:SOURce alm$,A$`

### Query

`:ALARm:SOURce? alm$`

### Response

`alm$,A$`

`alm$ = ALM1 to ALM4`

`A$ = OR, AND`

## Details

---

Sets the alarm source AND/OR.

Returns the current alarm source AND/OR as character data.

AND: Logical product

OR: Logical sum

## Example

---

`:ALARm:SOURce ALM1,AND`

`:ALARm:SOURce? ALM1`

(Response) `ALARM:SOURCE ALM1,AND` (when headers are enabled)

## Note

---

## Usage Conditions

---

Sets and queries the type of alarm.

[Analog Channel Alarm]

:ALARm:ANALog:KIND alm\$,ch\$,A\$

[Waveform Calculation Alarm]

:ALARm:CALCulate:KIND alm\$,w\$,A\$

[Pulse Channel Alarm]

:ALARm:PULSe:KIND alm\$,pls\$,A\$

## Syntax

---

### Command

[Analog Channel Alarm]

:ALARm:ANALog:KIND alm\$,ch\$,A\$

[Waveform Calculation Alarm]

:ALARm:CALCulate:KIND alm\$,w\$,A\$

[Pulse Channel Alarm]

:ALARm:PULSe:KIND alm\$,pls\$,A\$

### Query

[Analog Channel Alarm]

:ALARm:ANALog:KIND? alm\$,ch\$

[Waveform Calculation Alarm]

:ALARm:CALCulate:KIND? alm\$,w\$

[Pulse Channel Alarm]

:ALARm:PULSe:KIND? alm\$,pls\$

### Response

[Analog Channel Alarm]

alm\$,ch\$,A\$

[Waveform Calculation Alarm]

alm\$,w\$,A\$

[Pulse Channel Alarm]

alm\$,pls\$,A\$

alm\$ = ALM1 to ALM4

ch\$ = CH1\_1 to CH10\_30

w\$ = W1 to W30

pls\$ = PLS1

A\$ = OFF, LEVEL, WINDow, SLOPe, SLOPE2

## Details

---

Sets the type of alarm for the channel designated by alm\$,ch\$/w\$/pls\$.

Returns as character data the type of the current alarm for the channel designated by alm\$,ch\$/w\$/pls\$.

OFF

LEVEL: Level alarm

WINDow: Window alarm

SLOPe: Slope alarm

SLOPE2: Variation

## Example

---

```
:ALARm:ANALog:KIND ALM1,CH1_1,LEVEL
```

```
:ALARm:ANALog:KIND? ALM1,CH1_1
```

(Response) :ALARM:ANALOG:KIND ALM1,CH1\_1,LEVEL (when headers are enabled)

## Note

---

It is also possible to set and query using the following conventional commands.

```
:ALARm:KIND alm$,ch$,A$ / :ALARm:KIND? alm$,ch$
```

```
:ALARm:WKIND alm$,w$,A$ / :ALARm:WKIND? alm$,w$
```

```
:ALARm:PKIND alm$,pls$,A$ / :ALARm:PKIND? alm$,pls$
```

## Usage Conditions

---

Sets and queries the level of the level alarm.

**[Analog Channel Alarm]**

:ALARm:ANALog:LEVEL alm\$,ch\$,A

**[Waveform Calculation Alarm]**

:ALARm:CALCulate:LEVEL alm\$,w\$,A

**[Pulse Channel Alarm]**

:ALARm:PULSe:LEVEL alm\$,pls\$,A

## Syntax

---

### Command

**[Analog Channel Alarm]**

:ALARm:ANALog:LEVEL alm\$,ch\$,A

**[Waveform Calculation Alarm]**

:ALARm:CALCulate:LEVEL alm\$,w\$,A

**[Pulse Channel Alarm]**

:ALARm:PULSe:LEVEL alm\$,pls\$,A

### Query

**[Analog Channel Alarm]**

:ALARm:ANALog:LEVEL? alm\$,ch\$

**[Waveform Calculation Alarm]**

:ALARm:CALCulate:LEVEL? alm\$,w\$

**[Pulse Channel Alarm]**

:ALARm:PULSe:LEVEL? alm\$,pls\$

### Response

**[Analog Channel Alarm]**

alm\$,ch\$,A<NR3> (3 decimal places)

**[Waveform Calculation Alarm]**

alm\$,w\$,A<NR3> (4 decimal places)

**[Pulse Channel Alarm]**

alm\$,pls\$,A<NR3> (9 decimal places)

alm\$ = ALM1 to ALM4

ch\$ = CH1\_1 to CH10\_30

w\$ = W1 to W30

pls\$ = PLS1

[Analog Channel Alarm]

A = Configurable Range : (measurement range) $\times$ ( $\pm 1.5$ ), Minimum Resolution : (measurement range) $\times$ (1/1000)

[Waveform Calculation Alarm]

A = -9.9999E+29 to 9.9999E+29

[Pulse Channel Alarm]

A = 0 to 1000000000(count), 0 to 15000(r/s), 0 to 900000(r/min)

## Details

---

Sets the alarm level of the level alarm of the channel designated by alm\$,ch\$/w\$/pls\$.

Returns as an NR3 numerical value the current alarm level of the channel designated by alm\$,ch\$/w\$/pls\$.

## Example

---

```
:ALARm:ANALog:LEVEL ALM1,CH1_1,0.1
```

```
:ALARm:ANALog:LEVEL? ALM1,CH1_1
```

(Response) :ALARM:ANALOG:LEVEL ALM1,CH1\_1,+1.000E-01 (when headers are enabled)

## Note

---

If you enter a value greater than the configurable limit, the maximum value is set.

If you enter a value less than the configurable lower limit, the minimum value is set.

It is also possible to set and query using the following conventional commands.

```
:ALARm:LEVEL alm$,ch$,A$ / :ALARm:LEVEL? alm$,ch$
```

```
:ALARm:WLEVEL alm$,w$,A / :ALARm:WLEVEL? alm$,w$
```

```
:ALARm:PLEVEL alm$,pls$,A / :ALARm:PLEVEL? alm$,pls$
```

## Usage Conditions

---

Sets and queries the lower limit level for a window-in/-out alarm.

**[Analog Channel Alarm]**

:ALARm:ANALog:LOWEr alm\$,ch\$,A

**[Waveform Calculation Alarm]**

:ALARm:CALCulate:LOWEr alm\$,w\$,A

**[Pulse Channel Alarm]**

:ALARm:PULSe:LOWEr alm\$,pls\$,A

## Syntax

---

### Command

**[Analog Channel Alarm]**

:ALARm:ANALog:LOWEr alm\$,ch\$,A

**[Waveform Calculation Alarm]**

:ALARm:CALCulate:LOWEr alm\$,w\$,A

**[Pulse Channel Alarm]**

:ALARm:PULSe:LOWEr alm\$,pls\$,A

### Query

**[Analog Channel Alarm]**

:ALARm:ANALog:LOWEr? alm\$,ch\$

**[Waveform Calculation Alarm]**

:ALARm:CALCulate:LOWEr? alm\$,w\$

**[Pulse Channel Alarm]**

:ALARm:PULSe:LOWEr? alm\$,pls\$

### Response

**[Analog Channel Alarm]**

alm\$,ch\$,A<NR3> (3 decimal places)

**[Waveform Calculation Alarm]**

alm\$,w\$,A<NR3> (4 decimal places)

**[Pulse Channel Alarm]**

alm\$,pls\$,A<NR3> (9 decimal places)

alm\$ = ALM1 to ALM4

ch\$ = CH1\_1 to CH10\_30

w\$ = W1 to W30

pls\$ = PLS1

[Analog Channel Alarm]

A = Configurable Range : (measurement range) $\times$ ( $\pm 1.5$ ), Minimum Resolution : (measurement range) $\times$ (1/1000)

[Waveform Calculation Alarm]

A = -9.9999E+29 to 9.9999E+29

[Pulse Channel Alarm]

A = 0 to 1000000000(count), 0 to 15000(r/s), 0 to 900000(r/min)

## Details

---

Sets the lower limit level of the window alarm of the channel designated by alm\$,ch\$/w\$/pls\$ as a voltage value.

Returns the current lower limit level of the window alarm as an NR3 numerical value.

## Example

---

```
:ALARm:ANALog:LOWEr ALM1,CH1_1,-0.5
```

```
:ALARm:ANALog:LOWEr? ALM1,CH1_1
```

(Response) :ALARM:ANALOG:LOWER ALM1,CH1\_1,-5.000E-01 (when headers are enabled)

## Note

---

You cannot enter a value greater than or equal to the upper limit level of the window alarm.

If you enter a value greater than the configurable limit, the maximum value is set.

If you enter a value less than the configurable lower limit, the minimum value is set.

It is also possible to set and query using the following conventional commands.

```
:ALARm:LOWEr alm$,ch$,A$ / :ALARm:LOWEr? alm$,ch$
```

```
:ALARm:WLOWEr alm$,w$,A / :ALARm:WLOWEr? alm$,w$
```

```
:ALARm:PLOWEr alm$,pls$,A / :ALARm:PLOWEr? alm$,pls$
```

## Usage Conditions

---

Sets and queries the window alarm direction (side).

[Analog Channel Alarm]

:ALARm:ANALog:SIDE alm\$,ch\$,A\$

[Waveform Calculation Alarm]

:ALARm:CALCulate:SIDE alm\$,w\$,A\$

[Pulse Channel Alarm]

:ALARm:PULSe:SIDE alm\$,pls\$,A\$

## Syntax

---

### Command

[Analog Channel Alarm]

:ALARm:ANALog:SIDE alm\$,ch\$,A\$

[Waveform Calculation Alarm]

:ALARm:CALCulate:SIDE alm\$,w\$,A\$

[Pulse Channel Alarm]

:ALARm:PULSe:SIDE alm\$,pls\$,A\$

### Query

[Analog Channel Alarm]

:ALARm:ANALog:SIDE? alm\$,ch\$

[Waveform Calculation Alarm]

:ALARm:CALCulate:SIDE? alm\$,w\$

[Pulse Channel Alarm]

:ALARm:PULSe:SIDE? alm\$,pls\$

### Response

[Analog Channel Alarm]

alm\$,ch\$,A\$

[Waveform Calculation Alarm]

alm\$,w\$,A\$

[Pulse Channel Alarm]

alm\$,pls\$,A\$

alm\$ = ALM1 to ALM4

ch\$ = CH1\_1 to CH10\_30

w\$ = W1 to W30

pls\$ = PLS1  
A\$ = IN, OUT

## Details

---

Sets the alarm direction of the window alarm of the channel designated by alm\$,ch\$/w\$/pls\$.  
Returns as a character value the current window alarm direction of the channel designated by alm\$,ch\$/w\$/pls\$.

IN: Window-in

OUT: Window-out

## Example

---

```
:ALARm:ANALog:SIDE ALM1,CH1_1,IN
```

```
:ALARm:ANALog:SIDE? ALM1,CH1_1
```

(Response) :ALARM:ANALOG:SIDE ALM1,CH1\_1,IN (when headers are enabled)

## Note

---

It is also possible to set and query using the following conventional commands.

```
:ALARm:SIDE alm$,ch$,A$ / :ALARm:SIDE? alm$,ch$
```

```
:ALARm:WSIDE alm$,w$,A$ / :ALARm:WSIDE? alm$,w$
```

```
:ALARm:PSIDE alm$,pls$,A$ / :ALARm:PSIDE? alm$,pls$
```

## Usage Conditions

---

Sets and queries the level alarm direction (slope).

[Analog Channel Alarm]

:ALARm:ANALog:SLOPe alm\$,ch\$,A\$

[Waveform Calculation Alarm]

:ALARm:CALCulate:SLOPe alm\$,w\$,A\$

[Pulse Channel Alarm]

:ALARm:PULSe:SLOPe alm\$,pls\$,A\$

## Syntax

---

### Command

[Analog Channel Alarm]

:ALARm:ANALog:SLOPe alm\$,ch\$,A\$

[Waveform Calculation Alarm]

:ALARm:CALCulate:SLOPe alm\$,w\$,A\$

[Pulse Channel Alarm]

:ALARm:PULSe:SLOPe alm\$,pls\$,A\$

### Query

[Analog Channel Alarm]

:ALARm:ANALog:SLOPe? alm\$,ch\$

[Waveform Calculation Alarm]

:ALARm:CALCulate:SLOPe? alm\$,w\$

[Pulse Channel Alarm]

:ALARm:PULSe:SLOPe? alm\$,pls\$

### Response

[Analog Channel Alarm]

alm\$,ch\$,A\$

[Waveform Calculation Alarm]

alm\$,w\$,A\$

[Pulse Channel Alarm]

alm\$,pls\$,A\$

alm\$ = ALM1 to ALM4

ch\$ = CH1\_1 to CH10\_30

w\$ = W1 to W30

pls\$ = PLS1  
A\$ = HIGH, LOW

## Details

---

Sets the alarm direction of the level alarm of the channel designated by alm\$,ch\$/w\$/pls\$.  
Returns as a character value the current level alarm direction of the channel designated by alm\$,ch\$/w\$/pls\$.

HIGH: High

LOW: Low

## Example

---

```
:ALARm:ANALog:SLOPe ALM1,CH1_1,HIGH
```

```
:ALARm:ANALog:SLOPe? ALM1,CH1_1
```

(Response) :ALARM:ANALOG:SLOPE ALM1,CH1\_1,HIGH (when headers are enabled)

## Note

---

It is also possible to set and query using the following conventional commands.

```
:ALARm:SLOPe alm$,ch$,A$ / :ALARm:SLOPe? alm$,ch$
```

```
:ALARm:WSLOPe alm$,w$,A$ / :ALARm:WSLOPe? alm$,w$
```

```
:ALARm:PSLOPe alm$,pls$,A$ / :ALARm:PSLOPe? alm$,pls$
```

## Usage Conditions

---

Sets and queries the time of alarm change amount.

**[Analog Channel Alarm]**

:ALARm:ANALog:SLP2:TIME alm\$,ch\$,hour,min,sec,ms

**[Waveform Calculation Alarm]**

ALARm:CALCulate:SLP2:TIME alm\$,w\$,hour,min,sec,ms

**[Pulse Channel Alarm]**

ALARm:PULSe:SLP2:TIME alm\$,pls\$,hour,min,sec,ms

## Syntax

---

### Command

**[Analog Channel Alarm]**

:ALARm:ANALog:SLP2:TIME alm\$,ch\$,hour,min,sec,ms

**[Waveform Calculation Alarm]**

ALARm:CALCulate:SLP2:TIME alm\$,w\$,hour,min,sec,ms

**[Pulse Channel Alarm]**

ALARm:PULSe:SLP2:TIME alm\$,pls\$,hour,min,sec,ms

### Query

**[Analog Channel Alarm]**

:ALARm:ANALog:SLP2:TIME? alm\$,ch\$

**[Waveform Calculation Alarm]**

ALARm:CALCulate:SLP2:TIME? alm\$,w\$

**[Pulse Channel Alarm]**

ALARm:PULSe:SLP2:TIME? alm\$,pls\$

### Response

**[Analog Channel Alarm]**

alm\$,ch\$,hour<NR1>,min<NR1>,sec<NR1>,ms<NR1>

**[Waveform Calculation Alarm]**

alm\$,w\$,hour<NR1>,min<NR1>,sec<NR1>,ms<NR1>

**[Pulse Channel Alarm]**

alm\$,pls\$,hour<NR1>,min<NR1>,sec<NR1>,ms<NR1>

alm\$ = ALM1 to ALM4

ch\$ = CH1\_1 to CH10\_30

w\$ = W1 to W30

pls\$ = PLS1

hour = 0 to 23 (hour)  
min = 0 to 59 (min)  
sec = 0 to 59 (sec)  
ms = 0 to 999 (millisec)

## Details

---

Sets the time of alarm change amount of the channel designated by alm\$,ch\$/w\$/pls\$.  
Returns as an NR1 numerical value the current time of alarm change amount of the channel designated by alm\$,ch\$/w\$/pls\$.

## Example

---

```
:ALARm:ANALog:SLP2:TIME ALM1,CH1_1,0,1,20,30  
:ALARm:ANALog:SLP2:TIME? ALM1,CH1_1  
(Response) :ALARM:ANALOG:SLP2:TIME ALM1,CH1_1,0,1,20,30 (when headers are enabled)
```

## Note

---

Cannot be set to a time longer than "10000 x sampling interval"

## Usage Conditions

---

Sets and queries the time width of slope alarm.

**[Analog Channel Alarm]**

:ALARm:ANALog:STIME alm\$,ch\$,hour,min,sec

**[Waveform Calculation Alarm]**

:ALARm:CALCulate:STIME alm\$,ch\$,hour,min,sec

**[Pulse Channel Alarm]**

:ALARm:PULSe:STIME alm\$,ch\$,hour,min,sec

## Syntax

---

### Command

**[Analog Channel Alarm]**

:ALARm:ANALog:STIME alm\$,ch\$,hour,min,sec

**[Waveform Calculation Alarm]**

:ALARm:CALCulate:STIME alm\$,ch\$,hour,min,sec

**[Pulse Channel Alarm]**

:ALARm:PULSe:STIME alm\$,ch\$,hour,min,sec

### Query

**[Analog Channel Alarm]**

:ALARm:ANALog:STIME? alm\$,ch\$

**[Waveform Calculation Alarm]**

:ALARm:CALCulate:STIME? alm\$,w\$

**[Pulse Channel Alarm]**

:ALARm:PULSe:STIME? alm\$,pls\$

### Response

**[Analog Channel Alarm]**

alm\$,ch\$,hour<NR1>,min<NR1>,sec<NR1>

**[Waveform Calculation Alarm]**

alm\$,w\$,hour<NR1>,min<NR1>,sec<NR1>

**[Pulse Channel Alarm]**

alm\$,pls\$,hour<NR1>,min<NR1>,sec<NR1>

alm\$ = ALM1 to ALM4

ch\$ = CH1\_1 to CH10\_30

hour = 0 to 6 (hour)

min = 0 to 59 (min)

sec = 0 to 59 (sec)

## Details

---

Sets the time width of slope alarm of the channel designated by alm\$,ch\$/w\$/pls\$.

Returns as an NR1 numerical value the current time width of slope alarm of the channel designated by alm\$,ch\$/w\$/pls\$.

## Example

---

```
:ALARm:ANALog:STIME ALM1,CH1_1,0,1,20
```

```
:ALARm:ANALog:STIME? ALM1,CH1_1
```

```
(Response) :ALARM:ANALOG:STIME ALM1,CH1_1,0,1,20 (when headers are enabled)
```

## Note

---

0,0,0 can not be set.

