UNI-T

MSO7000X Series Mixed Signal Oscilloscope

10GSa/s | 2GHz | 1Gpts | 2,000,000wfms/s

Programming Manual REV.3.0

2025.4

Introduction

This programming manual describes the SCPI commands for the MSO7000X series.

Copyright Information

Copyright is owned by Uni-Trend Technology (China) Co., Ltd.
If the original purchaser sells or transfers the product to a third party within three years from the date of purchase, the warranty period of three year shall be from the date of the original purchase from UNI-T or an authorized UNI-T distributor. Power cords, accessories, fuses, and other similar items are not covered by this warranty.
If the product is proved to be defective within the warranty period, UNI-T reserves the rights to either repair the defective product without charging of parts and labor, or exchange the defected product to a working equivalent product (determined by UNI-T). Replacement parts, modules and products may be brand new, or perform at the same specifications as brand-new products. All original parts, modules, or products which were defective become the property of UNI-T.
The "customer" refers to the individual or entity that is declared in the guarantee. To obtain warranty service, the customer must notify UNI-T of any defects within the applicable warranty period to UNI-T, and perform appropriate arrangements for the warranty service.
The customer is responsible for packing and shipping defective products to the individual or entity that is declared in the guarantee. In order obtain the warranty service, customer must inform the defects within the applicable warranty period to UNI-T, and perform appropriate arrangements for the warranty service. The customer is responsible for packing and shipping defective products to the designated maintenance center of UNI-T, pay the shipping cost, and provide a copy of the purchase receipt of the original purchaser. If the product is shipped domestically, the original purchaser must provide a purchase receipt. If the product is shipped to the location of the UNI-T service center, UNI-T shall pay the return shipping fee. If the product is sent to any other location, the customer is responsible for all shipping, duties, taxes, and any other expenses.

The warranty is inapplicable to any defects, failures or damages caused by accident, normal wear of components, use beyond specified scope or improper use of product, or improper or insufficient maintenance. UNI-T is not obligated to provide the following services as prescribed by the warranty:

- Repair damage caused by installation, repair or maintenance of personnel other than service representatives of UNI-T;
- Repair damage caused by improper use or connection to incompatible equipment;
- Repair any damages or failures caused by using power source not provided by UNI-T;

- Repair products that have been changed or integrated with other products (if such change or integration increases time or difficulty of repair).

The warranty is formulated by UNI-T for this product, replacing any other express or implied warranties. UNI-T and its distributors disclaim any implied warranties of marketability or fitness for a particular purpose. For violation of the warranty, repair or replacement of defective products is the only and all remedial measure UNI-T provides for customers. Regardless of whether UNI-T and its distributors are informed of possible indirect, special, incidental, or consequential damages, they assume no responsibility for such damages.

Trademark

UNI-T is the registered trademark of Uni-Trend Technology (China) Co., Ltd.

File Version

MSO7000X-V3.0

Statement

- UNI-T products are protected by patent rights in China and foreign countries, including issued and pending patents.
- UNI-T reserves the rights to any product specification and pricing changes.
- UNI-T reserves all rights. Licensed software products are properties of Uni-Trend and its subsidiaries or suppliers, which are protected by national copyright laws and international treaty provisions. Information in this manual supersedes all previously published versions.
- Technical data are subject to change without prior notice.

Software Version

3.0.0.0

The software may upgrade or add new function, please subscribe UNI-T official website www.instruments.uni-trend.com to get the latest edition and firmware.

Introduce SCPI

SCPI (Standard Commands for Programmable Instruments) is a standardized command set based on IEEE 488.1 and IEEE 488.2 standards. It follows IEEE 754 floating-point arithmetic rules, ISO646 information exchange 7bits code symbol (equivalent to ASCII programming) and other standard standardized instrument programming language.

This section describes the format, symbols, parameters, and abbreviation rules of the SCPI command.

Instruction Format

SCPI commands follow a hierarchical structure consisting of multiple subsystems, each consisting of a root keyword and one or more hierarchical key words.

The command line begins with a colon ":"; Keywords are separated by the colon ":", followed by optional parameter settings. The command keyword is separated by spaces from the first parameter. The command string must end with a newline <NL> character. Add the question mark "? " after the command line. It is usually indicated that this feature is being queried.

Symbol Description

The following three symbols are not part of SCPI command, it cannot send with the command. It usually used as supplementary description of command parameter.

- **Braces { }**

  Braces usually contain multiple optional parameters, and one of the parameters must be selected when sending the command.

  Such as : :ACQuire:TYPe <type>,type can be chose{NORMal | PEAK | HIGHres | AVERages | ENVelope}.