## Usage Conditions

---

Sets and queries the upper limit level for a window-in/-out alarm.

**[Analog Channel Alarm]**

:ALARm:ANALog:UPPEr alm\$,ch\$,A

**[Waveform Calculation Alarm]**

:ALARm:CALCulate:UPPEr alm\$,w\$,A

**[Pulse Channel Alarm]**

:ALARm:PULSe:UPPEr alm\$,pls\$,A

## Syntax

---

### Command

**[Analog Channel Alarm]**

:ALARm:ANALog:UPPEr alm\$,ch\$,A

**[Waveform Calculation Alarm]**

:ALARm:CALCulate:UPPEr alm\$,w\$,A

**[Pulse Channel Alarm]**

:ALARm:PULSe:UPPEr alm\$,pls\$,A

### Query

**[Analog Channel Alarm]**

:ALARm:ANALog:UPPEr? alm\$,ch\$

**[Waveform Calculation Alarm]**

:ALARm:CALCulate:UPPEr? alm\$,w\$

**[Pulse Channel Alarm]**

:ALARm:PULSe:UPPEr? alm\$,pls\$

### Response

**[Analog Channel Alarm]**

alm\$,ch\$,A<NR3> (3 decimal places)

**[Waveform Calculation Alarm]**

alm\$,w\$,A<NR3> (4 decimal places)

**[Pulse Channel Alarm]**

alm\$,pls\$,A<NR3> (9 decimal places)

alm\$ = ALM1 to ALM4

ch\$ = CH1\_1 to CH10\_30

w\$ = W1 to W30

pls\$ = PLS1

[Analog Channel Alarm]

A = Configurable Range : (measurement range) $\times$ ( $\pm 1.5$ ), Minimum Resolution : (measurement range) $\times$ (1/1000)

[Waveform Calculation Alarm]

A = -9.9999E+29 to 9.9999E+29

[Pulse Channel Alarm]

A = 0 to 1000000000(count), 0 to 15000(r/s), 0 to 900000(r/min)

## Details

---

Sets the upper limit level of the window alarm of the channel designated by alm\$,ch\$/w\$/pls\$ as a voltage value.

Returns the current upper limit level of the window alarm as an NR3 numerical value.

## Example

---

```
:ALARm:ANALog:UPPEr ALM1,CH1_1,0.5
```

```
:ALARm:ANALog:UPPEr? ALM1,CH1_1
```

(Response) :ALARM:ANALOG:UPPER ALM1,CH1\_1,+5.000E-01 (when headers are enabled)

## Note

---

You cannot enter a value less than or equal to the lower limit level of the window alarm.

If you enter a value greater than the configurable limit, the maximum value is set.

If you enter a value less than the configurable lower limit, the minimum value is set.

It is also possible to set and query using the following conventional commands.

```
:ALARm:UPPEr alm$,ch$,A$ / :ALARm:UPPEr? alm$,ch$
```

```
:ALARm:WUPPEr alm$,w$,A / :ALARm:WUPPEr? alm$,w$
```

```
:ALARm:PUPPEr alm$,pls$,A / :ALARm:PUPPEr? alm$,pls$
```

## Usage Conditions

---

Sets and queries the alarm pattern for a logic alarm.

```
:ALARm:LOGic:PATtern alm$,"A$"
```

## Syntax

---

### Command

```
:ALARm:LOGic:PATtern alm$,"A$"
```

### Query

```
:ALARm:LOGic:PATtern? alm$
```

### Response

```
alm$ = ALM1 to ALM4  
"A$"  
A$ = X, 0, 1
```

## Details

---

Sets the alarm pattern for the logic alarm.

Returns the setting for the alarm pattern for the logic alarm as that specified by the given character data.

X: Ignore signal

0: Alarm at low level

1: Alarm at high level

## Example

---

```
:ALARm:LOGic:PATtern ALM1,"1"
```

```
:ALARm:LOGic:PATtern? ALM1
```

(Response) :ALARM:LOGIC:PATTERN ALM1,"1" (when headers are enabled)

## Note

---

It is also possible to set and query using the following conventional commands.

```
:ALARm:LOGPat alm$,"A$" / :ALARm:LOGPat? alm$
```

## Usage Conditions

---

## 3.10 Environment(SYSTEM)

Sets and queries the title comment addition for manual save file name.

`:SYSTEM:ADDComment A$`

### Syntax

---

#### Command

`:SYSTEM:ADDComment A$`

#### Query

`:SYSTEM:ADDComment?`

#### Response

A\$

A\$ = OFF, ON

### Details

---

Sets the title comment addition for manual save file name.

Returns the current title comment addition for manual save file name as character data.

### Example

---

`:SYSTEM:ADDComment ON`

`:SYSTEM:ADDComment?`

(Response) `:SYSTEM:ADDCOMMENT ON` (when headers are enabled)

### Note

---

### Usage Conditions

---

Sets and queries the trigger date addition for manual save file name.

**:SYSTem:ADDDate A\$**

## Syntax

---

### Command

:SYSTem:ADDDate A\$

### Query

:SYSTem:ADDDate?

### Response

A\$

A\$ = OFF, ON

## Details

---

Sets the trigger date addition for manual save file name.

Returns the current trigger date addition for manual save file name as character data.

## Example

---

:SYSTem:ADDDate ON

:SYSTem:ADDDate?

(Response) :SYSTEM:ADDDATE ON (when headers are enabled)

## Note

---

## Usage Conditions

---

Queries the adjustment date.

`:SYSTem:ADJDate? A$`

## Syntax

---

### Query

`:SYSTem:ADJDate? A$`

### Response

A\$ = MAIN, MODULE1 to MODULE10

Y<NR1>,M<NR1>,D<NR1>

Y = Year

M = Month

D = Day

## Details

---

Returns the adjustment date of the instrument or module.

MAIN: Gets the date of the last adjustment of the instrument.

MODULE1 to MODULE10: Gets the date of the last adjustment of the target module.

## Example

---

`:SYSTem:ADJDate? MAIN`

(Response) `:SYSTEM:ADJDATE 23,1,25` (when headers are enabled)

## Note

---

Y, M, and D are 0, 0, and 0 if there is no target module.

## Usage Conditions

---

Sets and queries the beep sound.

`:SYSTem:BEEP A$`

## Syntax

---

### Command

`:SYSTem:BEEP A$`

### Query

`:SYSTem:BEEP?`

### Response

A\$

A\$ = OFF, ON

## Details

---

Sets the beep sound according to character data.

Returns the current beep sound setting as character data.

## Example

---

`:SYSTem:BEEP ON`

`:SYSTem:BEEP?`

(Response) `:SYSTEM:BEEP ON` (when headers are enabled)

## Note

---

## Usage Conditions

---

Sets and queries the division of files to save numerical calculation results

**:SYSTem:CALCSplit A\$**

## Syntax

---

### Command

:SYSTem:CALCSplit A\$

### Query

:SYSTem:CALCSplit?

### Response

A\$

A\$ = OFF, ON

## Details

---

Executes the setting of the division of files to save numerical calculation results.

Returns the division of files to save numerical calculation results.

OFF: Saves the results of numerical operations in a single file.

ON: Saves the numerical operation results in a separate file for each operation.

## Example

---

:SYSTem:CALCSplit ON

:SYSTem:CALCSplit?

(Response) :SYSTEM:CALCSPLIT ON (when headers are enabled)

## Note

---

## Usage Conditions

---

Executes the ROM/RAM check and queries the result.

```
:SYSTem:CHECK  
:SYSTem:CHECK:ROMRam
```

## Syntax

---

### Command

```
:SYSTem:CHECK  
:SYSTem:CHECK:ROMRam
```

### Query

```
:SYSTem:CHECK?  
:SYSTem:CHECK:ROMRam?
```

### Response

```
A$  
A$ = NONE, RUN, PASS, FAIL
```

## Details

---

Executes the ROM/RAM check.  
Returns the ROM/RAM check result as character data.  
NONE: Not running  
RUN: Running  
PASS: Success  
FAIL: Failure

## Example

---

```
:SYSTem:CHECK:ROMRam?  
(Response) :SYSTEM:CHECK:ROMRAM PASS (when headers are enabled)
```

## Note

---

The ROM/RAM check takes about 20 minutes to complete. During ROMRAM check, each LED blinks in turn according to progress.  
It is also possible to execute and query with the conventional command:SYSTem:CHECK.

## Usage Conditions

---

Executes the module check and queries the result.

**:SYSTem:CHECK:MODule**

## Syntax

---

### Command

:SYSTem:CHECK:MODule

### Query

:SYSTem:CHECK:MODule?

### Response

m1m2m3m4m5m6m7m8m9m10

m1 to m10 = 0, 1, \*, -, R

## Details

---

Executes the module check.

Returns the module check result as character data.

0: Success

1: Failure

\*: No Modules

-: No results

R: Running

## Example

---

:SYSTem:CHECK:MODule

:SYSTem:CHECK:MODule?

(Response) :SYSTEM:CHECK:MODULE 010101\*\*\*\* (when headers are enabled)

## Note

---

## Usage Conditions

---

Executes the LAN1 check and queries the result.

```
:SYSTem:CHECK:IF:LAN1 ip1,ip2,ip3,ip4
```

## Syntax

---

### Command

```
:SYSTem:CHECK:IF:LAN1 ip1,ip2,ip3,ip4
```

### Query

```
:SYSTem:CHECK:IF:LAN1?
```

### Response

```
A$
```

```
A$ = NONE, RUN, PASS, FAIL
```

## Details

---

Specifies the IP of the ping destination.

```
ip1<NR1>,ip2<NR1>,ip3<NR1>,ip4<NR1>
```

```
ip1 = 0 to 255
```

```
ip2 = 0 to 255
```

```
ip3 = 0 to 255
```

```
ip4 = 0 to 255
```

Returns the result of executing the LAN1 check.

```
NONE: Not running
```

```
RUN: Running
```

```
PASS: Success
```

```
FAIL: Failure
```

## Example

---

```
:SYSTem:CHECK:IF:LAN1 192,168,1,1
```

```
:SYSTem:CHECK:IF:LAN1?
```

```
(Response) :SYSTEM:CHECK:IF:LAN1 PASS (when headers are enabled)
```

## Note

---

## Usage Conditions

---

Executes the LAN2 check and queries the result.

```
:SYSTem:CHECK:IF:LAN2 ip1,ip2,ip3,ip4
```

## Syntax

---

### Command

```
:SYSTem:CHECK:IF:LAN2 ip1,ip2,ip3,ip4
```

### Query

```
:SYSTem:CHECK:IF:LAN2?
```

### Response

```
A$
```

```
A$ = NONE, RUN, PASS, FAIL
```

## Details

---

Specifies the IP of the ping destination.

```
ip1<NR1>,ip2<NR1>,ip3<NR1>,ip4<NR1>
```

```
ip1 = 0 to 255
```

```
ip2 = 0 to 255
```

```
ip3 = 0 to 255
```

```
ip4 = 0 to 255
```

Returns the result of executing the LAN2 check.

```
NONE: Not running
```

```
RUN: Running
```

```
PASS: Success
```

```
FAIL: Failure
```

## Example

---

```
:SYSTem:CHECK:IF:LAN2 192,168,1,1
```

```
:SYSTem:CHECK:IF:LAN2?
```

```
(Response) :SYSTEM:CHECK:IF:LAN2 PASS (when headers are enabled)
```

## Note

---

## Usage Conditions

---

Only LR8102 is available.

Executes the media check and queries the result.

`:SYSTem:CHECK:MEDia:SD`  
`:SYSTem:CHECK:MEDia:USB`

## Syntax

---

### Command

`:SYSTem:CHECK:MEDia:SD`  
`:SYSTem:CHECK:MEDia:USB`

### Query

`:SYSTem:CHECK:MEDia:SD?`  
`:SYSTem:CHECK:MEDia:USB?`

### Response

A\$  
A\$ = NONE, RUN, PASS, FAIL, ERR

## Details

---

Executes the media check.  
Returns the media check result as character data.  
NONE: Not running  
RUN: Running  
PASS: Success  
FAIL: Failure  
ERR: Error

## Example

---

`:SYSTem:CHECK:MEDia:SD?`  
(Response) `:SYSTEM:CHECK:MEDIA:SD PASS` (when headers are enabled)

## Note

---

## Usage Conditions

---

Queries the calibration date.

`:SYSTem:CLBDate? A$`

## Syntax

---

### Query

`:SYSTem:CLBDate? A$`

### Response

A\$ = MAIN, MODULE1 to MODULE10

Y<NR1>,M<NR1>,D<NR1>

Y = Year (Y = 0 to 99)

M = Month (M = 1 to 12)

D = Day (D = 1 to 31)

## Details

---

Returns the calibration date of the instrument or module.

MAIN: Gets the date of the last calibration of the instrument.

MODULE1 to MODULE10: Gets the date of the last calibration of the target module.

## Example

---

`:SYSTem:CLBDate? MODULE1`

(Response) `:SYSTEM:CLBDATE 23,12,31` (when headers are enabled)

## Note

---

Returns 0,0,0 for modules that are not implemented.

## Usage Conditions

---

Sets and queries the operation clock.

**:SYSTem:CLOCK:OUT A\$**

## Syntax

---

### Command

:SYSTem:CLOCK:OUT A\$

### Query

:SYSTem:CLOCK:OUT?

### Response

A\$

A\$ = OFF, ON, PRECISION

## Details

---

Sets the operation clock.

Returns the setting of operation clock as character data.

OFF: No confirmation clock is output from the SMPL pin.

ON: Outputs the clock for checking clock accuracy from the SMPL pin.

PRECISION: Outputs the clock for checking time accuracy from the SMPL pin.

## Example

---

:SYSTem:CLOCK:OUT ON

:SYSTem:CLOCK:OUT?

(Response) :SYSTEM:CLOCK:OUT ON (when headers are enabled)

## Note

---

Check that the clock frequency output from the SMPL pin is within the following range.

When ON:  $32.768\text{kHz} \pm 0.000379\text{kHz}$

When PRECISION:  $10.000\text{kHz} \pm 0.0000231\text{kHz}$

## Usage Conditions

---

Sets and queries the DHCP of LAN/LAN2.

```
:SYSTem:COMMunicate:LAN:DHCP A$  
:SYSTem:COMMunicate:LAN2:DHCP A$
```

## Syntax

---

### Command

```
:SYSTem:COMMunicate:LAN:DHCP A$  
:SYSTem:COMMunicate:LAN2:DHCP A$
```

### Query

```
:SYSTEM:COMMunicate:LAN:DHCP?  
:SYSTEM:COMMunicate:LAN2:DHCP?  
:SYSTem:COMMunicate:LAN:DHCP:PREParation?  
:SYSTem:COMMunicate:LAN2:DHCP:PREParation?
```

### Response

A\$ = OFF, ON

## Details

---

The DHCP settings take effect after the  
:SYSTem:COMMunicate:LAN:UPDate/:SYSTem:COMMunicate:LAN2:UPDate command is executed.  
Sets the DHCP.  
Returns the DHCP.  
OFF: Disables the DHCP.  
ON: Enables the DHCP.  
PREParation query returns the settings before reflecting LAN/LAN2 settings.

## Example

---

```
:SYSTem:COMMunicate:LAN:DHCP ON  
:SYSTem:COMMunicate:LAN:UPDate  
:SYSTem:COMMunicate:LAN:DHCP?  
(Response) :SYSTEM:COMMUNICATE:LAN:DHCP ON (when headers are enabled)
```

## Note

---

When you turn on the DHCP server, IP address and subnet mask can be obtained automatically.

## Usage Conditions

---

Settings cannot be changed during measurement.

Sets and queries the hostname of LAN/LAN2.

```
:SYSTem:COMMunicate:LAN:HOSTname "A$"  
:SYSTem:COMMunicate:LAN2:HOSTname "A$"
```

## Syntax

---

### Command

```
:SYSTem:COMMunicate:LAN:HOSTname "A$"  
:SYSTem:COMMunicate:LAN2:HOSTname "A$"
```

### Query

```
:SYSTEM:COMMunicate:LAN:HOSTname?  
:SYSTEM:COMMunicate:LAN2:HOSTname?  
:SYSTem:COMMunicate:LAN:HOSTname:PREParation?  
:SYSTem:COMMunicate:LAN2:HOSTname:PREParation?
```

### Response

```
"A$"  
A$ = Host name string (up to 12 characters)
```

## Details

---

The hostname settings take effect after the  
:SYSTem:COMMunicate:LAN:UPDate/:SYSTem:COMMunicate:LAN2:UPDate command is executed.  
Sets the hostname  
Returns the hostname.  
PREParation query returns the settings before reflecting LAN/LAN2 settings.

## Example

---

```
:SYSTem:COMMunicate:LAN:HOSTname "LR8100"  
:SYSTem:COMMunicate:LAN:UPDate  
:SYSTem:COMMunicate:LAN:HOSTname?  
(Response) :SYSTEM:COMMUNICATE:LAN:HOSTNAME "LR8100" (when headers are enabled)
```

## Note

---

## Usage Conditions

---

Settings cannot be changed during measurement.

Sets and queries the IP address of LAN/LAN2.

```
:SYSTem:COMMunicate:LAN:IPADdress ip1,ip2,ip3,ip4  
:SYSTem:COMMunicate:LAN2:IPADdress ip1,ip2,ip3,ip4
```

## Syntax

---

### Command

```
:SYSTem:COMMunicate:LAN:IPADdress ip1,ip2,ip3,ip4  
:SYSTem:COMMunicate:LAN2:IPADdress ip1,ip2,ip3,ip4
```

### Query

```
:SYSTEM:COMMunicate:LAN:IPADdress?  
:SYSTEM:COMMunicate:LAN2:IPADdress?  
:SYSTEM:COMMunicate:LAN:IPADdress:PREParation?  
:SYSTEM:COMMunicate:LAN2:IPADdress:PREParation?
```

### Response

```
ip1<NR1>,ip2<NR1>,ip3<NR1>,ip4<NR1>  
ip1 = 0 to 255  
ip2 = 0 to 255  
ip3 = 0 to 255  
ip4 = 0 to 255
```

## Details

---

The IP address settings take effect after the  
:SYSTem:COMMunicate:LAN:UPDate/:SYSTem:COMMunicate:LAN2:UPDate command is executed.  
Sets the IP address.  
Returns the IP address.  
PREParation query returns the settings before reflecting LAN/LAN2 settings.

## Example

---

```
:SYSTem:COMMunicate:LAN:IPADdress 192,168,1,10  
:SYSTem:COMMunicate:LAN:UPDate  
:SYSTem:COMMunicate:LAN:IPADdress?  
(Response) :SYSTEM:COMMUNICATE:LAN:IPADDRESS 192,168,1,10 (when headers are enabled)
```

## Note

---

Please make sure that the settings do not overlap with other devices.  
When you turn on the DHCP server, IP address can be set automatically.

## Usage Conditions

---

Settings cannot be changed during measurement.

Sets and queries the subnet mask of LAN/LAN2.

```
:SYSTem:COMMunicate:LAN:SMASk mask1,mask2,mask3,mask4
```

```
:SYSTem:COMMunicate:LAN2:SMASk mask1,mask2,mask3,mask4
```

## Syntax

---

### Command

```
:SYSTem:COMMunicate:LAN:SMASk mask1,mask2,mask3,mask4
```

```
:SYSTem:COMMunicate:LAN2:SMASk mask1,mask2,mask3,mask4
```

### Query

```
:SYSTEM:COMMunicate:LAN:SMASk?
```

```
:SYSTEM:COMMunicate:LAN2:SMASk?
```

```
:SYSTEM:COMMunicate:LAN:SMASk:PREParation?
```

```
:SYSTEM:COMMunicate:LAN2:SMASk:PREParation?
```

### Response

```
mask1<NR1>,mask2<NR1>,mask3<NR1>,mask4<NR1>
```

```
mask1 = 0 to 255
```

```
mask2 = 0 to 255
```

```
mask3 = 0 to 255
```

```
mask4 = 0 to 255
```

## Details

---

The subnet mask settings take effect after the

:SYSTem:COMMunicate:LAN:UPDate/:SYSTem:COMMunicate:LAN2:UPDate command is executed.

Sets the subnet mask.

Returns the subnet mask.

PREParation query returns the settings before reflecting LAN/LAN2 settings.

## Example

---

```
:SYSTem:COMMunicate:LAN:SMASk 255,255,255,0
```

```
:SYSTem:COMMunicate:LAN:UPDate
```

```
:SYSTem:COMMunicate:LAN:SMASk?
```

```
(Response) :SYSTEM:COMMUNICATE:LAN:SMASK 255,255,255,0 (when headers are enabled)
```

## Note

---

Set the subnet mask to be the same as that of the devices in the same network.

When you turn on the DHCP server, the subnet mask can be set automatically.

## Usage Conditions

---

Settings cannot be changed during measurement.

Sets and queries the default gateway of LAN/LAN2.

```
:SYSTem:COMMunicate:LAN:GATeway ip1,ip2,ip3,ip4  
:SYSTem:COMMunicate:LAN2:GATeway ip1,ip2,ip3,ip4
```

## Syntax

---

### Command

```
:SYSTem:COMMunicate:LAN:GATeway ip1,ip2,ip3,ip4  
:SYSTem:COMMunicate:LAN2:GATeway ip1,ip2,ip3,ip4
```

### Query

```
:SYSTEM:COMMunicate:LAN:GATeway?  
:SYSTEM:COMMunicate:LAN2:GATeway?  
:SYSTEM:COMMunicate:LAN:GATeway:PREParation?  
:SYSTEM:COMMunicate:LAN2:GATeway:PREParation?
```

### Response

```
ip1<NR1>,ip2<NR1>,ip3<NR1>,ip4<NR1>  
ip1 = 0 to 255  
ip2 = 0 to 255  
ip3 = 0 to 255  
ip4 = 0 to 255
```

## Details

---

The default gateway settings take effect after the  
:SYSTem:COMMunicate:LAN:UPDate/:SYSTem:COMMunicate:LAN2:UPDate command is executed.  
Sets the default gateway.  
Returns the default gateway.  
PREParation query returns the settings before reflecting LAN/LAN2 settings.

## Example

---

```
:SYSTem:COMMunicate:LAN:GATeway 192,168,1,12  
:SYSTem:COMMunicate:LAN:UPDate  
:SYSTem:COMMunicate:LAN:GATeway?  
(Response) :SYSTEM:COMMUNICATE:LAN:GATEWAY 192,168,1,12 (when headers are enabled)
```

## Note

---

When you turn on the DHCP server, the default gateway can be set automatically.

## Usage Conditions

---

Settings cannot be changed during measurement.

Sets and queries the port number of LAN/LAN2.

:SYSTem:COMMunicate:LAN:CONTRol no  
:SYSTem:COMMunicate:LAN2:CONTRol no

## Syntax

---

### Command

:SYSTem:COMMunicate:LAN:CONTRol no  
:SYSTem:COMMunicate:LAN2:CONTRol no

### Query

:SYSTEM:COMMunicate:LAN:CONTRol?  
:SYSTEM:COMMunicate:LAN2:CONTRol?  
:SYSTEM:COMMunicate:LAN:CONTRol:PREPARation?  
:SYSTEM:COMMunicate:LAN2:CONTRol:PREPARation?

### Response

no<NR1>  
no = 1020 to 65520

## Details

---

The port number settings take effect after the  
:SYSTem:COMMunicate:LAN:UPDate/:SYSTem:COMMunicate:LAN2:UPDate command is executed.  
Sets the port number.  
Returns the port number.  
PREPARation query returns the settings before reflecting LAN/LAN2 settings.

## Example

---

:SYSTem:COMMunicate:LAN:CONTRol 8800  
:SYSTem:COMMunicate:LAN:UPDate  
:SYSTem:COMMunicate:LAN:CONTRol?  
(Response) :SYSTEM:COMMUNICATE:LAN:CONTROL 8800 (when headers are enabled)

## Note

---

(LAN1)  
A number between 8800 and 8809 can be specified, but 8800 will be returned in any case in that range.(Last 1 digit 0:Logger Utility, 2:Communication Command, 5:XCP on Ethernet)  
(LAN2)

A number between 8800 and 8809 can be specified, but 8800 will be returned in any case in that range.(Last 1 digit 1:UDP output, 5:XCP on Ethernet)

## **Usage Conditions**

---

Settings cannot be changed during measurement.

Sets and queries the DNS of LAN/LAN2.

```
:SYSTem:COMMunicate:LAN:DNS ip1,ip2,ip3,ip4  
:SYSTem:COMMunicate:LAN2:DNS ip1,ip2,ip3,ip4
```

## Syntax

---

### Command

```
:SYSTem:COMMunicate:LAN:DNS ip1,ip2,ip3,ip4  
:SYSTem:COMMunicate:LAN2:DNS ip1,ip2,ip3,ip4
```

### Query

```
:SYSTEM:COMMunicate:LAN:DNS?  
:SYSTEM:COMMunicate:LAN2:DNS?  
:SYSTEM:COMMunicate:LAN:DNS:PREParation?  
:SYSTEM:COMMunicate:LAN2:DNS:PREParation?
```

### Response

```
ip1<NR1>,ip2<NR1>,ip3<NR1>,ip4<NR1>  
ip1 = 0 to 255  
ip2 = 0 to 255  
ip3 = 0 to 255  
ip4 = 0 to 255
```

## Details

---

The DNS settings take effect after the  
:SYSTem:COMMunicate:LAN:UPDate/:SYSTem:COMMunicate:LAN2:UPDate command is executed.  
Sets the DNS  
Returns the DNS.  
PREParation query returns the settings before reflecting LAN/LAN2 settings.

## Example

---

```
:SYSTem:COMMunicate:LAN:DNS 192,168,1,13  
:SYSTem:COMMunicate:LAN:UPDate  
:SYSTem:COMMunicate:LAN:DNS?  
(Response) :SYSTEM:COMMUNICATE:LAN:DNS 192,168,1,13 (when headers are enabled)
```

## Note

---

Setting the DNS to 0,0,0,0 disables the gateway.

## Usage Conditions

---

Settings cannot be changed during measurement.

Determines LAN/LAN2 settings.

:SYSTEM:COMMunicate:LAN:UPDate  
:SYSTEM:COMMunicate:LAN2:UPDate

## Syntax

---

### Command

:SYSTEM:COMMunicate:LAN:UPDate  
:SYSTEM:COMMunicate:LAN2:UPDate

## Details

---

Updates LAN/LAN2 settings.

## Example

---

```
:SYSTem:COMMunicate:LAN:IPADdress 192,168,1,10  
:SYSTem:COMMunicate:LAN:SMASk 255,255,255,0  
:SYSTem:COMMunicate:LAN:GATeway 192,168,1,12  
:SYSTem:COMMunicate:LAN:CONTRol 8800  
:SYSTEM:COMMunicate:LAN:UPDate
```

## Note

---

If this command is not executed, the actual operation doesn't change.

## Usage Conditions

---

Settings cannot be changed during measurement.  
Connections in progress on the LAN is disconnected.

## Setting the destination IP address of UDP sending operation for LAN2

**:SYSTem:COMMunicate:LAN2:SEND:IPADdress ip1,ip2,ip3,ip4**

### Syntax

---

#### Command

**:SYSTem:COMMunicate:LAN2:SEND:IPADdress ip1,ip2,ip3,ip4**

#### Query

**:SYSTEM:COMMunicate:LAN2:SEND:IPADdress?**

#### Response

ip1<NR1>,ip2<NR1>,ip3<NR1>,ip4<NR1>

ip1 = 0 to 255

ip2 = 0 to 255

ip3 = 0 to 255

ip4 = 0 to 255

### Details

---

Sets the destination IP address of UDP sending operation for LAN2

Returns the destination IP address of UDP sending operation for LAN2

### Example

---

**:SYSTem:COMMunicate:LAN2:SEND:IPADdress 192,168,10,3**

**:SYSTem:COMMunicate:LAN2:SEND:IPADdress?**

(Response) **:SYSTEM:COMMUNICATE:LAN2:SEND:IPADDRESS 192,168,10,3** (when headers are enabled)

### Note

---

Specifying multicast IP address (224,0,0,0 to 239,255,255,255) enables Multicast communication

### Usage Conditions

---

Settings cannot be changed during measurement.

## Setting the destination port number of UDP sending operation for LAN2

**:SYSTem:COMMunicate:LAN2:SEND:PORT no**

### Syntax

---

#### Command

:SYSTem:COMMunicate:LAN2:SEND:PORT no

#### Query

:SYSTEM:COMMunicate:LAN2:SEND:PORT?

#### Response

no<NR1> (n = 1020 to 65535)

### Details

---

Sets the destination port number of UDP sending operation for LAN2

Returns the destination port number of UDP sending operation for LAN2

### Example

---

:SYSTem:COMMunicate:LAN2:SEND:PORT 10000

:SYSTem:COMMunicate:LAN2:SEND:PORT?

(Response) :SYSTEM:COMMUNICATE:LAN2:SEND:PORT 10000 (when headers are enabled)

### Note

---

### Usage Conditions

---

Settings cannot be changed during measurement.

## Setting the order of bytes of transmitted data of UDP sending operation for LAN2

**:SYSTem:COMMunicate:LAN2:SEND:ENDian A\$**

### Syntax

---

#### Command

:SYSTem:COMMunicate:LAN2:SEND:ENDian A\$

#### Query

:SYSTEM:COMMunicate:LAN2:SEND:ENDian?

#### Response

A\$ = BIG,LITTLE

### Details

---

Sets the order of bytes of transmitted data of UDP sending operation for LAN2

Returns the order of bytes of transmitted data of UDP sending operation for LAN2

### Example

---

:SYSTem:COMMunicate:LAN2:SEND:ENDian LITTLE

:SYSTem:COMMunicate:LAN2:SEND:ENDian?

(Response) :SYSTEM:COMMUNICATE:LAN2:SEND:ENDIAN LITTLE (when headers are enabled)

### Note

---

### Usage Conditions

---

Settings cannot be changed during measurement.

## Setting the output format of transmitted data of UDP sending operation for LAN2

**:SYSTem:COMMunicate:LAN2:SEND:FORMat A\$**

### Syntax

---

#### Command

:SYSTem:COMMunicate:LAN2:SEND:FORMat A\$

#### Query

:SYSTEM:COMMunicate:LAN2:SEND:FORMat?

#### Response

A\$ = INT32,FLOAT,INDex

### Details

---

Sets the output format of transmitted data of UDP sending operation for LAN2

Returns the output format of transmitted data of UDP sending operation for LAN2

INT32: Signed, 32-bit, INT format

FLOAT: Single-precision floating-point number format

INDex: Exponential format (ASCII)

### Example

---

:SYSTem:COMMunicate:LAN2:SEND:FORMat FLOAT

:SYSTem:COMMunicate:LAN2:SEND:FORMat?

(Response) :SYSTEM:COMMUNICATE:LAN2:SEND:FORMAT FLOAT (when headers are enabled)

### Note

---

### Usage Conditions

---

Settings cannot be changed during measurement.

Sets and queries the output for all secondary instrument data (for primary instruments)

`:SYSTem:COMMunicate:LAN2:SEND:SYNC A$`

## Syntax

---

### Command

`:SYSTem:COMMunicate:LAN2:SEND:SYNC A$`

### Query

`:SYSTEM:COMMunicate:LAN2:SEND:SYNC?`

### Response

A\$ = OFF, ON

## Details

---

Sets the output for all secondary instrument data.

Returns the output for all secondary instrument data.

OFF: When this device is the primary device, data from all secondaries is not output from LAN2 of this device.

ON: If this device is the primary device, all secondary data is output from LAN2 of this device. The output setting depends on the primary instrument. Outputs when the primary instrument's measurement value output function is set to LAN2udp.

## Example

---

`:SYSTem:COMMunicate:LAN2:SEND:SYNC ON`

`:SYSTem:COMMunicate:LAN2:SEND:SYNC?`

(Response) `:SYSTEM:COMMUNICATE:LAN2:SEND:SYNC ON` (when headers are enabled)

## Note

---

## Usage Conditions

---

Settings cannot be changed during measurement.

Clearing waveform data.

`:SYSTem:DATAClear`

## Syntax

---

### Command

`:SYSTem:DATAClear`

## Details

---

Clear the waveform data.

## Example

---

`:SYSTem:DATAClear`

## Note

---

Clearing the waveform data takes a few seconds.

## Usage Conditions

---

Sets the calendar date, and queries the current calendar date.

**:SYSTem:DATE year,month,day**

## Syntax

---

### Command

:SYSTem:DATE year,month,day

### Query

:SYSTem:DATE?

### Response

year<NR1>,month<NR1>,day<NR1>

year = 0 to 37 (year)

month = 1 to 12 (month)

day = 1 to 31 (day)

## Details

---

Sets the date on the internal calendar.

Returns the current date.

## Example

---

:SYSTem:DATE 20,1,2

:SYSTem:DATE?

(Response) :SYSTEM:DATE 20,01,02 (when headers are enabled)

## Note

---

The internal calendar cannot be set during measurement.

Setting The internal calendar takes about 1 second.

## Usage Conditions

---

Sets the date and time, and queries the current date and time.

**:SYSTem:DATETime year,month,day,hour,minute,second**

## Syntax

---

### Command

:SYSTem:DATETime year,month,day,hour,minute,second

### Query

:SYSTem:DATETime?

### Response

year<NR1>,month<NR1>,day<NR1>,hour<NR1>,minute<NR1>,second<NR1>

year = 0 to 37 (year)

month = 1 to 12 (month)

day = 1 to 31 (day)

hour = 0 to 23 (hour)

minute = 0 to 59 (minute)

second = 0 to 59(second)

## Details

---

Sets the date and time.

Returns the current date and time.

## Example

---

:SYSTem:DATETime 20,1,2,12,34,56

:SYSTem:DATETime?

(Response) :SYSTEM:DATETIME 20,01,02,12,34,56 (when headers are enabled)

## Note

---

The date and time cannot be set during measurement.

Setting the date and time takes about 1 second.

## Usage Conditions

---

Sets the date format, and queries the current date format

**:SYSTem:DFORmat A\$**

## Syntax

---

### Command

:SYSTem:DFORmat A\$

### Query

:SYSTem:DFORmat?

### Response

A\$

A\$ = YYYYMMDD ,MMDDYYYY ,DDMMYYYY

## Details

---

Sets the date format.

Returns the current date format.

## Example

---

:SYSTem:DFORmat YYYYMMDD

:SYSTem:DFORmat?

(Response) :SYSTEM:DFORMAT YYYYMMDD (when headers are enabled)

## Note

---

The date format cannot be set during measurement.

## Usage Conditions

---

Sets the date separator character, and queries the current date separator character

**:SYSTem:DSEParator A\$**

## Syntax

---

### Command

:SYSTem:DSEParator A\$

### Query

:SYSTem:DSEParator?

### Response

A\$

A\$ = HYPHen ,SLASH ,PERiod

## Details

---

Sets the date separator character.

Returns the current date separator character.

## Example

---

:SYSTem:DSEParator HYPHen

:SYSTem:DSEParator?

(Response) :SYSTEM:DSEPARATOR HYPHEN (when headers are enabled)

## Note

---

The date separator character cannot be set during measurement.

## Usage Conditions

---

Sets and queries the file name for manual save.

`:SYSTem:FILEName "A$"`

## Syntax

---

### Command

`:SYSTem:FILEName "A$"`

### Query

`:SYSTem:FILEName?`

### Response

"A\$"

A\$ = file name (up to 8 characters)

## Details

---

Sets the file name for manual save.

Returns a string of current file name for manual save.

## Example

---

`:SYSTem:FILEName "MANUAL"`

`:SYSTem:FILEName?`

(Response) `:SYSTEM:FILENAME "MANUAL"` (when headers are enabled)

## Note

---

Characters exceeding the maximum number of characters are not set.

## Usage Conditions

---

Sets and queries the language.

**:SYSTem:LANGUage A\$**

## Syntax

---

### Command

:SYSTem:LANGUage A\$

### Query

:SYSTem:LANGUage?

### Response

A\$

A\$ = JAPANese, ENGLISH

## Details

---

Sets the language.

Returns the current language setting as character data.

JAPANese: Japanese

ENGLISH: English

## Example

---

:SYSTem:LANGUage JAPANese

:SYSTem:LANGUage?

(Response) :SYSTEM:LANGUGE JAPANESE (when headers are enabled)

## Note

---

## Usage Conditions

---

Sets and queries the event mark at alarm.

`:SYSTem:MARK A$`

## Syntax

---

### Command

`:SYSTem:MARK A$`

### Query

`:SYSTem:MARK?`

### Response

A\$

A\$ = OFF, ON

## Details

---

Sets the event mark at alarm.

Returns the event mark at alarm as character data.

## Example

---

`:SYSTem:MARK ON`

`:SYSTem:MARK?`

(Response) `:SYSTEM:MARK ON` (when headers are enabled)

## Note

---

## Usage Conditions

---

Sets and queries the quick save priority.