- **Vertical bar |**

  The vertical line is used to separate multiple parameter options, and one of the parameters must be selected when sending the command.

  Such as : :ACQuire:TYPe <type> { NORMal | PEAK | HIGHres | AVERages | ENVelope}。

- **Square Brackets < >**

  The parameters in the triangular brackets must be replaced with a valid value.

Such as, Send the :ACQuire:TYPe <type> command in the form of :ACQuire:TYPe NORMal.

Parameter Description

The parameters in this manual's commands fall into five types: Boolean, integer, real, discrete, and ASCII string.

- **Boolean**

  The parameter can be set to "ON" (1) or "OFF" (0).

  For example, the <enable> in the AWG1: ENABle <enable> command is of Boolean type {ON | OFF} or {1 | 0}.

- **Int**

  Unless otherwise specified, the parameter can take any integer value within the valid value range. Note: Do not set the parameter in decimal format, as it may cause an exception.

  For example, the parameter <value> in the :DISPlay:SCReen:BRIGtness <value> command can take any integer within the range of 5 to 100.

- **Real**

  Unless otherwise specified, the parameter can take any value within the valid value range.

  For example: For CH1, the parameter <offset> in the CHANnel1:OFFSet <offset> command is of real type.

- **Discrete**

  Parameters can only take several specified values or characters.

  For example, The :ACQuire:TYPe <type>{NORMal | PEAK | HIGHres | AVERages | ENVelope} command can only take the parameters: NORMal, PEAK, HIGHres, AVERages, ENVelope.

- **ASCII**

  String parameters can actually contain all ASCII character sets. Strings must start and end with paired quotes, using either single or double quotes. Quote delimiters can also be part of the string. Just type them twice without adding any characters in the middle. For example, set IP:SYST:COMM:LAN:IPAD "192.168.1.10".

Shorthand Rule

All commands are case-sensitive and may be written in uppercase or lowercase. If using abbreviations, all uppercase letters in the command must be included.

Data Return

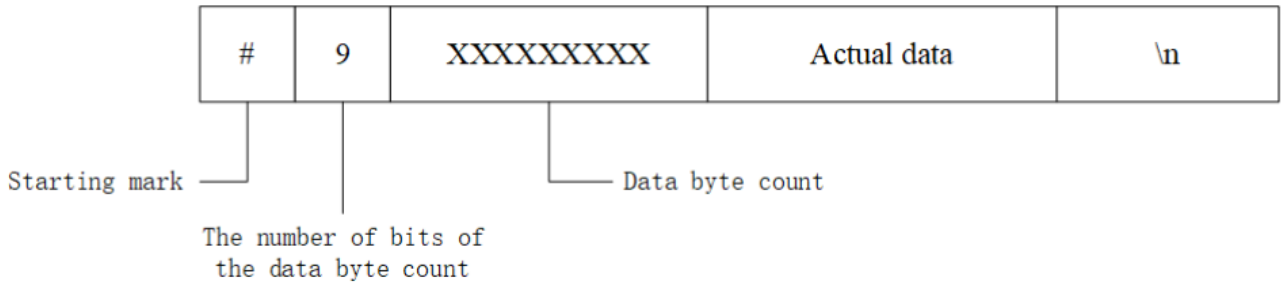
Data return is classified into single and batch returns.

For single data return, the corresponding parameter type is returned. For real type return, it is represented by scientific notation. When returning frequency and time values, 13 digits are retained after the decimal point in the front part of E, and 3 digits are retained in the E part. When returning other values, 5 digits are retained after the decimal point in the front part of E, and 3 digits are retained in the E part.

Batch data is returned in the format 'data block header + data block'. The data block header has the following format: The first digit after "#" (9) represents the number of remaining digits in the data block header; the remaining digits in the data block header represent the number of data bytes transmitted this time (if less than 9 digits, pad with zeros at the front). For example, the data block header for 1000 bytes is formatted as #9000001000.

Data block format:

DATA is a data stream, and others are ASCII characters. As shown in the figure below:<#9XXXXXXXXX + DATA + \n>.



Note: If invalid data is returned as an ASCII type, it is represented by '*'; For real data types, the maximum representable value is returned.

Detailed Explanation of SCPI Instructions

IEEE 488.2 General Commands

*** IDN**

- **Command Format:**
  *IDN?
- **Function Description:**
  Queries the manufacturer name, oscilloscope model, product serial number, and software version number.
- **Return Format:**
  Manufacturer name, oscilloscope model, product serial number, software version number separated by dots.
- **For example:**
  UNI-T, MSO7000X, MSO7204, 00.00.01

*** RST**

- **Command Format:**
  *RST
- **Function Description:**
  Restores factory settings and clears all error messages and send/receive queue buffers.