**:SYSTem:SAVEPri A\$**

## Syntax

---

### Command

:SYSTem:SAVEPri A\$

### Query

SYSTem:SAVEPri?

### Response

A\$

A\$ = SD, USB

## Details

---

Sets the quick save priority as character data.

Returns the current setting of the quick save priority as character data.

SD: SD card

USB: USB flash drive

## Example

---

:SYSTem:SAVEPri SD

:SYSTem:SAVEPri?

(Response) :SYSTEM:SAVEPRI SD (when headers are enabled)

## Note

---

## Usage Conditions

---

Sets and queries the start backup function.

**:SYSTem:STARt A\$**

## Syntax

---

### Command

:SYSTem:STARt A\$

### Query

:SYSTem:STARt?

### Response

A\$

A\$ = OFF, ON

## Details

---

Enables and disables the start backup function.

Returns the current enablement state of the start backup function as character data.

## Example

---

:SYSTem:STARt ON

:SYSTem:STARt?

(Response) :SYSTEM:START ON (when headers are enabled)

## Note

---

## Usage Conditions

---

## Sets and queries the thinning save for auto save

**:SYSTem:THINData A\$**

### Syntax

---

#### Command

:SYSTem:THINData A\$

#### Query

:SYSTem:THINData?

#### Response

A\$

A\$ = INSTANT, STATISTICS

### Details

---

Sets the thinning save for auto save.

Returns the thinning save for auto save.

INSTANT: Saves the data at the beginning.

STATISTICS: Saves statistical data (maximum value, minimum value, average value, and data at the beginning).

### Example

---

:SYSTem:THINData INSTANT

:SYSTem:THINData?

(Response) :SYSTEM:THINDATA INSTANT (when headers are enabled)

### Note

---

Sets the save format to text and sets 2 or more in the parameter of :SYSTem:THINOut command.

### Usage Conditions

---

Sets and queries the thinning save for manual save.

**:SYSTem:THINOut A**

## Syntax

---

### Command

:SYSTem:THINOut A

### Query

:SYSTem:THINOut?

### Response

A<NR1>

A = 1(OFF) to 100000

## Details

---

Sets the text save thinning interval for manual save.

Returns the text save thinning interval for the current manual save.

If A=1, downsampling is off.

## Example

---

:SYSTem:THINOut 5

:SYSTem:THINOut?

(Response) :SYSTEM:THINOUT 5 (when headers are enabled)

## Note

---

## Usage Conditions

---

Sets the time, and queries the current time.

**:SYSTem:TIME h,m,s**

## Syntax

---

### Command

:SYSTem:TIME h,m,s

### Query

:SYSTem:TIME?

### Response

h<NR1>,m<NR1>,s<NR1>

h = 0 to 23 (hour)

m = 0 to 59 (min)

s = 0 to 59 (sec)

## Details

---

Sets the time.

Returns the current time as an NR1 numerical value.

## Example

---

:SYSTem:TIME 12,34,56

:SYSTem:TIME?

(Response) :SYSTEM:TIME 12,34,56 (when headers are enabled)

## Note

---

The time cannot be set during measurement.

Setting the time takes about 1 second.

## Usage Conditions

---

Sets and queries the time zone.

**:SYSTem:TIMEZone hour(,min)**

## Syntax

---

### Command

:SYSTem:TIMEZone hour(,min)

### Query

:SYSTem:TIMEZone?

### Response

hour<NR1>(,min<NR1>)

hour = -12 to +14 (hour)

min = 30, 45 (min) (0 min when omitted)

## Details

---

Sets the time zone.

Returns the current time zone as an NR1 numerical value.

## Example

---

:SYSTem:TIMEZone 9

:SYSTem:TIMEZone?

(Response) :SYSTEM:TIMEZONE +9 (when headers are enabled)

## Note

---

When you change the time zone, the time on the clock changes accordingly.

When the combination of hours and minutes cannot be set will result in an error.

## Usage Conditions

---

Sets and queries the time axis display.

**:SYSTem:TMAXis A\$**

## Syntax

---

### Command

:SYSTem:TMAXis A\$

### Query

:SYSTem:TMAXis?

### Response

A\$

A\$ = TIME, DATE, SCALE

## Details

---

Sets the time axis display as character data.

Returns the current time axis display setting as character data.

TIME: Time

DATE: Date

SCALE: Data points

## Example

---

:SYSTem:TMAXis TIME

:SYSTem:TMAXis?

(Response) :SYSTEM:TMAXIS TIME (when headers are enabled)

## Note

---

## Usage Conditions

---

Sets and queries the input kind of external input terminal 1.

`:SYSTem:EXT:IO1:KIND A$`

## Syntax

---

### Command

`:SYSTem:EXT:IO1:KIND A$`

### Query

`:SYSTem:EXT:IO1:KIND?`

### Response

A\$

A\$ = OFF, STARTIN, STOPIN, S\_SIN, EVENTIN

## Details

---

Sets the input kind of external input terminal 1

Returns the current input kind of external input terminal 1 as character data.

OFF

STARTIN: Start

STOPIN: Stop

S\_SIN: Start & Stop

EVENTIN: Event input

## Example

---

`:SYSTem:EXT:IO1:KIND STARTIN`

`:SYSTem:EXT:IO1:KIND?`

(Response) `:SYSTEM:EXT:IO1:KIND STARTIN` (when headers are enabled)

## Note

---

## Usage Conditions

---

Sets and queries the start slope of external input terminal 1.

`:SYSTem:EXT:IO1:SLOPe:STARt A$`

## Syntax

---

### Command

`:SYSTem:EXT:IO1:SLOPe:STARt A$`

### Query

`:SYSTem:EXT:IO1:SLOPe:STARt?`

### Response

A\$

A\$ = UP, DOWN

## Details

---

Sets the start slope of external input terminal 1.

Returns the current start slope of external input terminal 1 as character data.

UP: Rising

DOWN: Falling

## Example

---

`:SYSTem:EXT:IO1:SLOPe:STARt UP`

`:SYSTem:EXT:IO1:SLOPe:STARt?`

(Response) `:SYSTEM:EXT:IO1:SLOPE:START UP` (when headers are enabled)

## Note

---

## Usage Conditions

---

Sets and queries the stop slope of external input terminal 1.

`:SYSTem:EXT:IO1:SLOPe:STOP A$`

## Syntax

---

### Command

`:SYSTem:EXT:IO1:SLOPe:STOP A$`

### Query

`:SYSTem:EXT:IO1:SLOPe:STOP?`

### Response

A\$

A\$ = UP, DOWN

## Details

---

Sets the stop slope of external input terminal 1.

Returns the current stop slope of external input terminal 1 as character data.

UP: Rising

DOWN: Falling

## Example

---

`:SYSTem:EXT:IO1:SLOPe:STOP UP`

`:SYSTem:EXT:IO1:SLOPe:STOP?`

(Response) `:SYSTEM:EXT:IO1:SLOPE:STOP UP` (when headers are enabled)

## Note

---

## Usage Conditions

---

Sets and queries the input kind of external input terminal 2.

`:SYSTem:EXT:IO2:KIND A$`

## Syntax

---

### Command

`:SYSTem:EXT:IO2:KIND A$`

### Query

`:SYSTem:EXT:IO2:KIND?`

### Response

A\$

A\$ = OFF, STARTIN, STOPIN, S\_SIN, EVENTIN

## Details

---

Sets the input kind of external input terminal 2

Returns the current input kind of external input terminal 2 as character data.

OFF

STARTIN: Start

STOPIN: Stop

S\_SIN: Start & Stop

EVENTIN: Event input

## Example

---

`:SYSTem:EXT:IO2:KIND STARTIN`

`:SYSTem:EXT:IO2:KIND?`

(Response) `:SYSTEM:EXT:IO2:KIND STARTIN` (when headers are enabled)

## Note

---

## Usage Conditions

---

Sets and queries the start slope of external input terminal 2.

`:SYSTem:EXT:IO2:SLOPe:STARt A$`

## Syntax

---

### Command

`:SYSTem:EXT:IO2:SLOPe:STARt A$`

### Query

`:SYSTem:EXT:IO2:SLOPe:STARt?`

### Response

A\$

A\$ = UP, DOWN

## Details

---

Sets the start slope of external input terminal 2.

Returns the current start slope of external input terminal 2 as character data.

UP: Rising

DOWN: Falling

## Example

---

`:SYSTem:EXT:IO2:SLOPe:STARt UP`

`:SYSTem:EXT:IO2:SLOPe:STARt?`

(Response) `:SYSTEM:EXT:IO2:SLOPE:START UP` (when headers are enabled)

## Note

---

## Usage Conditions

---

Sets and queries the stop slope of external input terminal 2.

`:SYSTem:EXT:IO2:SLOPe:STOP A$`

## Syntax

---

### Command

`:SYSTem:EXT:IO2:SLOPe:STOP A$`

### Query

`:SYSTem:EXT:IO2:SLOPe:STOP?`

### Response

A\$

A\$ = UP, DOWN

## Details

---

Sets the stop slope of external input terminal 2.

Returns the current stop slope of external input terminal 2 as character data.

UP: Rising

DOWN: Falling

## Example

---

`:SYSTem:EXT:IO2:SLOPe:STOP UP`

`:SYSTem:EXT:IO2:SLOPe:STOP?`

(Response) `:SYSTEM:EXT:IO2:SLOPE:STOP UP` (when headers are enabled)

## Note

---

## Usage Conditions

---

Sets and queries the input kind of external input terminal 3.

`:SYSTem:EXT:IO3:KIND A$`

## Syntax

---

### Command

`:SYSTem:EXT:IO3:KIND A$`

### Query

`:SYSTem:EXT:IO3:KIND?`

### Response

A\$

A\$ = OFF, TRIGIN, EVENTIN

## Details

---

Sets the input kind of external input terminal 3

Returns the current input kind of external input terminal 3 as character data.

OFF

TRIGIN: Trigger input

EVENTIN: Event input

## Example

---

`:SYSTem:EXT:IO3:KIND TRIGIN`

`:SYSTem:EXT:IO3:KIND?`

(Response) `:SYSTEM:EXT:IO3:KIND TRIGIN` (when headers are enabled)

## Note

---

When the external trigger setting is ON, the setting cannot be changed except for the trigger input.

## Usage Conditions

---

Sets and queries the start slope of external input terminal 3.

`:SYSTem:EXT:IO3:SLOPe:STARt A$`

## Syntax

---

### Command

`:SYSTem:EXT:IO3:SLOPe:STARt A$`

### Query

`:SYSTem:EXT:IO3:SLOPe:STARt?`

### Response

A\$

A\$ = UP, DOWN

## Details

---

Sets the start slope of external input terminal 3.

Returns the current start slope of external input terminal 3 as character data.

UP: Rising

DOWN: Falling

## Example

---

`:SYSTem:EXT:IO3:SLOPe:STARt UP`

`:SYSTem:EXT:IO3:SLOPe:STARt?`

(Response) `:SYSTEM:EXT:IO3:SLOPE:START UP` (when headers are enabled)

## Note

---

## Usage Conditions

---

Sets and queries the stop slope of external input terminal 3.

`:SYSTem:EXT:IO3:SLOPe:STOP A$`

## Syntax

---

### Command

`:SYSTem:EXT:IO3:SLOPe:STOP A$`

### Query

`:SYSTem:EXT:IO3:SLOPe:STOP?`

### Response

A\$

A\$ = UP, DOWN

## Details

---

Sets the stop slope of external input terminal 3.

Returns the current stop slope of external input terminal 3 as character data.

UP: Rising

DOWN: Falling

## Example

---

`:SYSTem:EXT:IO3:SLOPe:STOP UP`

`:SYSTem:EXT:IO3:SLOPe:STOP?`

(Response) `:SYSTEM:EXT:IO3:SLOPE:STOP UP` (when headers are enabled)

## Note

---

## Usage Conditions

---

Sets and queries the output kind of external output terminal.

`:SYSTem:EXT:IO4:KIND A$`

## Syntax

---

### Command

`:SYSTem:EXT:IO4:KIND A$`

### Query

`:SYSTem:EXT:IO4:KIND?`

### Response

A\$

A\$ = OFF, TRIGOUT

## Details

---

Sets the output kind of external output terminal.

Returns the current output kind of external output terminal as character data.

OFF

TRIGOUT: Trigger output

## Example

---

`:SYSTem:EXT:IO4:KIND TRIGOUT`

`:SYSTem:EXT:IO4:KIND?`

(Response) `:SYSTEM:EXT:IO4:KIND TRIGOUT` (when headers are enabled)

## Note

---

## Usage Conditions

---

Sets and queries the filter of external sampling and external trigger terminals.

**:SYSTem:EXTFILTer A\$**

## Syntax

---

### Command

:SYSTem:EXTFILTer A\$

### Query

:SYSTem:EXTFILTer?

### Response

A\$

A\$ = OFF, ON

## Details

---

Sets the filter for the external sampling and external trigger terminals.

Returns the current filter setting of the external sampling and external trigger terminals.

OFF: Disables the filter.

ON: Enables the filter.

## Example

---

:SYSTem:EXTFILTer ON

:SYSTem:EXTFILTer?

(Response) :SYSTEM:EXTFILTER ON (when headers are enabled)

## Note

---

## Usage Conditions

---

Sets and queries the slope of external sampling and external trigger terminals.

`:SYSTem:EXTSLOPe A$`

## Syntax

---

### Command

`:SYSTem:EXTSLOPe A$`

### Query

`:SYSTem:EXTSLOPe?`

### Response

A\$

A\$ = UP, DOWN

## Details

---

Sets the slope for the external sampling and external trigger terminals.

Returns the current slope setting of the external sampling and external trigger terminals.

UP: Operates on the rising edge from Low level to High level.

DOWN: Operates on the falling edge from High level to Low level.

## Example

---

`:SYSTem:EXTSLOPe UP`

`:SYSTem:EXTSLOPe?`

(Response) `:SYSTEM:EXTSLOPE UP` (when headers are enabled)

## Note

---

## Usage Conditions

---

Sets and queries NTP server address.

```
:SYSTem:NTP:ADDRess "A$"
```

## Syntax

---

### Command

```
:SYSTem:NTP:ADDRess "A$"
```

### Query

```
:SYSTem:NTP:ADDRess?
```

### Response

```
"A$"
```

A\$ = NTP server address(64 characters)

## Details

---

Sets the server address for NTP synchronization.

Returns the server address for NTP synchronization.

## Example

---

```
:SYSTem:NTP:ADDRess "abcdef.com"
```

```
:SYSTem:NTP:ADDRess?
```

(Response) :SYSTEM:NTP:ADDRESS "abcdef.com" (when headers are enabled)

## Note

---

If maximum number of characters is exceeded, a command error occurs.

## Usage Conditions

---

Executes the NTP time synchronization and query the result.

**:SYSTem:NTP:CHECK?**

## Syntax

---

### Query

:SYSTem:NTP:CHECK?

### Response

A

A = 0(PASS), 1(FAIL)

## Details

---

Executes the NTP time synchronization and query the result.

## Example

---

:SYSTem:NTP:CHECK?

(Response) :SYSTEM:NTP:CHECK 0 (when headers are enabled)

## Note

---

## Usage Conditions

---

The NTP client function must be set to ON.

Sets and queries the NTP client function.

**:SYSTem:NTP:KIND A\$**

## Syntax

---

### Command

:SYSTem:NTP:KIND A\$

### Query

SYSTem:NTP:KIND?

### Response

A\$

A\$ = OFF, ON

## Details

---

Sets the NTP client function.

Returns the NTP client function.

OFF: Disable

ON: Enable

## Example

---

:SYSTem:NTP:KIND ON

:SYSTem:NTP:KIND?

(Response) :SYSTEM:NTP:KIND ON (when headers are enabled)

## Note

---

## Usage Conditions

---

Sets and queries the timing of NTP synchronization.

**:SYSTem:NTP:SYNC A\$**

## Syntax

---

### Command

:SYSTem:NTP:SYNC A\$

### Query

:SYSTem:NTP:SYNC?

### Response

A\$

A\$ = OFF, HOUR, DAY

## Details

---

Sets the timing of NTP synchronization.

Returns the timing of NTP synchronization.

OFF: No Sync

HOUR: Sync Every Hour

DAY: Sync Every Day

## Example

---

:SYSTem:NTP:SYNC HOUR

:SYSTem:NTP:SYNC?

(Response) :SYSTEM:NTP:SYNC HOUR (when headers are enabled)

## Note

---

## Usage Conditions

---

## Sets and queries the NTP synchronization before start measuring

**:SYSTem:NTP:STARt A\$**

### Syntax

---

#### Command

:SYSTem:NTP:STARt A\$

#### Query

:SYSTem:NTP:STARt?

#### Response

A\$

A\$ = OFF, ON

### Details

---

Sets the the NTP synchronization before start measuring.

Returns the the NTP synchronization before start measuring.

OFF: Disable

ON: Enable

### Example

---

:SYSTem:NTP:STARt ON

:SYSTem:NTP:STARt?

(Response) :SYSTEM:NTP:START ON (when headers are enabled)

### Note

---

### Usage Conditions

---

Sets and queries the FTP server address for FTP data auto send.

`:SYSTem:FTP:ADDRess "A$"`

## Syntax

---

### Command

`:SYSTem:FTP:ADDRess "A$"`

### Query

`:SYSTem:FTP:ADDRess?`

### Response

`"A$"`

A\$ = FTP server address(45 characters)

## Details

---

Sets the address of the Destination FTP server in the FTP data auto send. Queries to the address of the destination FTP server.

## Example

---

`:SYSTem:FTP:ADDRess "abcdef"`

`:SYSTem:FTP:ADDRess?`

(Response) `:SYSTEM:FTP:ADDRESS "abcdef"` (when headers are enabled)

## Note

---

If maximum number of characters is exceeded, a command error occurs.

## Usage Conditions

---

## Sets and queries the Deletion of Sent File.

`:SYSTem:FTP:AUTODel A$`

### Syntax

---

#### Command

`:SYSTem:FTP:AUTODel A$`

#### Query

`:SYSTem:FTP:AUTODel?`

#### Response

A\$

A\$ = OFF, ON

### Details

---

Sets the sent file deletion feature for FTP date auto send.

Returns sent file deletion setting.

OFF: OFF

ON: ON

### Example

---

`:SYSTem:FTP:AUTODel ON`

`:SYSTem:FTP:AUTODel?`

(Response) `:SYSTEM:FTP:AUTODEL ON` (when headers are enabled)

### Note

---

### Usage Conditions

---

Sets and queries the verification of server certificate.

**:SYSTem:FTP:CErTificate A\$**

## Syntax

---

### Command

:SYSTem:FTP:CErTificate A\$

### Query

:SYSTem:FTP:CErTificate?

### Response

A\$

A\$ = OFF, ON

## Details

---

Sets the verification of server certificate.

Returns the verification of server certificate setting.

OFF: OFF

ON: ON

## Example

---

:SYSTem:FTP:CErTificate ON

:SYSTem:FTP:CErTificate?

(Response) :SYSTEM:FTP:CErTIFICATE ON (when headers are enabled)

## Note

---

## Usage Conditions

---

Executes the auto send test of FTP data and queries the result.

**:SYSTem:FTP:CHECK?**

## Syntax

---

### Query

:SYSTem:FTP:CHECK?

### Response

A

A = 0(PASS), 1(FAIL)

## Details

---

Executes the auto send test of FTP data and queries the result.

## Example

---

:SYSTem:FTP:CHECK?

(Response) :SYSTEM:FTP:CHECK 0 (when headers are enabled)

## Note

---

## Usage Conditions

---

Sets and queries the Directory for FTP data auto send.

```
:SYSTem:FTP:DIR "A$"
```

## Syntax

---

### Command

```
:SYSTem:FTP:DIR "A$"
```

### Query

```
:SYSTem:FTP:DIR?
```

### Response

```
"A$"
```

A\$ = Destination Directory(45 characters)

## Details

---

Sets the Directory of the Destination FTP server in the FTP data auto send.

Returns to the Directory of the destination FTP server.

## Example

---

```
:SYSTem:FTP:DIR "/abc/def/"
```

```
:SYSTem:FTP:DIR?
```

(Response) :SYSTEM:FTP:DIR "/abc/def/" (when headers are enabled)

## Note

---

If maximum number of characters is exceeded, a command error occurs.

## Usage Conditions

---

Sets and queries the Filename identifier(Hostname).

**:SYSTem:FTP:FILE:HOST A\$**

## Syntax

---

### Command

:SYSTem:FTP:FILE:HOST A\$

### Query

:SYSTem:FTP:FILE:HOST?

### Response

A\$

A\$ = OFF, ON

## Details

---

Sets whether to give the sending file a hostname as an identifier.

Returns the current sending file identifier(Hostname) setting.

OFF: Not append hostname

ON: Append hostname

## Example

---

:SYSTem:FTP:FILE:HOST ON

:SYSTem:FTP:FILE:HOST?

(Response) :SYSTEM:FTP:FILE:HOST ON (when headers are enabled)

## Note

---

## Usage Conditions

---

Sets and queries the Filename identifier(IP address).

`:SYSTem:FTP:FILE:IP A$`

## Syntax

---

### Command

`:SYSTem:FTP:FILE:IP A$`

### Query

`:SYSTem:FTP:FILE:IP?`

### Response

A\$

A\$ = OFF, ON

## Details

---

Sets whether to give the sending file an IP address as an identifier.

Returns the current sending file identifier(IP address) setting.

OFF: Not append IP address

ON: Append IP address

## Example

---

`:SYSTem:FTP:FILE:IP ON`

`:SYSTem:FTP:FILE:IP?`

(Response) `:SYSTEM:FTP:FILE:IP ON` (when headers are enabled)

## Note

---

## Usage Conditions

---

Sets and queries the Filename identifier(Time).

**:SYSTem:FTP:FILE:TIME A\$**

## Syntax

---

### Command

:SYSTem:FTP:FILE:TIME A\$

### Query

:SYSTem:FTP:FILE:TIME?

### Response

A\$

A\$ = OFF, ON

## Details

---

Sets whether to give the sending file a time as an identifier.

Returns the current sending file identifier(Time) setting.

OFF: Not append Time

ON: Append Time

## Example

---

:SYSTem:FTP:FILE:TIME ON

:SYSTem:FTP:FILE:TIME?

(Response) :SYSTEM:FTP:FILE:TIME ON (when headers are enabled)

## Note

---

## Usage Conditions

---

Sets the Password for FTP data auto send. Check that the password is correct and Queries the result.

```
:SYSTem:FTP:PASSword "A$"
```

## Syntax

---

### Command

```
:SYSTem:FTP:PASSword "A$"
```

### Query

```
:SYSTem:FTP:PASSword? "A$"
```

### Response

B\$

A\$ = Password(32 characters)

B\$ = PASS,FAIL

PASS: Password is correct

FAIL: Password is incorrect

## Details

---

Sets the Password for FTP data auto send.

Check that the password is correct and Queries the result.

## Example

---

```
:SYSTem:FTP:PASSword "abcd"
```

```
:SYSTem:FTP:PASSword? "abcd"
```

(Response) :SYSTEM:FTP:PASSWORD PASS (when headers are enabled)

## Note

---

If maximum number of characters is exceeded, a command error occurs.

## Usage Conditions

---

Sets and queries the Passive mode for FTP data auto send.

**:SYSTem:FTP:PASV A\$**

## Syntax

---

### Command

:SYSTem:FTP:PASV A\$

### Query

:SYSTem:FTP:PASV?

### Response

A\$

A\$ = OFF, ON

## Details

---

Sets the passive mode ON/OFF for FTP data auto send.

Returns the current passive mode setting.

OFF: Passive mode OFF

ON: Passive mode ON

## Example

---

:SYSTem:FTP:PASV ON

:SYSTem:FTP:PASV?

(Response) :SYSTEM:FTP:PASV ON (when headers are enabled)

## Note

---

## Usage Conditions

---

Sets and queries the Port number for FTP data auto send.

**:SYSTem:FTP:PORT A**

## Syntax

---

### Command

:SYSTem:FTP:PORT A

### Query

:SYSTem:FTP:PORT?

### Response

A<NR1>

A = Port number(0 to 65535)

## Details

---

Sets the Port number for FTP data auto send.

Returns the currently set server port number.

## Example

---

:SYSTem:FTP:PORT 1234

:SYSTem:FTP:PORT?

(Response) :SYSTEM:FTP:PORT 1234 (when headers are enabled)

## Note

---

## Usage Conditions

---

Queries whether there are unsent files.

`:SYSTem:FTP:PROGress?`

## Syntax

---

### Query

`:SYSTem:FTP:PROGress?`

### Response

A\$

A\$ = YES, NO

## Details

---

Checking the existence of unsent files in the FTP data auto sent function and queries the result.

YES: There are unsent files

NO: No unsent files

## Example

---

`:SYSTem:FTP:PROGress?`

(Response) `:SYSTEM:FTP:PROGRESS YES` (when headers are enabled)

## Note

---

## Usage Conditions

---

Sets and queries the FTP connection security.

`:SYSTem:FTP:SECURity A$`

## Syntax

---

### Command

`:SYSTem:FTP:SECURity A$`

### Query

`:SYSTem:FTP:SECURity?`

### Response

A\$

A\$ = OFF, EXPLICIT, IMPLICIT

## Details

---

Sets the FTP connection security.

Returns the current FTP connection security settings.

OFF: FTP connection security function is OFF

EXPLICIT: Explicit FTP over TLS

IMPLICIT: Implicit FTP over TLS

## Example

---

`:SYSTem:FTP:SECURity EXPLICIT`

`:SYSTem:FTP:SECURity?`

(Response) `:SYSTEM:FTP:SECURITY EXPLICIT` (when headers are enabled)

## Note

---

## Usage Conditions

---

Queries FTP communication status.

`:SYSTem:FTP:STATe?`

## Syntax

---

### Query

`:SYSTem:FTP:STATe?`

### Response

`A<NR1>,B<NR1>,C<NR1>,D<NR1>`

A = Number of total files

B = Number of sent files

C = Number of failed files to send

D = Number of unsent files

## Details

---

Queries the file transmission status of the FTP data auto send function.

## Example

---

`:SYSTem:FTP:STATe?`

(Response) `:SYSTEM:FTP:STATE 10,1,5,4` (when headers are enabled)

## Note

---

## Usage Conditions

---

Sets and queries the FTP data auto send.

**:SYSTem:FTP:USE A\$**

## Syntax

---

### Command

:SYSTem:FTP:USE A\$

### Query

:SYSTem:FTP:USE?

### Response

A\$

A\$ = OFF, ON

## Details

---

Sets the ON/OFF of the FTP data auto send function.

Returns the current FTP data auto send function settings

OFF: FTP data auto send function is OFF

ON: FTP data auto send function is ON

## Example

---

:SYSTem:FTP:USE ON

:SYSTem:FTP:USE?

(Response) :SYSTEM:FTP:USE ON (when headers are enabled)

## Note

---

## Usage Conditions

---

Sets and queries the User Name for FTP data auto send.

```
:SYSTem:FTP:USER "A$"
```

## Syntax

---

### Command

```
:SYSTem:FTP:USER "A$"
```

### Query

```
:SYSTem:FTP:USER?
```

### Response

```
"A$"
```

A\$ = User Name(32 characters)

## Details

---

Sets the User name for FTP data auto send.

Returns the currently set user name.

## Example

---

```
:SYSTem:FTP:USER "LR8101"
```

```
:SYSTem:FTP:USER?
```

(Response) :SYSTEM:FTP:USER "LR8101" (when headers are enabled)

## Note

---

If maximum number of characters is exceeded, a command error occurs.

## Usage Conditions

---

Sets and queries the measured value output function.

**:SYSTem:RTOuT A\$**

## Syntax

---

### Command

:SYSTem:RTOuT A\$

### Query

:SYSTem:RTOuT?

### Response

A\$

A\$ = OFF, CAN, LAN2udp

## Details

---

Sets the measured value output function.

Returns the measured value output function.

OFF: No data output from CAN and LAN2.

CAN: Outputs data from CAN terminal.

LAN2udp: Outputs data from LAN2.

## Example

---

:SYSTem:RTOuT CAN

:SYSTem:RTOuT?

(Response) :SYSTEM:RTOuT CAN (when headers are enabled)

## Note

---

## Usage Conditions

---

## 3.11 Memory(MEMory)

Outputs data from memory.(ASCII)

:MEMory:ADATa? A

### Syntax

#### Query

:MEMory:ADATa? A

#### Response

B1,B2,...<NR1>(Wave Calc Result = <NR3>)

A = 1 to 2000(amount of output data)

Bi = -2147483648 to 2147483647(Analog)

Bi = 0 to 2147483647(Count, Revolve)

Bi = 0 to 1(Logic)

Bi = 0 to 15(Alarm)

Bi = Wave Calc Result(Wave Calc)

### Details

The number of data values specified by A are output from the memory channel and point set by the :MEMory:APOINT.

The output point is incremented by the number of data values.

Use the following formula to convert analog data values to physical quantities.

physical quantities = (data value) × (voltage range) / (number of data per range)

If scaling is set, the following formula is used.

Converted value = Physical quantity × Scaling factor + Scaling offset

The number of data per range is shown in the table below.

Module		Mode	Number of data per range *The number of data from 0V to 1V in the 1V range
M7100 M7102	Voltage/Temp Module(15ch) Voltage/Temp Module(30ch)	Voltage(All range) ※1-5V is the same as 6V	100000
		Thermocouple(100°C range)	10000
		Thermocouple(500°C range)	10000
		Thermocouple(2000°C range)	20000

Count, Wave Calc returns a measurement value.

The number of revolutions can be calculated by dividing the data value by the number of pulses per revolution set in :MODule:PCOUNT.

revolutions = data value / number of pulses per revolution  
The alarm channel returns an integer representation of all channels (4bits).  
(Output from the bottom bit to alarm 1, alarm 2...)

## Example

---

:MEMory:APOINT CH1\_1,0  
:MEMory:ADATa? 10  
(Response) :MEMORY:ADATA 3176,3176,3176,3186,3186,3186,3186,3186,3198,3198 (when headers are enabled)

## Note

---

The correspondence between the value of A and the main module notation is as follows.

Value of A	Main unit notation
2147483647	+OVER
-2147483648	-OVER
2147483646	BURNOUT
2147483645	NO DATA

When 1,000,000 points of analog channel data are recorded, the time required to send the command ":MEMory:ADATa? 2000" 500 times is about 26 seconds.  
(Running Microsoft Windows 10 Pro (22H2) on a Intel(R) i7-9700F 3.00GHz CPU and 16 GB RAM).  
Please note that these values are for reference only, and depend on the actual data transfer speed of the LAN connection.

## Usage Conditions

---

The output point must be set to :MEMory:POINT.  
Provided that the output point is lower than the amount of data stored.  
Provided that stored data is present.

## Outputs hold data.(ASCII)

`:MEMory:AFETch? ch$`

### Syntax

---

#### Query

`:MEMory:AFETch? ch$`

#### Response

A<NR1>(Wave Calc Result = <NR3>  
ch\$ = CH1\_1 to CH10\_30, PLS1, LOG, ALARM, W1 to W30  
A = -2147483648 to 2147483647(Analog)  
A = 0 to 2147483647(Count, Revolve)  
A = 0 to 1(Logic)  
A = 0 to 15(Alarm)  
A = Wave Calc Result(Wave Calc)

### Details

---

Returns the value of the channel specified by ch\$ from the hold data captured by the :MEMory:GETReal.  
The relationship between the returned data value and the measured value is same as :MEMory:ADATa?.

### Example

---

`:MEMory:AFETch? CH1_1`  
(Response) :MEMORY:AFETCH 3176 (when headers are enabled)

### Note

---

If you have not run the :MEMory:GETReal before this command, it returns 2147483645 as a value indicating NO DATA.

### Usage Conditions

---

Queries the end of data samples stored.(when longer data is stored than the inside memory)

**:MEMory:AMAXPoint?**

## Syntax

---

### Query

**:MEMory:AMAXPoint?**

### Response

A<NR1>

A = 0(no data stored), 1 to

## Details

---

Returns the end of data samples stored in the memory as a numerical value in NR1 format.

## Example

---

**:MEMory:AMAXPoint?**

(Response) **:MEMORY:AMAXPOINT 800** (when headers are enabled)

## Note

---

## Usage Conditions

---

Sets and queries the point in memory for output.(when longer data is stored than the inside memory)

**:MEMory:APOINT ch\$,A**

## Syntax

---

### Command

:MEMory:APOINT ch\$,A

### Query

:MEMory:APOINT?

### Response

ch\$,A<NR1>  
ch\$ = CH1\_1 to CH10\_30, PLS1, LOG, ALARM, W1 to W30  
A = 0 to (number of storage data - 1)

## Details

---

Sets the output point in memory.  
Returns the current output point in memory as an NR1 numerical value.

## Example

---

:MEMory:APOINT CH1\_1,100  
:MEMory:APOINT?  
(Response) :MEMORY:APOINT CH1\_1,100 (when headers are enabled)

## Note

---

If there is no stored data, the read pointer cannot be set.  
Argument A can be set only to a value less than that returned by the :MEMory:AMAXPoint?.  
If the measurement data is longer than the internal memory, the data for the specified point may not be available.

## Usage Conditions

---

## Output real time data (ASCII)

**:MEMory:AREAL? ch\$**

### Syntax

---

#### Query

:MEMory:AREAL? ch\$

#### Response

A<NR1>(Wave Calc Result = <NR3>  
ch\$ = CH1\_1 to CH10\_30, PLS1, LOG, ALARM, W1 to W30  
A = -2147483648 to 2147483647(Analog)  
A = 0 to 2147483647(Count, Revolve)  
A = 0 to 1(Logic)  
A = 0 to 15(Alarm)  
A = Wave Calc Result(Wave Calc)

### Details

---

Returns the final measurement of the channel designated by ch\$.

### Example

---

:MEMory:AREAL? CH1\_1  
(Response) :MEMORY:AREAL 3176 (when headers are enabled)

### Note

---

Returns the value of the final storage data if any of the following operations are performed to capture the measurement data.

- Measurements
- Display Monitor Values
- Execute :MEMory:GETReal

If the above operation is not performed, 2147483645 is returned as a value indicating NO DATA.

### Usage Conditions

---

## Binary transfer of stored data.

**:MEMory:BDATA? A**

### Syntax

---

#### Query

:MEMory:BDATA? A

#### Response

#0<binary data>

A = 1 to 5000 (Output number)

### Details

---

Outputs the data stored by a :MEMory:POINT command specification in binary format. The data format is big-endian.

The output point is incremented by the number of data values.

The read data is structured as follows:

(1)Initially: "#0" (Indicates binary format)

(2)After "#0", data in binary format is returned by the number of data specified in A.

Data sizes per point for each channel type are as follows:

- Logic, Alarm:2byte
- Analog, Pulse:4byte
- Wave Calc:8byte

(The response of the waveform calc channel is output in double-precision floating-point format (IEEE 754 format).)

The data obtained is the same as that for :MEMory:ADATA?. For details refer to :MEMory:ADATA?.

### Example

---

:MEMory:POINT CH1\_1,0

:MEMory:BDATA? 10

(Response) :MEMory:BDATA #0...(binary data) (when headers are enabled)

### Note

---

Returns an error value if no storage data exists for the specified channel/output point.

Error values for each channel type are as follows.

[Analog channel]

0x7fffffd(4byte)

[Pulse channel]  
0x00000000(4byte)

[Wave calc channel]  
0x7ff0000000000001(8byte)

Binary data may include delimiter codes such as 0Ah or 0Dh.

If the PC software misinterprets such codes as data terminators, the PC will not handle the data correctly, so the PC software should always read the number of words specified by A.

A line feed code (LF or CR + LF) is not added at the end of the data.

When 1,000,000 points of analog channel data are recorded, the time required to send the command ":\MEMory:BDATA? 2000" 500 times is about 4 seconds.

(Running Microsoft Windows 10 Pro (22H2) on a Intel(R) i7-9700F 3.00GHz CPU and 16 GB RAM).

Please note that these values are for reference only, and depend on the actual data transfer speed of the LAN connection.

## Usage Conditions

---

Provided that the output point is lower than the amount of data stored.

Provided that stored data is present.

## Output hold data.(binary)

:MEMory:BFETch? ch\$

### Syntax

---

#### Query

:MEMory:BFETch? ch\$

#### Response

A

ch\$ = CH1\_1 to CH10\_30, PLS1, LOG, ALARM, W1 to W30

A = Binary data

### Details

---

Returns the value of the channel specified by ch\$ from the hold data captured by the :MEMory:GETReal.

The relationship between the returned data value and the measured value is same as:MEMory:BDATA?.

### Example

---

:MEMory:BFETch? CH1\_1

(Response) :MEMORY:BFETCH (binary data) (when headers are enabled)

### Note

---

Returns the value of the final storage data if any of the following operations are performed to capture the measurement data.