* SYSTem command

It is used for the most basic operations on the oscilloscope, mainly including operations such as run control, full keyboard locking, error queue, and system setting data.

:RUN

- **Command Format:**
  :RUN
- **Function Description:**
  Starts waveform sampling on the oscilloscope. To stop sampling, execute the :STOP command.

:STOP

- **Command Format:**

  :STOP

- **Function Description:**

  It is used to stop the waveform sampling work of the oscilloscope. To stop sampling, execute the :RUN command.

:AUTO

- **Command Format:**

  :AUTO

- **Function Description:**

  The automatic setting of the oscilloscope will automatically adjust the vertical range, horizontal time base and trigger mode according to the input signal to achieve the best waveform display. Note that the application of the automatic setting requires that the frequency of the measured signal is not less than 20Hz, the duty cycle is greater than 1%, and the amplitude is at least 10mVpp.

Note: This command is unavailable when some advanced functions are enabled.

:SYSTem:SINGle

- **Command Format:**

  :SYSTem:SINGle

- **Function Description:**

  Set the oscilloscope to the single-trigger mode.

:SYSTem:RTC

- **Command Format:**

  :SYSTem:RTC <value>

  :SYSTem:RTC ?

- **Function Description:**

  Set or query the system setting time。

- **Parameter:**

value:String type

- **Return format:**

  Query the return system time。

- **For example:**

  :SYSTem:RTC "2025,4,18,17,8,57" Set the system time to2025/4/28/17/8/57

  :SYSTem:RTC ? Query return2025,4,18,17,8,57

:SYSTem:TFORce

- **Command Format:**

  :SYSTem:TFORce

- **Function Description:**

  Force the generation of a trigger signal. Forced triggering is applicable to the "normal" and "single" trigger mode.

:SYSTem:CALibration

- **Command Format:**

  :SYSTem:CALibration

- **Function Description:**

  Oscilloscope self-calibration.

:SYSTem:CLEar

- **Command Format:**

  :SYSTem:CLEar

- **Function Description:**

  Clear the measurement results of the oscilloscope, etc.

:SYSTem:LANGuage

- **Command Format:**

  :SYSTem:LANGuage <language>

  :SYSTem:LANGuage ?

- **Function Description:**

Set or query the software language type.

- **Parameter:**

  Language: Discrete type {SIMPlifiedchinese | ENGLish}

  SIMPlifiedchinese: Simplified Chinese

  ENGLish: English

- **Return format:**

  Query and return the software language type.

- **For example:**

:SYSTem:LANGuage ENGLish	Set the language to English.
:SYSTem:LANGuage ?	Query and return ENGLish.

:SYSTem:PRTSc

- **Command Format:**

  :SYSTem:PRTSc ?

- **Function Description:**

  Screenshot.

- **Return format:**

  The query returns a byte array that represents an image.

  The read data format is header + waveform data points + terminator. The header is in the form of #NXXXXXX. # is the specified header identifier. N indicates that there are N bytes behind, describing the length of the waveform data points in the form of ASCII characters. The terminator signals the end of communication. For example, the data read at one time is #9000001000XXXX, indicating that 9 bytes describe the length of the data, and 000001000 indicates the length of the waveform data, that is, 1000 bytes.

:SYSTem:TOUChlock

- **Command Format:**

  :SYSTem:TOUChlock <active>

  :SYSTem:TOUChlock ?

- **Function Description:**

  Set or query the switch status of the touch lock.

- **Parameter:**

  active:Boolean type {ON | OFF} or {1 | 0}

- **Return format:**

The query returns that "1" and "0" represent "ON" and "OFF" respectively.

- **For example:**

  :SYSTem:TOUChlock ON Set the touch lock to be open.

  :SYSTem:TOUChlock ? The query returns 1.

:SYSTem:AUTO:TIMebase

- **Command Format:**

  :SYSTem:AUTO:TIMebase <status>

  :SYSTem:AUTO:TIMebase ?

- **Function Description:**

  When used for automatic setting, whether the horizontal time base scale attribute maintains the current state.

- **Parameter:**

  status:{AUTO | KEEP}

  AUTO:Autoset

  KEEP:Keep the original settings

- **Return format:**

  The query returns "AUTO", "KEEP".

- **For example:**

  :SYSTem:AUTO:TIMebase KEEP

  Set the horizontal time base scale attribute to maintain the current state.

  :SYSTem:AUTO:TIMebase ?

  The query returns KEEP.

:SYSTem:AUTO:VERTical

- **Command Format:**

  :SYSTem:AUTO:VERTical <status>

  :SYSTem:AUTO:VERTical ?