- Measurements
- Display Monitor Values
- Execute :MEMory:GETReal

If the above operation is not performed, 2147483645 is returned as a value indicating NO DATA.

### Usage Conditions

---

## Outputs real time data.(binary)

**:MEMory:BreAL? ch\$**

### Syntax

---

#### Query

:MEMory:BreAL? ch\$

#### Response

A

ch\$ = CH1\_1 to CH10\_30, PLS1, LOG, ALARM, W1 to W30

A = Binary data

### Details

---

Returns the final measurement of the channel designated by ch\$.

The relationship between the returned data value and the measured value is same as

:MEMory:BDATA?.

### Example

---

:MEMory:BreAL? CH1\_1

(Response) :MEMORY:BreAL (binary data) (when headers are enabled)

### Note

---

Returns the value of the final storage data if any of the following operations are performed to capture the measurement data.

- Measurements
- Display Monitor Values
- Execute :MEMory:GETReal

If the above operation is not performed, an error value corresponding to the specified channel is returned.

### Usage Conditions

---

Queries storage data existence for each channel.

`:MEMory:CHSTore? ch$`

## Syntax

---

### Query

`:MEMory:CHSTore? ch$`

### Response

`ch$,A$`

`ch$ = CH1_1 to CH10_30, PLS1, LOG, ALARM, W1 to W30`

`A$ = OFF, ON`

## Details

---

Returns the presence or absence of storage data of the channel designated by `ch$`.

## Example

---

`:MEMory:CHSTore? CH1_1`

(Response) `:MEMORY:CHSTORE CH1_1,ON` (when headers are enabled)

## Note

---

## Usage Conditions

---

Queries hold data existence for each channel.

`:MEMory:FCHSTore? ch$`

## Syntax

---

### Query

`:MEMory:FCHSTore? ch$`

### Response

`ch$,A$`

`ch$ = CH1_1 to CH10_30, PLS1, LOG, ALARM, W1 to W30`

`A$ = OFF, ON`

## Details

---

Returns the presence or absence of the hold data of the channel specified by `ch$`.

## Example

---

`:MEMory:FCHSTore? CH1_1`

(Response) `:MEMORY:FCHSTORE CH1_1,ON` (when headers are enabled)

## Note

---

## Usage Conditions

---

Captures hold data.

**:MEMory:GETReal**

## Syntax

---

### Command

:MEMory:GETReal

## Details

---

Causes the instrument to acquire real-time measurement values on all channels and retains it as hold data.

## Example

---

:MEMory:GETReal

## Note

---

It takes a few seconds to hold the data.

If the:MEMory: GETReal command is executed again while the command is running, an execution error will occur.

Use \*OPC or \*WAI if you want to wait for the hold completion and than run the following command.

## Usage Conditions

---

Queries the number of data samples stored.

**:MEMory:MAXPoint?**

## Syntax

---

### Query

:MEMory:MAXPoint?

### Response

A<NR1>

A = number of data samples (0 = no data stored)

## Details

---

Returns the number of data samples stored in the memory as a numerical value in NR1 format.

## Example

---

:MEMory:MAXPoint?

(Response) :MEMORY:MAXPOINT 2500 (when headers are enabled)

## Note

---

## Usage Conditions

---

Sets and queries the point in memory for output.

**:MEMory:POINT ch\$,A**

## Syntax

---

### Command

:MEMory:POINT ch\$,A

### Query

:MEMory:POINT?

### Response

ch\$,A<NR1>

ch\$ = CH1\_1 to CH10\_30, PLS1, LOG, ALARM, W1 to W30

A\$ = 0 to 268435456(maximum at only 1 channel)

## Details

---

Sets the output point in memory.

Returns the current output point in memory as an NR1 numerical value.

## Example

---

:MEMory:POINT CH1\_1,100

:MEMory:POINT?

(Response) :MEMORY:POINT CH1\_1,100 (when headers are enabled)

## Note

---

If there is no storage data, the output point cannot be set.

The value that can be set for A is smaller than the value returned by :MEMory:MAXPoint?.

## Usage Conditions

---

Outputs hold data.(ASCII) (ALL storage channels in the Module)

:MEMory:TAFETch? module\$

## Syntax

---

### Query

:MEMory:TAFETch? module\$

### Response

A1,A2,...<NR1>(Wave Calc Result = <NR3>)  
module\$ = MODULE1 to MODULE10, PLS&ALM, CALC1, CALC2  
A = -2147483648 to 2147483647(Analog)  
A = 0 to 2147483647(Count, Revolve)  
A = 0 to 1(Logic)  
A = 0 to 15(Alarm)  
A = Wave Calc Result(Wave Calc)

## Details

---

Returns the value of the channel containing the hold data of the specified module (data type) among the hold data captured by the :MEMory:GETReal.

The relationship between the returned data value and the measured value is same as :MEMory:ADATa?.

## Example

---

:MEMory:TAFETch? MODULE1  
(Response) :MEMORY:TAFETCH  
3176,3176,3176,3186,3186,3186,3186,3186,3198,3198,3186,3186,3186,3198,3198 (when headers are enabled)

## Note

---

If you have not run the :MEMory:GETReal before this command, a command error is occur.  
If there is no channel with hold data in the specified module, a command error will occur.

## Usage Conditions

---

## Outputs real time store channel.(ASCII)

`:MEMory:TARCH? module$`

### Syntax

---

#### Query

`:MEMory:TARCH? module$`

#### Response

`ch1$,ch2$,...`

`module$ = MODULE1 to MODULE10, PLS&ALM, CALC1, CALC2`

`ch$ = CH1_1 to CH10_30, PLS1, LOG, ALARM, W1 to W30`

### Details

---

Returns the measurement ON channel of the module (data type) designated by module\$.

### Example

---

`:MEMory:TARCH? MODULE1`

(Response) `:MEMORY:TARCH`

`CH1_1,CH1_2,CH1_3,CH1_4,CH1_5,CH1_6,CH1_7,CH1_8,CH1_9,CH1_10,CH1_11,CH1_12,CH1_13,CH1_14,CH1_15` (when headers are enabled)

### Note

---

### Usage Conditions

---

## Outputs real time data (ASCII) (ALL storage channels in the Module)

**:MEMory:TAREAL? module\$**

### Syntax

---

#### Query

:MEMory:TAREAL? module\$

#### Response

A1,A2,...<NR1>(Wave Calc Result = <NR3>)  
module\$ = MODULE1 to MODULE10, PLS&ALM, W1 to W30  
A = -2147483648 to 2147483647(Analog)  
A = 0 to 2147483647(Count, Revolve)  
A = 0 to 1(Logic)  
A = 0 to 15(Alarm)  
A = Wave Calc Result(Wave Calc)

### Details

---

Returns the last storage data value of the measurement ON channel of the module (data type) designated by module\$.

### Example

---

:MEMory:TAREAI? MODULE1  
(Response) :MEMORY:TAREAL  
3176,3176,3176,3186,3186,3186,3186,3186,3198,3198,3186,3186,3186,3198,3198 (when headers are enabled)

### Note

---

Returns the value of the final storage data if any of the following operations are performed to capture the measurement data.

- Measurements
- Display Monitor Values
- Execute :MEMory:GETReal

If the above operation is not performed, 2147483645 is returned as a value indicating NO DATA.

If there is no measurement ON channel in the specified module, NO\_STORAGE is returned.

### Usage Conditions

---

Queries storage data existence for each channel.(ALL storage channels in the Module)

`:MEMory:TCHSTore? module$`

## Syntax

---

### Query

`:MEMory:TCHSTore? module$`

### Response

ch1\$,ch2\$,...

module\$ = MODULE1 to MODULE10, PLS&ALM, W1 to W30

ch\$ = CH1\_1 to CH10\_30, PLS1, LOG, ALARM, W1 to W30

## Details

---

Returns the presence or absence of storage data of the module (data type) specified by module\$.

## Example

---

`:MEMory:TCHSTore? MODULE1`

(Response) `:MEMORY:TCHSTORE`

CH1\_1,CH1\_2,CH1\_3,CH1\_4,CH1\_5,CH1\_6,CH1\_7,CH1\_8,CH1\_9,CH1\_10,CH1\_11,CH1\_12,CH1\_13,CH1\_14,CH1\_15 (when headers are enabled)

## Note

---

Returns "MODULE\_NONE" if the specified module is not implemented.

Returns "NO DATA" if all measurements of the channels in the specified module are OFF.

## Usage Conditions

---

Queries hold data existence for each module.(ALL storage channels in the Module)

`:MEMory:TFCHSTore? module$`

## Syntax

---

### Query

`:MEMory:TFCHSTore? module$`

### Response

`ch1$,ch2$,...`

`module$ = MODULE1 to MODULE10, PLS&ALM, W1 to W30`

`ch$ = CH1_1 to CH10_30, PLS1, LOG, ALARM, W1 to W30`

## Details

---

Returns the presence or absence of the hold data of the module (data type) specified by module\$.

## Example

---

`:MEMory:TFCHSTore? MODULE1`

(Response) `:MEMORY:TFCHSTORE`

`CH1_1,CH1_2,CH1_3,CH1_4,CH1_5,CH1_6,CH1_7,CH1_8,CH1_9,CH1_10,CH1_11,CH1_12,CH1_13,CH1_14,CH1_15` (when headers are enabled)

## Note

---

Returns "MODULE\_NONE" if the specified module is not implemented.

Returns "NO DATA" if all measurements of the channels in the specified module are OFF.

## Usage Conditions

---

Queries the top of data samples stored.(when longer data is stored than the inside memory)

**:MEMory:TOPPoint?**

## Syntax

---

### Query

**:MEMory:TOPPoint?**

### Response

A<NR1>

A = Top data number (0 = no data stored)

## Details

---

Returns the top of data samples stored in the memory as a numerical value in NR1 format.

## Example

---

**:MEMory:TOPPoint?**

(Response) **:MEMORY:TOPPOINT 100** (when headers are enabled)

## Note

---

## Usage Conditions

---

## Output hold data.(measured value) (ALL storage channels in the Module)

`:MEMory:TVFETch? module$`

### Syntax

---

#### Query

`:MEMory:TVFETch? module$`

#### Response

A1,A2,...<NR3>

module\$ = MODULE1 to MODULE10, PLS&ALM, CALC1, CALC2

Ax = measured value

### Details

---

Returns the measured value of the channel containing the hold data of the specified module (data type) among the hold data captured by the `:MEMory:GETReal`.

See the description of `:MEMory:VDATA?` For the contents of the returned data.

### Example

---

`:MEMory:TVFETch? MODULE1`

(Response) `:MEMORY:TVFETCH`

`+1.00000E+03,+2.00000E+03,+3.00000E+03,+4.00000E+03,+5.00000E+03,+6.00000E+03,+7.00000E+03,+8.00000E+03,+1.00000E+03,+2.00000E+03,+3.00000E+03,+4.00000E+03,+5.00000E+03,+6.00000E+03,+7.00000E+03` (when headers are enabled)

### Note

---

If you have not run the `:MEMory:GETReal` before this command, a command error is occur.

If there is no channel with hold data in the specified module, a command error will occur.

### Usage Conditions

---

Outputs real time store channel.(measured value) (ALL storage channels in the Module)

`:MEMory:TVRCH? module$`

## Syntax

---

### Query

`:MEMory:TVRCH? module$`

### Response

ch1\$,ch2\$,...

module\$ = MODULE1 to MODULE10, PLS&ALM, CALC1, CALC2

ch\$ = CH1\_1 to CH10\_30, PLS1, LOG, ALARM, W1 to W30

## Details

---

Returns the measurement ON channel of the module (data type) designated by module\$. (Same as :MEMory:TARCH?)

## Example

---

`:MEMory:TVRCH? MODULE1`

(Response) `:MEMORY:TVRCH`

`CH1_1,CH1_2,CH1_3,CH1_4,CH1_5,CH1_6,CH1_7,CH1_8,CH1_9,CH1_10,CH1_11,CH1_12,CH1_13,CH1_14,CH1_15` (when headers are enabled)

## Note

---

## Usage Conditions

---

Outputs real time data.(measured value) (ALL storage channels in the Module)

`:MEMory:TVREAL? module$`

## Syntax

---

### Query

`:MEMory:TVREAL? module$`

### Response

A1,A2,...<NR3>

module\$ = MODULE1 to MODULE10, PLS&ALM, CALC1, CALC2

Ax = measured value

## Details

---

Returns the last storage data measured value of the measurement ON channel of the module (data type) designated by module\$.

See the description of :MEMory:VDATA? For the contents of the returned data.

## Example

---

`:MEMory:TVREAL? MODULE1`

(Response) `:MEMORY:TVREAL`

`+1.00000E+03,+2.00000E+03,+3.00000E+03,+4.00000E+03,+5.00000E+03,+6.00000E+03,+7.00000E+03,+8.00000E+03,+1.00000E+03,+2.00000E+03,+3.00000E+03,+4.00000E+03,+5.00000E+03,+6.00000E+03,+7.00000E+03` (when headers are enabled)

## Note

---

Returns the value of the final storage data if any of the following operations are performed to capture the measurement data.

- Measurements
- Display Monitor Values
- Execute `:MEMory:GETReal`

If the above operation is not performed, 9.99999E+99 is returned as an error value for any input type.

If there is no measurement ON channel in the specified module, NO\_STORAGE is returned.

## Usage Conditions

---

## Output voltage data from memory.

**:MEMory:VDAa? A**

### Syntax

---

#### Query

:MEMory:VDAa? A

#### Response

B1,B2,...<NR3>

A = 1 to 1000

Bi = measured value

The number of decimal places depends on the specified channel.

[Analog channel]

7 significant digits

[Pulse channel]

10 significant digits

[Wave calc channel]

12 significant digits

[Logic]

Integer(1bit)

[Alarm]

Integer(4bit)

### Details

---

The number of measured values specified by A are output as voltage values from the memory channel and point set by the :MEMory:POINT.

The output point is incremented by the number of data values.

When scaling, the scaled values are output.

For Analog channel, Pulse channel, and Wave calc channels, the decimal point position is variable so that the exponent part is a multiple of 3.

[Example]

+1.234567E+03,+12.34567E+03,+123.4567E+03,+1.234567E+06

### Example

---

:MEMory:POINT CH1\_1,0

:MEMory:VDAa? 2

(Response) :MEMORY:VDATA +5.000000E+03,+4.000000E+03 (when headers are enabled)

## Note

---

Returns 9.99999E+99 as a error value if no storage data exists for the specified channel/output point. When 1,000,000 points of analog channel data are recorded, the time required to send the command ":MEMory:VDAa? 2000" 500 times is about 70 seconds.

(Running Microsoft Windows 10 Pro (22H2) on a Intel(R) i7-9700F 3.00GHz CPU and 16 GB RAM). Please note that these values are for reference only, and depend on the actual data transfer speed of the LAN connection.

## Usage Conditions

---

The output point must be set to :MEMory:POINT.

Provided that the output point is lower than the amount of data stored.

Provided that stored data is present.

## Output hold data.(measured value)

`:MEMory:VFETch? ch$`

### Syntax

---

#### Query

`:MEMory:VFETch? ch$`

#### Response

A<NR3>  
ch\$ = CH1\_1 to CH10\_30, PLS1, LOG, ALARM, W1 to W30  
A = measured value

### Details

---

Returns the measured value of the channel specified by ch\$ from the hold data captured by the :MEMory:GETReal.

The relationship between the returned data value and the measured value is same as :MEMory:VDATA?.

### Example

---

`:MEMory:VFETch? CH1_1`  
(Response) :MEMORY:VFETCH +1.23000E+03 (when headers are enabled)

### Note

---

If you have not run the :MEMory:GETReal before this command, it returns 9.99999E+99 as a error value.

### Usage Conditions

---

Outputs real time data.(measured value)

`:MEMory:VREAL? ch$`

## Syntax

---

### Query

`:MEMory:VREAL? ch$`

### Response

A<NR3>

ch\$ = CH1\_1 to CH10\_30, PLS1, LOG, ALARM, W1 to W30

A = measured value

## Details

---

Returns the final measurement of the channel designated by ch\$.

## Example

---

`:MEMory:VREAL? CH1_1`

(Response) `:MEMORY:VREAL +1.23000E+03` (when headers are enabled)

## Note

---

Returns the value of the final storage data if any of the following operations are performed to capture the measurement data.

- Measurements
- Display Monitor Values
- Execute `:MEMory:GETReal`

If the above operation is not performed, `9.99999E+99` is returned as an error value.

## Usage Conditions

---

## 3.12 Screen(DISPlay)

Insert the event mark.

`:DISPlay:MARK`

### Syntax

---

#### Command

`:DISPlay:MARK`

### Details

---

Insert the event mark.

### Example

---

`:DISPlay:MARK`

### Note

---

### Usage Conditions

---

Insert event mark at only start.

Queries the event mark num.

`:DISPlay:MARK?`

## Syntax

---

### Query

`:DISPlay:MARK?`

### Response

A<NR1>

A = 0 to 1000

## Details

---

Queries the event mark num as an NR1 numerical value.

## Example

---

`:DISPlay:MARK?`

(Response) `:DISPLAY:MARK 10` (when headers are enabled)

## Note

---

## Usage Conditions

---

Queries the data num at the event mark designated.

`:DISPlay:MARKJump? A`

## Syntax

---

### Query

`:DISPlay:MARKJump? A`

### Response

`A<NR1>,B<NR1>`

A = 1 to event mark num

B = data num

## Details

---

Queries the data num at the event mark designated as an NR1 numerical value.

## Example

---

`:DISPlay:MARKJump? 10`

(Response) `:DISPLAY:MARKJUMP 10,500` (when headers are enabled)

## Note

---

## Usage Conditions

---

## 3.13 Calculations(CALCulate)

Sets and queries the numerical calculation.

:CALCulate:MEASure A\$

### Syntax

---

#### Command

:CALCulate:MEASure A\$

#### Query

:CALCulate:MEASure?

#### Response

A\$

A\$ = OFF, ON

### Details

---

Sets the numerical calculation.

Returns the current setting of the numerical calculation as character data.

OFF: Disabled

ON: Enabled

### Example

---

:CALCulate:MEASure ON

:CALCulate:MEASure?

(Response) :CALCULATE:MEASURE ON (when headers are enabled)

### Note

---

### Usage Conditions

---

Queries result of a numerical calculation.