- **Function Description:**

  Whether the attribute of the vertical scale of the analog channel is maintained in the current state when used for automatic setting.

- **Parameter:**

  status:{AUTO| KEEP}

  AUTO:Autoset

KEEP:Keep the original settings

- **Return format:**

  The query returns "AUTO", "KEEP".

- **For example:**

  :SYSTem:AUTO:VERTical KEEP

  Set the trigger mode and source to maintain the current state.

  :SYSTem:AUTO:VERTical ?

  The query returns KEEP.

:SYSTem:AUTO:ACQuire

- **Command format:**

  :SYSTem:AUTO:ACQuire <status>

  :SYSTem:AUTO:ACQuire ?

- **Function description:**

  When used for automatic settings, whether the acquisition mode maintains the current state.

- **Parameter:**

  status:{AUTO | KEEP}

  AUTO:Autoset

  KEEP:Keep the original settings

- **Return format:**

  The query returns "AUTO", "KEEP".

- **For example:**

  :SYSTem:AUTO:ACQuire KEEP

  Set the attribute of the acquisition mode to maintain the current state.

  :SYSTem:AUTO:ACQuire ?

  The query returns KEEP.

:SYSTem:AUTO:TRIGger

- **Command format:**

  :SYSTem:AUTO:TRIGger <status>

  :SYSTem:AUTO:TRIGger ?

- **Function description:**

  When used for automatic setting, whether the trigger mode and trigger source maintain

the current state.

- **Parameter:**

  status:{AUTO| KEEP}

  AUTO:Autoset

  KEEP:Keep the original settings

- **Return format:**

  The query returns "AUTO", "KEEP".

- **For example:**

  :SYSTem:AUTO:TRIGger KEEP

  Set the attributes of the trigger mode and trigger source to maintain the current state.

  :SYSTem:AUTO:TRIGger ?

  The query returns KEEP.

:SYSTem:AUTO:CHANel:COUPling

- **Command format:**

  :SYSTem:AUTO:CHANel:COUPling <status>

  :SYSTem:AUTO:CHANel:COUPling ?

- **Function description:**

  When used for automatic setting, whether the channel coupling maintains the current state.

- **Parameter:**

  status:{AUTO | KEEP}

  AUTO:Autoset

  KEEP:Keep the original settings

- **Return format:**

  The query returns "AUTO", "KEEP".

- **For example:**

  :SYSTem:AUTO:CHANel:COUPling KEEP

  Set the channel coupling to maintain the current state.

  :SYSTem:AUTO:CHANel:COUPling ?

  The query returns KEEP.

:SYSTem:AUTO:CHANel:ACTive

- **Command format:**

:SYSTem:AUTO:CHANel:ACTive <status>

:SYSTem:AUTO:CHANel:ACTive ?

- **Function description:**

  When used for automatic setting, whether the channel is opened or closed to maintain the current state.

- **Parameter:**

  status:{AUTO| KEEP}

  AUTO:Autoset

  KEEP:Keep the original settings

- **Return format:**

  The query returns "AUTO", "KEEP".

- **For example:**

  :SYSTem:AUTO:CHANel:ACTive KEEP

  Set the channel switch to maintain the current state.

  :SYSTem:AUTO:CHANel:ACTive ?

  The query returns KEEP.

:SYSTem:AUTO:DISPlay:MODe

- **Command format:**

  :SYSTem:AUTO:DISPlay:MODe <status>

  :SYSTem:AUTO:DISPlay:MODe ?

- **Function description:**

  Whether the channels are stacked and displayed when used for automatic settings.

- **Parameter:**

  status:{AUTO | KEEP}

  AUTO:Autoset

  KEEP:Keep the original settings

- **Return format:**

  The query returns "AUTO", "KEEP".

- **For example:**

  :SYSTem:AUTO:DISPlay:MODe KEEP Set the channel stacked display.

  :SYSTem:AUTO:DISPlay:MODe ? The query returns KEEP.

Acquire

:ACQuire:STATus

- **Command format:**

  :ACQuire:STATus ?

- **Function description:**

  Query the sampling status.

- **Return format:**

  The query returns "STOP", "ARMED", "READY", "TRIGED", "AUTO", "ROLL", "RESET".

- **For example:**

  :ACQuire:STATus ? The query returns READY.

:ACQuire:TYPe

- **Command format:**

  :ACQuire:TYPe <type>

  :ACQuire:TYPe ?

- **Function description:**

  Set or query the sampling mode.

- **Parameter:**

  type:{NORMal | PEAK | HIGHres | AVERage | ENVelope}

  NORMal: Normal

  PEAK: Peak detection

  HIGHres: High resolution

  AVERage: Average

  ENVelope: Envelope

- **Return format:**

  The query returns "NORMal", "PEAK", "HIGHres", "AVERage", "ENVelope".

- **For example:**

  :ACQuire:TYPe NORMal Set the sampling mode to normal.

  :ACQuire:TYPe ? The query returns NORMal.

:ACQuire:AVERage

- **Command format:**

  :ACQuire:AVERage <count>

  :ACQuire:AVERage ?

- **Function description:**

  Set or query the number of averages in average sampling mode.

- **Parameter:**

  count: Integer type, range 2 - 65536

- **Return format:**

  The query returns an integer representing the average number of times.

- **For example:**

  :ACQuire:AVERage 20 Set the average number of times to 20.

  :ACQuire:AVERage ? The query returns 20.

:ACQuire:ENVelope

- **Command format:**

  :ACQuire:ENVelope <count>

  :ACQuire:ENVelope ?

- **Function description:**

  Specify or query the number of envelopes in envelope sampling mode.

- **Parameter:**

  count: Int, range 2 - 65536

- **Return format:**

  The query returns the integer of the number of envelopes in the envelope sampling mode.

- **For example:**

  :ACQuire:ENVelope 30 Set the number of envelopes to 30 times.

  :ACQuire:ENVelope ? The query returns 30.

:ACQuire:INTerpolation:MODe

- **Command format:**

  :ACQuire:INTerpolation:MODe <mode>

  :ACQuire:INTerpolation:MODe ?

- **Function description:**

  Set or query the interpolation method.

- **Parameter:**

  mode:{LINe | SINX}

  LINe:Linear

  SINX:Sine

- **Return format:**

  The query returns "LINe", "SINX".

- **For example:**

  :ACQuire:INTerpolation:MODe LINe Set the interpolation method to linear.

  :ACQuire:INTerpolation:MODe ? The query returns LINe.

:ACQuire:CLKSrc

- **Command format:**

  :ACQuire:CLKSrc <clksrc>

  :ACQuire:CLKSrc ?

- **Function description:**

  Set or query the source of the 10 MHz clock signal.

- **Parameter:**

  clksrc:{INNer | OUTTer}

  INNer:internal

  OUTTer:outside

- **Return format:**

  The query returns "INNer" and "OUTTer".

- **For example:**

  :ACQuire:CLKSrc OUTTer Set the source of the 10 MHz clock signal to external.

  :ACQuire:CLKSrc ? The query returns OUTTer.

:ACQuire:EXTLocked

- **Command format:**

  :ACQuire:EXTLocked ?

- **Function description:**

  Query whether the 10MHz clock frequency of the external input is locked.

- **Return format:**

  The query returns "1", "0"

  1: Locked

  0: Unlocked

:ACQuire:FAST

- **Command format:**

  :ACQuire:FAST <active>

  :ACQuire:FAST ?

- **Function description:**
- Set or query the switch status of fast acquisition.
- **Parameter:**

  active: Boolean type {ON | OFF} or {1 | 0}

- **Return format:**

  The query returns "1" and "0", representing "ON" and "OFF" respectively.

- **For example:**

  :ACQuire:FAST ON Set UltraAcq to be on.

  :ACQuire:FAST ? The query returns 1.

:ACQuire:STORage

- **Command format:**

  :ACQuire:STORage <type>

  :ACQuire:STORage ?

- **Function description:**

  Set or query the memory depth.

- **Parameter:**

  Type: Discrete type, default unit Pts

  When choosing AUTO, the oscilloscope automatically selects the storage depth based on the current sampling rate.

  In 2.5G mode: {25k | 250k | 2.5M | 25M | 250M}

  In 5G mode: {50k | 500k | 5M | 50M | 500M}

  In 10G mode: {100k | 1M | 10M | 100M | 1G}

- **Return format:**

The query returns the actual memory depth.

- **For example:**

:ACQuire:STORage 25k Set the memory depth to 25kPts.

:ACQuire:STORage ? The query returns 2.50000E+004.

:ACQuire:ERES:ENABle

- **Command format:**

:ACQuire:ERES:ENABle <enable>

:ACQuire:ERES:ENABle ?

- **Function description:**

Set or query the on state of enhanced resolution.

- **Parameter:**

enable: Boolean type {ON | OFF} or {1 | 0}

- **Return format:**

The query returns "1" and "0", representing "ON" and "OFF" respectively.

- **For example:**

:ACQuire:ERES:ENABle ON Set enhanced resolution on.

:ACQuire:ERES:ENABle ? The query returns 1.

:ACQuire:ERES:BIT

- **Command format:**

:ACQuire:ERES:BIT <bit>

:ACQuire:ERES:BIT ?