`:CALCulate:MEAS:ANSWer? no$,ch$`

## Syntax

---

### Query

`:CALCulate:MEAS:ANSWer? no$,ch$`

### Response

`no$,ch$,A<NR3>` (11 decimal places)

`no$` = NO1 to NO10

`ch$` = CH1\_1 to CH10\_30, PLS1, W1 to W30

`A` = calculation result

## Details

---

Returns the calculation result for the numerical calculation number and channel specified by `no$`, `ch$`.

## Example

---

`:CALCulate:MEAS:ANSWer? NO1,CH1_1`

(Response) `:CALCULATE:MEAS:ANSWER NO1,CH1_1,+1.23456789012E-03` (when headers are enabled)

## Note

---

Returns `A=NONE` for the following conditions:

- Numerical calculation is OFF
- Numerical calculation item for the specified numerical calculation number is OFF
- Calculation result does not exist or is not obtained

It is also possible to query using conventional commands. (`:CALCulate:ANSWer? no$,ch$`)

## Usage Conditions

---

Sets and queries the calculation file division.

**:CALCulate:MEAS:FILE A\$**

## Syntax

---

### Command

:CALCulate:MEAS:FILE A\$

### Query

:CALCulate:MEAS:FILE?

### Response

A\$

A\$ = OFF, ON

## Details

---

Sets the calculation file division.

Returns the current setting of the calculation file division as character data.

OFF: Single file

ON: Split save

## Example

---

:CALCulate:MEAS:FILE ON

:CALCulate:MEAS:FILE?

(Response) :CALCULATE:MEAS:FILE ON (when headers are enabled)

## Note

---

It is also possible to set and query using the following conventional commands.

:CALCulate:MEASFile A\$ / :CALCulate:MEASFile?

## Usage Conditions

---

Sets and queries the accumulation/integrations method.

`:CALCulate:MEAS:INTegra no$,A$`

## Syntax

---

### Command

`:CALCulate:MEAS:INTegra no$,A$`

### Query

`:CALCulate:MEAS:INTegra? no$`

### Response

no\$ = NO1 to NO10

A\$ = TOTAL, POSitive, NEGative, ABSolute

## Details

---

Sets the accumulation/integrations method.

Returns the current accumulation/integrations method as character data.

TOTAL: Add all values

POSitive: Add only positive values

NEGative: Add only negative values

ABSolute: Add absolute values

## Example

---

`:CALCulate:MEAS:INTegra NO1,TOTAL`

`:CALCulate:MEAS:INTegra? NO1`

(Response) `:CALCULATE:MEAS:INTEGRA NO1,TOTAL` (when headers are enabled)

## Note

---

## Usage Conditions

---

Only the following types of calculation can be set.

Accumulation (ACC)

Integration (INT)

Sets and queries the time split calculation kind.

**:CALCulate:MEAS:KIND A\$**

## Syntax

---

### Command

:CALCulate:MEAS:KIND A\$

### Query

:CALCulate:MEAS:KIND?

### Response

A\$

A\$ = OFF, DIVide, ONTIME

## Details

---

Sets the time split calculation kind.

Returns the current setting of the time split calculation kind as character data.

OFF: No split

DIVide: With split

ONTIME: With split (Fix time)

## Example

---

:CALCulate:MEAS:KIND DIVide

:CALCulate:MEAS:KIND?

(Response) :CALCULATE:MEAS:KIND DIVIDE (when headers are enabled)

## Note

---

It is also possible to set and query using the following conventional commands.

:CALCulate:MEASKind A\$ / :CALCulate:MEASKind?

## Usage Conditions

---

Sets and queries the split calculation division length.

`:CALCulate:MEAS:LEN day,hour,min`

## Syntax

---

### Command

`:CALCulate:MEAS:LEN day,hour,min`

### Query

`:CALCulate:MEAS:LEN?`

### Response

`day<NR1>,hour<NR1>,min<NR1>`

day = 0 to 30(day)

hour = 0 to 23(hour)

min = 0 to 59(min)

## Details

---

Sets the split calculation division length to a numerical value.

Returns the currently set value of the split calculation division length as an NR1 numerical value.

## Example

---

`:CALCulate:MEAS:LEN 0,1,30`

`:CALCulate:MEAS:LEN?`

(Response) `:CALCULATE:MEAS:LEN 0,1,30` (when headers are enabled)

## Note

---

The setting of division length might be limited by a set value of interval.

It is also possible to set and query using the following conventional commands.

`:CALCulate:MEASLen day,hour,min / :CALCulate:MEASLen?`

## Usage Conditions

---

Sets and queries the thresholds for numerical calculation.

`:CALCulate:MEAS:LEVEL ch$,A`

## Syntax

---

### Command

`:CALCulate:MEAS:LEVEL ch$,A`

### Query

`:CALCulate:MEAS:LEVEL? ch$`

### Response

ch\$,A<NR3> (4 decimal places)  
ch\$ = CH1\_1 to CH10\_30, PLS1, W1 to W30  
A = -9.9999E+29 to +9.9999E+29

## Details

---

Sets the thresholds for numerical calculation.  
Returns the current setting of numerical calculation threshold as an NR3 numerical value.

## Example

---

`:CALCulate:MEAS:LEVEL CH1_1,0.123`  
`:CALCulate:MEAS:LEVEL? CH1_1`  
(Response) `:CALCULATE:MEAS:LEVEL CH1_1,+1.2300E-01` (when headers are enabled)

## Note

---

Only the following types of calculation can be set.  
Usage ratio: (OPE)  
ON time: (ONT)  
OFF time: (OFFT)  
ON count: (ONC)  
OFF count: (OFFC)  
If you enter a value greater than the configurable limit, the maximum value is set.  
If you enter a value less than the configurable lower limit, the minimum value is set.

## Usage Conditions

---

Sets and queries the reference time for time-division calculations.

**:CALCulate:MEAS:REG hour,min**

## Syntax

---

### Command

:CALCulate:MEAS:REG hour,min

### Query

:CALCulate:MEAS:REG?

### Response

hour<NR1>,min<NR1>

hour = 0 to 23(hour)

min = 0 to 59(min)

## Details

---

Sets the separation time for time-separated calculation.

Returns the separation time setting for the current time separation operation.

## Example

---

:CALCulate:MEAS:REG 1,30

:CALCulate:MEAS:REG?

(Response) :CALCULATE:MEAS:REG 1,30 (when headers are enabled)

## Note

---

It is also possible to set and query using the following conventional commands.

:CALCulate:MEASReg hour,min / :CALCulate:MEASReg?

## Usage Conditions

---

Sets and queries the numerical calculations item.

**:CALCulate:MEAS:SET no\$,A\$**

## Syntax

---

### Command

:CALCulate:MEAS:SET no\$,A\$

### Query

:CALCulate:MEAS:SET? no\$

### Response

no\$,A\$

no\$ = NO1 to NO10

A\$ = OFF, AVE, PP, MAX, MIN, MAXT, MINT, ACC, INT, OPE, ONT, OFFT, ONC, OFFC

## Details

---

Sets the calculation item of the numerical calculation designated by no\$.

Returns the calculation item of the numerical calculation designated by no\$ as character data.

OFF: No calculation performed

AVE: Average value

PP: Peak-to-peak value

MAX: Maximum value

MIN: Minimum value

MAXT: Time to maximum value

MINT: Time to minimum value

ACC: Accumulation

INT: Integration

OPE: Usage ratio

ONT: ON time

OFFT: OFF time

ONC: ON count

OFFC: OFF count

## Example

---

:CALCulate:MEAS:SET NO1,AVE

:CALCulate:MEAS:SET? NO1

(Response) :CALCULATE:MEAS:SET NO1,AVE (when headers are enabled)

## Note

---

It is also possible to set and query using the following conventional commands.

:CALCulate:MEASSet no\$,A\$ / :CALCulate:MEASSet? no\$

## Usage Conditions

---

Sets and queries the calculation target channel.

**:CALCulate:MEAS:TARGet no\$,ch\$**

## Syntax

---

### Command

:CALCulate:MEAS:TARGet no\$,ch\$

### Query

:CALCulate:MEAS:TARGet? no\$

### Response

no\$,ch\$

no\$ = NO1 to NO10

ch\$ = ALL, CH1\_1 to CH10\_30, PLS1, W1 to W30

## Details

---

Sets the numerical calculation target channel designated by no\$.

Returns the numerical calculation target channel designated by no\$ as character data.

## Example

---

:CALCulate:MEAS:TARGet NO1,CH1\_1

:CALCulate:MEAS:TARGet? NO1

(Response) :CALCULATE:MEAS:TARGET NO1,CH1\_1 (when headers are enabled)

## Note

---

## Usage Conditions

---

## Sets and queries the time split calculation division time

**:CALCulate:MEAS:TIME A**

### Syntax

---

#### Command

:CALCulate:MEAS:TIME A

#### Query

:CALCulate:MEAS:TIME?

#### Response

A<NR1>

A = 1, 2, 5, 10, 15, 20, 30, 60, 120, 180, 240, 360, 480, 720, 1440 (unit min)

### Details

---

Sets the division time for time-separated operations.

Returns the split time of the current time-separated operation.

### Example

---

:CALCulate:MEAS:TIME 1

:CALCulate:MEAS:TIME?

(Response) :CALCulate:MEAS:TIME 1 (when headers are enabled)

### Note

---

If you specify a value that is not in the setting and there is a time longer than the value you are trying to set, it will be set to the nearest time.

It is also possible to set and query using the following conventional commands.

:CALCulate:MEAS:TIME A / :CALCulate:MEAS:TIME? A

### Usage Conditions

---

Sets and queries the wave calculation kind.

`:CALCulate:WAVE:KIND w$,A$`

## Syntax

---

### Command

`:CALCulate:WAVE:KIND w$,A$`

### Query

`:CALCulate:WAVE:KIND? w$`

### Response

w\$,A\$

w\$ = W1 to W30

A\$ = OPE, SUM, AVE, MOV, INT

## Details

---

Sets the wave calculation kind designated by w\$.

Returns the wave calculation kind designated by w\$ as character data.

OPE: Four arithmetic

SUM: Accumulation

AVE: Simple average

MOV: Moving average

INT: Integration

## Example

---

`:CALCulate:WAVE:KIND W1,OPE`

`:CALCulate:WAVE:KIND? W1`

(Response) `:CALCULATE:WAVE:KIND W1,OPE` (when headers are enabled)

## Note

---

It is also possible to set and query using the following conventional commands.

`:CALCulate:HTKIND w$,A$ / :CALCulate:HTKIND? w$`

## Usage Conditions

---

Sets and queries the wave calculation unit.

`:CALCulate:WAVE:STR w$,"A$"`

## Syntax

### Command

`:CALCulate:WAVE:STR w$,"A$"`

### Query

`:CALCulate:WAVE:STR? w$`

### Response

w\$,"A\$"

w\$ = W1 to W30

A\$ = scaling unit (up to 7 characters)

## Details

Sets the wave calculation unit for the channel designated by w\$.

Returns the current wave calculation unit for the channel designated by w\$ as character data.

Single quotation marks(') can be used instead of double quotation marks(").

Entry of the special characters is as follows.

PC	$\wedge 2$	$\wedge 3$	$\sim u$	$\sim o$	$\sim e$	$\sim c$	$\sim +$	$\sim ,$	$\sim ;$	$\wedge \wedge$	$\sim \sim$
LR8101 LR8102	2	3	$\mu$	$\Omega$	$\epsilon$	o	$\pm$	'	"	$\wedge$	$\sim$

## Example

`:CALCulate:WAVE:STR W1,"mA"`

`:CALCulate:WAVE:STR? W1`

(Response) `:CALCULATE:WAVE:STR W1,"mA"` (when headers are enabled)

## Note

Characters exceeding the maximum number of characters are not set.

It is also possible to set and query using the following conventional commands.

`:CALCulate:WVSTR w$,"A$"` / `:CALCulate:WVSTR? w$`

## Usage Conditions

Wave calculate : Sets and queries the four arithmetic coefficient.

:CALCulate:WAVE:ARITHmetic:COEF:A w\$,A (coefficient A)  
:CALCulate:WAVE:ARITHmetic:COEF:B w\$,B (coefficient B)  
:CALCulate:WAVE:ARITHmetic:COEF:C w\$,C (coefficient C)  
:CALCulate:WAVE:ARITHmetic:COEF:D w\$,D (coefficient D)  
:CALCulate:WAVE:ARITHmetic:COEF:E w\$,E (coefficient E)

## Syntax

---

### Command

:CALCulate:WAVE:ARITHmetic:COEF:A w\$,A (coefficient A)  
:CALCulate:WAVE:ARITHmetic:COEF:B w\$,B (coefficient B)  
:CALCulate:WAVE:ARITHmetic:COEF:C w\$,C (coefficient C)  
:CALCulate:WAVE:ARITHmetic:COEF:D w\$,D (coefficient D)  
:CALCulate:WAVE:ARITHmetic:COEF:E w\$,E (coefficient E)

### Query

:CALCulate:WAVE:ARITHmetic:COEF:A? w\$ (coefficient A)  
:CALCulate:WAVE:ARITHmetic:COEF:B? w\$ (coefficient B)  
:CALCulate:WAVE:ARITHmetic:COEF:C? w\$ (coefficient C)  
:CALCulate:WAVE:ARITHmetic:COEF:D? w\$ (coefficient D)  
:CALCulate:WAVE:ARITHmetic:COEF:E? w\$ (coefficient E)

### Response

w\$,A<NR3> (4 decimal places)  
w\$ = W1 to W30  
A = -9.9999E+29 to 9.9999E+29

## Details

---

Sets the four arithmetic coefficient of the channel specified by w\$.  
Returns the four arithmetic coefficient of the channel specified by w\$ as an NR3 number.

## Example

---

:CALCulate:WAVE:ARITHmetic:COEF:A W1,1  
:CALCulate:WAVE:ARITHmetic:COEF:A? W1  
(Response) :CALCULATE:WAVE:ARITHMETIC:COEF:A W1,+1.0000E+00 (when headers are enabled)

## Note

---

It is also possible to set and query using the following conventional commands.

:CALCulate:WVCOE1 w\$,A / :CALCulate:WVCOE1? w\$ (coefficient A)

:CALCulate:WVCOE1 w\$,B / :CALCulate:WVCOE1? w\$ (coefficient B)

:CALCulate:WVCOE1 w\$,E / :CALCulate:WVCOE1? w\$ (coefficient E)

## Usage Conditions

---

Wave calculate : Sets and queries the four arithmetic operator.

:CALCulate:WAVE:ARITHmetic:OPERator:A w\$,A\$ (Operator A)  
:CALCulate:WAVE:ARITHmetic:OPERator:B w\$,A\$ (Operator B)  
:CALCulate:WAVE:ARITHmetic:OPERator:C w\$,A\$ (Operator C)  
:CALCulate:WAVE:ARITHmetic:OPERator:D w\$,A\$ (Operator D)

## Syntax

---

### Command

:CALCulate:WAVE:ARITHmetic:OPERator:A w\$,A\$ (Operator A)  
:CALCulate:WAVE:ARITHmetic:OPERator:B w\$,A\$ (Operator B)  
:CALCulate:WAVE:ARITHmetic:OPERator:C w\$,A\$ (Operator C)  
:CALCulate:WAVE:ARITHmetic:OPERator:D w\$,A\$ (Operator D)

### Query

:CALCulate:WAVE:ARITHmetic:OPERator:A? w\$ (Operator A)  
:CALCulate:WAVE:ARITHmetic:OPERator:B? w\$ (Operator B)  
:CALCulate:WAVE:ARITHmetic:OPERator:C? w\$ (Operator C)  
:CALCulate:WAVE:ARITHmetic:OPERator:D? w\$ (Operator D)

### Response

w\$,A\$  
w\$ = W1 to W30  
A\$ = OFF, PLUS, MINUS, MULTI, DIV (Operator A, B, C)  
A\$ = OFF, PLUS, MINUS, MULTI, DIV, EXP (Operator D)

## Details

---

Sets the four arithmetic operator of the channel specified by w\$.  
Returns the four arithmetic operator of the channel specified by w\$ as character data.

## Example

---

:CALCulate:WAVE:ARITHmetic:OPERator:A W1,PLUS  
:CALCulate:WAVE:ARITHmetic:OPERator:A? W1  
(Response) :CALCULATE:WAVE:ARITHMETIC:OPERATOR:A W1,PLUS (when headers are enabled)

## Note

---

It is also possible to set and query using the following conventional commands.  
:CALCulate:WVKIND w\$,A\$ / :CALCulate:WVKIND? w\$ (Operator A)

## Usage Conditions

---

Operator B can be set if Operator A is not OFF.

Operator C can be set if operators A and B are not OFF.

Wave calculate : Sets and queries the moving average point.

`:CALCulate:WAVE:MOVE:POINT w$,A`

## Syntax

---

### Command

`:CALCulate:WAVE:MOVE:POINT w$,A`

### Query

`:CALCulate:WAVE:MOVE:POINT? w$`

### Response

`w$,A<NR1>`  
`w$ = W1 to W30`  
`A = 1 to 600 (point)`

## Details

---

Sets the moving average point of the channel specified by w\$.  
Returns the moving average point of the channel specified by w\$ as an NR1 number.

## Example

---

`:CALCulate:WAVE:MOVE:POINT W1,10`  
`:CALCulate:WAVE:MOVE:POINT? W1`  
(Response) `:CALCULATE:WAVE:MOVE:POINT W1,10` (when headers are enabled)

## Note

---

It is also possible to set and query using the following conventional commands.  
`:CALCulate:HTMVPoint w$,A / :CALCulate:HTMVPoint? w$`

## Usage Conditions

---

Wave calculate : Sets and queries the reset reference time for integral, simple average.

`:CALCulate:WAVE:RESet:BASE w$,hour,min`

## Syntax

---

### Command

`:CALCulate:WAVE:RESet:BASE w$,hour,min`

### Query

`:CALCulate:WAVE:RESet:BASE? w$`

### Response

`w$,hour<NR1>,min<NR1>`

`w$ = W1 to W30`

`hour = 0 to 23(hour)`

`min = 0 to 59(min)`

## Details

---

Sets the reset reference time of the channel specified by w\$.

Returns the reset reference time of the channel specified by w\$ as an NR1 number.

## Example

---

`:CALCulate:WAVE:RESet:BASE W1,0,0`

`:CALCulate:WAVE:RESet:BASE? W1`

(Response) `:CALCULATE:WAVE:RESET:BASE W1,0,0` (when headers are enabled)

## Note

---

It is also possible to set and query using the following conventional commands.

`:CALCulate:HTBASE w$,hour,min / :CALCulate:HTBASE? w$`

## Usage Conditions

---

Wave calculate : Sets and queries the reset interval for integral, simple average.