- **Function description:**

Set or query the enhanced resolution and the number of enhanced bits.

- **Parameter:**

bit: Real number type, range from 0.5 to 3, step is 0.5

- **Return format:**

The query returns the number of enhanced bits of the enhanced resolution, expressed in scientific notation.

- **For example:**

:ACQuire:ERES:BIT 1

Set the number of enhanced bits of the enhanced resolution to 1.

:ACQuire:ERES:BIT ?

The query returns 1.00000E+000.

AWG

:AWG<n>:LOAD

- **Command format:**

  :AWG<n>:LOAD <load>

  :AWG<n>:LOAD ?

- **Function description:**

  Set or query the load impedance of the specified channel.

- **Parameter:**

  load: Discrete type {HIGH | LOW}

  HIGH:1MΩ

  LOW:50Ω

- **Return format:**

  The query returns "HIGH", "LOW".

- **For example:**

  :AWG1:LOAD HIGH

  Set the load impedance of channel 1 to high impedance.

  :AWG1:LOAD ?

  The query returns that the load impedance of channel 1 is HIGH.

:AWG<n>:ENABle

- **Command format:**

  :AWG<n>:ENABle <enable>

  :AWG<n>:ENABle ?

- **Function description:**

  Set or query to turn on or off the output of the specified AWG channel.

- **Parameter:**

  enable: Boolean type {ON | OFF} or {1 | 0}

- **Return format:**

  The query returns that "1" and "0" represent "ON" and "OFF" respectively.

- **For example:**

  :AWG1:ENABle ON Set the output of GEN 1 to be on.

  :AWG1:ENABle ? The query returns 1.

:AWG<n>:MODE

- **Command format:**

  :AWG<n>:MODE <mode>

  :AWG<n>:MODE ?

- **Function description:**

  Set or query the output mode of the specified AWG channel.

- **Parameter:**

  mode:{CONTinuous | MODulation | SWEep}

  CONTinuous:Continuous wave

  MODulation: Modulation wave

  SWEep:Sweep wave

- **Return format:**

  The query returns "CONTinuous", "MODulation", "SWEep".

- **For example:**

  :AWG1:MODE MODulation Set the output mode of AWG1 as modulation wave.

  :AWG1:MODE ? The query returns MODulation.

Continuous

:AWG<n>:BASe:WAVe

- **Command format:**

  :AWG<n>:BASe:WAVe <type>

  :AWG<n>:BASe:WAVe ?

- **Function description:**

  Set or query the basic waveform type of the specified channel.

- **Parameter:**

  type: Discrete type{SINusoid | SQUare | PULSe | RAMP | NOISe | DC | SINC| EXPRise | EXPFall | LORentz | HAVersine | GAUSsian | ECG | ARBitrary}

  SINusoid: Sine

  SQUare: Square

PULSe: Pulse

RAMP: Ramp

NOISe: Noise

DC: Direct current

SINC: Sinc

EXPRise: Exponential rise

EXPFall: Exponential fall

LORentz: Lorentz

HAVersine: Half-sine

GAUSsian: Gaussian

ECG: Electrocardiogram

ARBitrary: Arbitrary waveform

- **Return format:**

  The query returns "SINusoid", "SQUare", "PULSe", "RAMP", "NOISe", "DC", "SINC", "EXPRise", "EXPFall", "LORentz", "HAVersine", "GAUSsian", "ECG", "ARBitrary".

- **For example:**

  :AWG1:BASe:WAVe SINusoid

  Set the basic waveform of channel 1 as sine waveform

  :AWG1:BASe:WAVe ?

  The query returns that the basic waveform of channel 1 is SINusoid

:AWG<n>:BASe:FREQuency

- **Command format:**

  :AWG<n>:BASe:FREQuency <freq>

  :AWG<n>:BASe:FREQuency ?

- **Function description:**

  Set or query the basic waveform frequency of the specified channel.

- **Parameter:**

  freq: Real number type, unit: uHz, mHz, Hz, kHz, MHz. If no unit is entered, the default unit is Hz.

- **Return format:**

  The query returns the basic waveform frequency of the specified channel.

- **For example:**

:AWG1:BASe:FREQuency 20

Set the waveform frequency of channel 1 to 20 Hz.

:AWG1:BASe:FREQuency ?

The query returns that the waveform frequency of channel 1 is 2.0000000000000E+001.

:AWG<n>:BASe:PHASe

- **Command format:**

  :AWG<n>:BASe:PHASe <phase>

  :AWG<n>:BASe:PHASe ?

- **Function description:**

  Set or query the amplitude output of the specified channel.

- **Parameter:**

  phase: Real number type, representing the phase, in units of °, ranging from 0° to 359.99°

- **Return format:**

  The query returns the phase of the basic waveform of the specified channel.

- **For example:**

  :AWG1:BASe:PHASe 20

  Set the waveform phase of channel 1 to 20°.

  :AWG1:BASe:PHASe?

  The query returns that the waveform phase of channel 1 is 2.00000E+001.

:AWG<n>:BASe:AMPLitude

- **Command format:**

  :AWG<n>:BASe:AMPLitude <amp>

  :AWG<n>:BASe:AMPLitude ?

- **Function description:**

  Set or query the output amplitude of the specified channel.

- **Parameter:**

  amp: Real number type, representing voltage, with units mVpp, Vpp. If no unit is input, the default unit is Vpp. High-impedance output range: 20mVpp ~ 6Vpp; Low-impedance output range: 10mVpp ~ 3Vpp.

- **Return format:**

  The query returns the amplitude of the basic waveform of the specified channel.

- **For example:**

  :AWG1:BASe:AMPLitude 2

  Set the waveform amplitude of channel 1 to 2Vpp.

  :AWG1:BASe:AMPLitude ?

  The query returns that the waveform amplitude of channel 1 is 2.00000E+000.

:AWG<n>:BASe:OFFSet

- **Command format:**

  :AWG<n>:BASe:OFFSet <offset>

  :AWG<n>:BASe:OFFSet ?

- **Function description:**

  Set or query the output offset of the specified channel.

- **Parameter:**

  Offset: Real number type, representing voltage, in units of mV, V. If no unit is input, the default unit is V. The high-impedance output range: -3V ~ 3V; the low-impedance output range: -1.5V ~ 1.5V.

- **Return format:**

  The query returns the output offset of the basic waveform of the specified channel.

- **For example:**

  :AWG1:BASe:OFFSet 2

  Set the waveform offset of channel 1 to 2V.

  :AWG1:BASe:OFFSet ?

  The query returns that the waveform offset of channel 1 is 2.00000E+000.

:AWG<n>:BASe:HIGH

- **Command format:**

  :AWG<n>:BASe:HIGH <voltage>

  :AWG<n>:BASe:HIGH ?

- **Function description:**

  Set or query the high value of the signal output of the specified channel.

- **Parameter:**

  Voltage: Real number type, representing voltage, in units of mV, V. If no unit is entered, the default unit is V. High-impedance output range: -3V ~ 3V; Low-impedance output range: -1.5V ~ 1.5V.

- **Return format:**

  The query returns the high value of the signal output of the specified channel.

- **For example:**

  :AWG1:BASe:HIGH 1

  Set the high value of the signal output of channel 1 to 1V.

  :AWG1:BASe:HIGH ?

  The query returns that the high value of the signal output of channel 1 is 1.00000E+000.

:AWG<n>:BASe:LOW

- **Command format:**

  :AWG<n>:BASe:LOW <voltage>

  :AWG<n>:BASe:LOW ?

- **Function description:**

  Set or query the low value of the signal output of the specified channel.

- **Parameter:**

  Voltage: Real type, representing voltage, with the unit V or mV. If the unit is not input, the default unit is V. The output range of high impedance: -3V ~ 3V; The output range of low impedance: -1.5V ~ 1.5V.

- **Return format:**

  The query returns the low value of the signal output of the specified channel.

- **For example:**

  :AWG1:BASe:LOW 1

  Set the low value of the signal output of channel 1 to 1V.

  :AWG1:BASe:LOW ?

  The query returns that the low value of the signal output of channel 1 is 1.00000E+000.

:AWG<n>:PULSe:RISe

- **Command format:**

  :AWG<n>:PULSe:RISe <time>

  :AWG<n>:PULSe:RISe ?

- **Function description:**

  Set or query the rising edge time of the pulse wave of the specified channel.

- **Parameter:**

  time: Real type, unit ns, us. The default unit is s. The range is 5ns to 399.999us.

- **Return format:**

  The query returns the rising edge time of the pulse waveform of the specified channel.

- **For example:**

  :AWG1:PULSe:RISe 10ns

  Set the rising time of the pulse wave of channel 1 to 10ns.

  :AWG1:PULSe:RISe ?

  The query returns 1.0000000000000E-008.

Note: This instruction is only used under the pulse wave.

:AWG<n>:PULSe:FALL

- **Command format:**

  :AWG<n>:PULSe:FALL <time>

  :AWG<n>:PULSe:FALL ?

- **Function description:**

  Set or query the falling edge time of the pulse wave of the specified channel.

- **Parameter:**

  time: Real number type, in units of ns, us. The default unit is s. The range is 5ns to 399.999us.

- **Return format:**

  Query and return the falling edge time of the pulse wave of the specified channel.