**:CALCulate:WAVE:RESet:INT w\$,day,hour,min**

## Syntax

---

### Command

**:CALCulate:WAVE:RESet:INT w\$,day,hour,min**

### Query

**:CALCulate:WAVE:RESet:INT? w\$**

### Response

w\$,day<NR1>,hour<NR1>,min<NR1>

w\$ = W1 to W30

day = 0 to 30(day)

hour = 0 to 23(hour)

min = 0 to 59(min)

## Details

---

Sets the reset interval of the channel specified by w\$.

Returns the reset interval of the channel specified by w\$ as an NR1 number.

## Example

---

**:CALCulate:WAVE:RESet:INT W1,0,0,1**

**:CALCulate:WAVE:RESet:INT? W1**

(Response) **:CALCULATE:WAVE:RESET:INT W1,0,0,1** (when headers are enabled)

## Note

---

It is also possible to set and query using the following conventional commands.

**:CALCulate:HTINT w\$,day,hour,min / :CALCulate:HTINT? w\$**

## Usage Conditions

---

Wave calculate : Sets and queries the reset timing for integral, simple average.

`:CALCulate:WAVE:RESet:KIND w$,A$`

## Syntax

---

### Command

`:CALCulate:WAVE:RESet:KIND w$,A$`

### Query

`:CALCulate:WAVE:RESet:KIND? w$`

### Response

w\$,A\$

w\$ = W1 to W30

A\$ = OFF, TRIG

## Details

---

Sets the reset timing of the channel specified by w\$.

Returns the reset timing of the channel specified by w\$ as character data.

OFF

TRIG: Trigger position

## Example

---

`:CALCulate:WAVE:RESet:KIND W1,OFF`

`:CALCulate:WAVE:RESet:KIND? W1`

(Response) `:CALCULATE:WAVE:RESET:KIND W1,OFF` (when headers are enabled)

## Note

---

It is also possible to set and query using the following conventional commands.

`:CALCulate:HTRESet w$,A$ / :CALCulate:HTRESet? w$`

## Usage Conditions

---

Wave calculate : Sets and queries the reset time for integral, simple average.

`:CALCulate:WAVE:RESet:TIME w$,A$`

## Syntax

---

### Command

`:CALCulate:WAVE:RESet:TIME w$,A$`

### Query

`:CALCulate:WAVE:RESet:TIME? w$`

### Response

w\$,A\$

w\$ = W1 to W30

A\$ = OFF, ON, ONTIME

## Details

---

Sets the reset time of the channel specified by w\$.

Returns the reset time of the channel specified by w\$ as character data.

OFF: No split

ON: With split

ONTIME: With split (Fix time)

## Example

---

`:CALCulate:WAVE:RESet:TIME W1,OFF`

`:CALCulate:WAVE:RESet:TIME? W1`

(Response) `:CALCULATE:WAVE:RESET:TIME W1,OFF` (when headers are enabled)

## Note

---

It is also possible to set and query using the following conventional commands.

`:CALCulate:HTRSTTime w$,A$ / :CALCulate:HTRSTTime? w$`

## Usage Conditions

---

Sets and queries the wave calculation source.

```
:CALCulate:WAVE:SOURce:SR1 w$,ch$ (Source A)
:CALCulate:WAVE:SOURce:SR2 w$,ch$ (Source B)
:CALCulate:WAVE:SOURce:SR3 w$,ch$ (Source C)
:CALCulate:WAVE:SOURce:SR4 w$,ch$ (Source D)
```

## Syntax

---

### Command

```
:CALCulate:WAVE:SOURce:SR1 w$,ch$ (Source A)
:CALCulate:WAVE:SOURce:SR2 w$,ch$ (Source B)
:CALCulate:WAVE:SOURce:SR3 w$,ch$ (Source C)
:CALCulate:WAVE:SOURce:SR4 w$,ch$ (Source D)
```

### Query

```
:CALCulate:WAVE:SOURce:SR1? w$ (Source A)
:CALCulate:WAVE:SOURce:SR2? w$ (Source B)
:CALCulate:WAVE:SOURce:SR3? w$ (Source C)
:CALCulate:WAVE:SOURce:SR4? w$ (Source D)
```

### Response

```
w$,ch$
w$ = W1 to W30
ch$ = CH1_1 to CH10_30, PLS1, W1 to W30
```

## Details

---

Sets the wave calculation source of the channel specified by w\$.  
Returns the wave calculation source of the channel specified by w\$ as character data.

## Example

---

```
:CALCulate:WAVE:SOURce:SR1 W1,CH1_1
:CALCulate:WAVE:SOURce:SR1? W1
(Response) :CALCULATE:WAVE:SOURCE:SR1 W1,CH1_1 (when headers are enabled)
```

## Note

---

It is also possible to set and query using the following conventional commands.

```
:CALCulate:WVSRc1 w$,ch$ / :CALCulate:WVSRc1? w$ (Source A)
:CALCulate:WVSRc2 w$,ch$ / :CALCulate:WVSRc2? w$ (Source B)
```

## Usage Conditions

---

## 3.14 Error(ERRor)

Queries the error bit.

```
:ERRor:BIT:ERRor?
```

### Syntax

---

#### Query

```
:ERRor:BIT:ERRor?
```

#### Response

```
errBit
```

### Details

---

Returns a hexadecimal value summarizing error information that has occurred since startup.

### Example

---

```
:ERRor:BIT:ERRor?
```

```
(Response) :ERROR:BIT:ERROR 40000 (when headers are enabled)
```

### Note

---

Errors immediately after they occur may not be reflected in the response to this command.  
Please check again after a short time.

### Usage Conditions

---

Queries the warning bit.

```
:ERRor:BIT:WARNing?
```

## Syntax

---

### Query

```
:ERRor:BIT:WARNing?
```

### Response

```
warnBit
```

## Details

---

Returns the bits of warning information that have occurred since startup or since the execution of :ERRor:BIT:WARNing:CLEAr, as a hexadecimal value.

## Example

---

```
:ERRor:BIT:WARNing?
```

(Response) :ERROR:BIT:WARNING 800 (when headers are enabled)

## Note

---

Warnings immediately after they occur may not be reflected in the response to this command. Please check again after a short time.

## Usage Conditions

---

Clears the warning bit.

`:ERRor:BIT:WARNing:CLEAr`

## Syntax

---

### Command

`:ERRor:BIT:WARNing:CLEAr`

## Details

---

Clear the bit of warning information.

## Example

---

`:ERRor:BIT:WARNing:CLEAr`

## Note

---

## Usage Conditions

---

Queries the error log.

```
:ERRor:LOG:ERRor? no$
```

## Syntax

---

### Query

```
:ERRor:LOG:ERRor? no$
```

### Response

```
error$ :Error Information String  
no$ = 1 to 50
```

## Details

---

Outputs the contents of the specified error log (time of occurrence and error number).  
no\$ = 1 is the error that occurred most recently.  
Up to 50 errors are stored in the error log.

## Example

---

```
:ERRor:LOG:ERRor? 1  
(Response) :ERROR:LOG:ERROR 2023/02/28 16:40:47 - ERR_SY01 (when headers are enabled)
```

## Note

---

Errors immediately after they occur may not be reflected in the response to this command.  
Please check again after a short time.

## Usage Conditions

---

Clears the error log.

```
:ERRor:LOG:ERRor:CLEAr
```

## Syntax

---

### Command

```
:ERRor:LOG:ERRor:CLEAr
```

## Details

---

Clears the error log.

## Example

---

```
:ERRor:LOG:ERRor:CLEAr
```

## Note

---

## Usage Conditions

---

Queries the warning log.

```
:ERRor:LOG:WARNing? no$
```

## Syntax

---

### Query

```
:ERRor:LOG:WARNing? no$
```

### Response

```
warn$ :Warning Information String  
no$ = 1 to 50
```

## Details

---

Outputs the contents of the specified warning log (time of occurrence and warning number).  
no\$ = 1 is the warning that occurred most recently.  
Up to 50 warnings are stored in the warning log.

## Example

---

```
:ERRor:LOG:WARNing? 1  
(Response) :ERROR:LOG:WARNING 2023/05/18 14:55:54 - WARN_SY01 (when headers are  
enabled)
```

## Note

---

warnings immediately after they occur may not be reflected in the response to this command.  
Please check again after a short time.

## Usage Conditions

---

Clears the warning log.

`:ERRor:LOG:WARNIng:CLEAr`

## Syntax

---

### Command

`:ERRor:LOG:WARNIng:CLEAr`

## Details

---

Clears the warning log.

## Example

---

`:ERRor:LOG:WARNIng:CLEAr`

## Note

---

## Usage Conditions

---

## 3.15 Media(MEDia)

Queries the reading status of a configuration condition file.

```
:MEDia:SD:FINFo:SET? "fname$"  
:MEDia:USB:FINFo:SET? "fname$"
```

### Syntax

---

#### Query

```
:MEDia:SD:FINFo:SET? "fname$"  
:MEDia:USB:FINFo:SET? "fname$"
```

#### Response

```
ans$, modules$  
ans$ = OK, NG_MODEL, NG_MODULE, BUSY  
modules$ = 0, 1, 3
```

### Details

---

Returns the reading status of a configuration condition file.

OK: Ready to read.

NG\_MODEL: Not readable because it is a configuration file for a different model.

NG\_MODULE: Not readable because it is a configuration file for a different module.

BUSY: Information cannot be obtained because the file is being processed.

The module type is returned as a 10-digit number in the order of module1,2,... from left to right.

0: No module.

1: M7100 15ch Voltage and temperature module.

3: M7102 30ch Voltage and temperature module.

### Example

---

```
:MEDia:SD:FINFo:SET? "CONF0001.SET"  
(Response) :MEDia:SD:FINFo:SET OK,1300000000 (when headers are enabled)
```

### Note

---

### Usage Conditions

---

Queries the free media capacity.

:MEDia:SD:FREE?  
:MEDia:USB:FREE?

## Syntax

---

### Query

:MEDia:SD:FREE?  
:MEDia:USB:FREE?

### Response

A<NR1>

## Details

---

Returns the free media capacity.

## Example

---

:MEDia:SD:FREE?  
(Response) :MEDIA:SD:FREE 100 (when headers are enabled)

## Note

---

## Usage Conditions

---

## Media Format.

:MEDia:SD:FORMat?  
:MEDia:USB:FORMat?

### Syntax

---

#### Query

:MEDia:SD:FORMat?  
:MEDia:USB:FORMat?

#### Response

A\$

### Details

---

Formats the media.  
FAIL: Formatting fails  
SUCCESS: Formatting succeeds

### Example

---

:MEDia:SD:FORMat?  
(Response) :MEDIA:SD:FORMAT SUCCESS (when headers are enabled)

### Note

---

### Usage Conditions

---

All data in the media will be erased.

Queries a list of files in /HIOKI/LR8100/CONFIG for each media.

:MEDia:SD:FLISt:SET?  
:MEDia:USB:FLISt:SET?

## Syntax

---

### Query

:MEDia:SD:FLISt:SET?  
:MEDia:USB:FLISt:SET?

### Response

A1\$,A2\$,...  
A\$ = Filename

## Details

---

Returns the filename in /HIOKI/LR8100/CONFIG for each media.

## Example

---

:MEDia:SD:FLISt:SET?  
(Response) :MEDIA:SD:FLISt:SET CONF0002.SET,CONF0001.SET (when headers are enabled)

## Note

---

## Usage Conditions

---

The maximum number of lists that can be obtained is 100.

Sets the file name and the loading options and queries the loading state

```
:MEDia:SD:LOAD:SET "fname$",option  
:MEDia:USB:LOAD:SET "fname$",option
```

## Syntax

---

### Command

```
:MEDia:SD:LOAD:SET "fname$",option  
:MEDia:USB:LOAD:SET "fname$",option
```

### Query

```
:MEDia:SD:LOAD:SET?  
:MEDia:USB:LOAD:SET?
```

### Response

"fname\$" = Setting condition file name to be read (xxxx.SET)  
option = 0 to 3 (loading options)  
A\$ = NONE, EXECUTING\_LOAD\_(filename), SUCCESS\_LOAD\_(filename),  
FAIL\_LOAD\_(filename)

## Details

---

Sets the file name and the loading options.

option = 0: Measurement settings

option = 1: Measurement settings + External terminal

option = 2: Measurement settings + Communication settings

option = 3: Measurement settings + External terminal + Communication settings

Returns the file loading state.

NONE: Before loading.

EXECUTING\_LOAD\_(filename): Loading in progress.

SUCCESS\_LOAD\_(filename): Succeeded in loading. The name of the loaded file is appended.

FAIL\_LOAD\_(filename): Failed to load.

## Example

---

```
:MEDia:SD:LOAD:SET "CONF0001.SET",0  
:MEDia:SD:LOAD:SET?  
(Response) :MEDia:SD:LOAD:SET SUCCESS_LOAD_CONF0001 (when headers are enabled)
```

## Note

---

## Usage Conditions

---

## Saves the data and queries the save state

[When saving the waveform data (binary format)]

:MEDia:SD:SAVE:DATA:MEM

:MEDia:USB:SAVE:DATA:MEM

[When saving the waveform data (text format)]

:MEDia:SD:SAVE:DATA:CSV

:MEDia:USB:SAVE:DATA:CSV

[When saving the waveform data (MDF format)]

:MEDia:SD:SAVE:DATA:MF4

:MEDia:USB:SAVE:DATA:MF4

[When saving the setting data]

:MEDia:SD:SAVE:SET

:MEDia:USB:SAVE:SET

[When saving the A2L setting data (LAN1)]

:MEDia:SD:SAVE:A2L:LAN1

:MEDia:USB:SAVE:A2L:LAN1

[When saving the A2L setting data (LAN2)]

:MEDia:SD:SAVE:A2L:LAN2

:MEDia:USB:SAVE:A2L:LAN2

[When saving the numerical operation results]

:MEDia:SD:SAVE:CALC:CSV

:MEDia:USB:SAVE:CALC:CSV

## Syntax

---

### Command

[When saving the waveform data (binary format)]

:MEDia:SD:SAVE:DATA:MEM

:MEDia:USB:SAVE:DATA:MEM

[When saving the waveform data (text format)]

:MEDia:SD:SAVE:DATA:CSV

:MEDia:USB:SAVE:DATA:CSV

[When saving the waveform data (MDF format)]

:MEDia:SD:SAVE:DATA:MF4

:MEDia:USB:SAVE:DATA:MF4

[When saving the setting data]

:MEDia:SD:SAVE:SET

:MEDia:USB:SAVE:SET

[When saving the A2L setting data (LAN1)]

:MEDia:SD:SAVE:A2L:LAN1

:MEDia:USB:SAVE:A2L:LAN1

[When saving the A2L setting data (LAN2)]

:MEDia:SD:SAVE:A2L:LAN2

:MEDia:USB:SAVE:A2L:LAN2

[When saving the numerical operation results]

:MEDia:SD:SAVE:CALC:CSV  
:MEDia:USB:SAVE:CALC:CSV

## Query

[When saving the waveform data (binary format)]

:MEDia:SD:SAVE:DATA:MEM?

:MEDia:USB:SAVE:DATA:MEM?

[When saving the waveform data (text format)]

:MEDia:SD:SAVE:DATA:CSV?

:MEDia:USB:SAVE:DATA:CSV?

[When saving the waveform data (MDF format)]

:MEDia:SD:SAVE:DATA:MF4?

:MEDia:USB:SAVE:DATA:MF4?

[When saving the setting data]

:MEDia:SD:SAVE:SET?

:MEDia:USB:SAVE:SET?

[When saving the A2L setting data (LAN1)]

:MEDia:SD:SAVE:A2L:LAN1?

:MEDia:USB:SAVE:A2L:LAN1?

[When saving the A2L setting data (LAN2)]

:MEDia:SD:SAVE:A2L:LAN2?

:MEDia:USB:SAVE:A2L:LAN2?

[When saving the numerical operation results]

:MEDia:SD:SAVE:CALC:CSV?

:MEDia:USB:SAVE:CALC:CSV?

## Response

A\$ = NONE,EXECUTING,SUCCESS\_ (filename) ,FAIL

## Details

---

Saves the data.

Returns the save state.

NONE: Before saving.

EXECUTING: Saving in progress.

SUCCESS\_(filename): Succeeded in saving. The name of the saved file is appended.

FAIL: Failed to save.

## Example

---

:MEDia:SD:SAVE:DATA:MEM

:MEDia:SD:SAVE:DATA:MEM?

(Response) :MEDIA:SD:SAVE:DATA:MEM? SUCCESS\_TEST (when headers are enabled)

## Note

---

## Usage Conditions

---

If the file size of the waveform data exceeds 1 GB, the file is automatically divided and saved every 1 GB.

# 4

## Troubleshooting

### 4.1 Wired LAN connection

#### Troubleshooting Wired LAN connection

---

---

**Is the cable properly connected?**

→ Verify that the instrument is properly connected to the PC.

**Is the IP address valid?**

→ Check the [Wired LAN settings](#) in "2.1 Device Specifications, Settings".

**Is the IP address of the instrument duplicated on another device?**

→ Change [IP address](#) according to the wired LAN settings in "2.1 Device Specifications, Settings".

**Is the power turned on?**

→ Turn on all devices.



- ✓ This instruction manual explains the communication commands for Model LR8101, LR8102 Data Logger.
  - ✓ Before using LR8101, LR8102, be sure to read the instruction manual of LR8101, LR8102.
  - ✓ For details regarding the command settings, please refer to “2.1 Setup Protocols” in the instruction manual for Model LR8101, LR8102.
- Although all reasonable care has been taken in the production of this
- ✓ instruction manual, should you find any points which are unclear or in error, please contact your local distributor or HIOKI's website.

<https://www.hioki.com/contact>

**HIOKI**  
**www.hioki.com/**



**All regional  
contact  
information**

**HIOKI E.E. CORPORATION**

81 Koizumi, Ueda, Nagano 386-1192 Japan

2309 EN

Edited and published by HIOKI E.E. CORPORATION

Printed in Japan

- Contents subject to change without notice.
- This document contains copyrighted content.
- It is prohibited to copy, reproduce, or modify the content of this document without permission.
- Company names, product names, etc. mentioned in this document are trademarks or registered trademarks of their respective companies.

**Europe only**

- EU declaration of conformity can be downloaded from our website.
- Contact in Europe: HIOKI EUROPE GmbH  
Helfmann-Park 2, 65760 Eschborn, Germany [hioki@hioki.eu](mailto:hioki@hioki.eu)