- **For example:**

  :AWG1:PULSe:FALL 10ns

  Set the falling time of the pulse wave of channel 1 to 10ns.

  :AWG1:PULSe:FALL ?

  Query and return 1.0000000000000E-008.

Note: This instruction is only used under the pulse wave.

:AWG<n>:BASe:DUTY

- **Command format:**

  :AWG<n>:BASe:DUTY <duty>

  :AWG<n>:BASe:DUTY ?

- **Function description:**

  Set or query the duty cycle of the signal output of the specified channel.

- **Parameter:**

Duty: Real number type, representing the duty cycle, in unit %, ranging from 0 to 100.

- **Return format:**

  Query and return the duty cycle of the signal output of the specified channel.

- **For example:**

  :AWG1:BASe:DUTY 50

  Set the duty cycle of the signal output of channel 1 to 50%.

  :AWG1:BASe:DUTY ?

  Query and return the duty cycle of the signal output of channel 1 as 5.00000E+001.

Note: This instruction is only used under the square wave and pulse wave.

:AWG<n>:BASe:SYMMetry

- **Command format:**

  :AWG<n>:BASe:SYMMetry <duty>

  :AWG<n>:BASe:SYMMetry ?

- **Function description:**

  Set or query the signal output symmetry of the specified channel.

- **Parameter:**

  Duty: Real number type, representing the duty cycle, in units of %, with a range of 0 to 100.

- **Return format:**

  Query and return the signal output symmetry of the specified channel.

- **For example:**

  :AWG1:BASe:SYMMetry 50

  Set the signal output symmetry of channel 1 to 50%.

  :AWG1:BASe:SYMMetry ?

  Query and return the signal output symmetry of channel 1 as 5.00000E+001.

Note: This instruction is only used under the ramp wave.

:AWG<n>:BASe:ARBitrary

- **Command format:**

  :AWG<n>:BASe:ARBitrary path

  :AWG<n>:BASe:ARBitrary ?

- **Function description:**

  Set or query the reading path of the arbitrary wave file of the specified channel.

- **Parameter:**

  path:The opening path of the AWG arbitrary wave.

- **Return format:**

  The query returns the path of the current arbitrary wave file of the specified channel.

- **For example:**

  :AWG1:BASe:ARBitrary “D:\\U2 DSO\\WaveForm\\Uni-t000.bin”

  Set the reading path of the arbitrary wave file of channel 1 to D:\\U2 DSO\\WaveForm\\Uni-t000.bin.

  :AWG1:BASe:ARBitrary ?

  The query returns D:\\U2 DSO\\WaveForm\\Uni-t000.bin.

Modulation

:AWG<n>:MODUlate:MODe

- **Command format:**

  :AWG<n>:MODUlate:MODe <mode>

  :AWG<n>:MODUlate:MODe ?

- **Function description:**

  Set or query the signal modulation mode of the specified channel.

- **Parameter:**

  mode: Discrete type {AM | PM | FM}

  AM:Amplitude modulation

  PM:Phase modulation

  FM:Frequency modulation

- **Return format:**

  The query returns "AM", "PM", "FM".

- **For example:**

  :AWG1:MODUlate:MODe AM

  Set the signal of channel 1 to amplitude modulation mode.

  :AWG1:MODUlate:MODe ?

  The query returns AM.

:AWG<n>:MODUlate:WAVe

- **Command format:**

:AWG<n>:MODUlate:WAVe <type>

:AWG<n>:MODUlate:WAVe ?

- **Function description:**

  Set or query the modulation waveform of the signal of the specified channel.

- **Parameter:**

  Type: Discrete type {SINusoid | SQUare | RAMP | NOISe | ARBitrary}

  SINusoid: Sinusoidal waveform

  SQUare: Square waveform

  RAMP: Ramp waveform

  NOISe: Noise

  ARBitrary: Arbitrary waveform

- **Return format:**

  The query returns "SINusoid", "SQUare", "RAMP", "NOISe", "ARBitrary".

- **For example:**

  :AWG1:MODUlate:WAVe SINusoid

  Set the modulation wave type of the signal of channel 1 to sinusoidal wave.

  :AWG1:MODUlate:WAVe ?

  The query returns SINusoid.

:AWG<n>:MODUlate:FREQuency

- **Command format:**

  :AWG<n>:MODUlate:FREQuency <freq>

  :AWG<n>:MODUlate:FREQuency ?

- **Function description:**

  Specify or query the modulation frequency for the channel signal.

- **Parameter:**

  Freq: Real number type, representing frequency, with units of mHz, Hz, kHz. If no unit is entered, the default unit is Hz.

- **Return format:**

  The query returns the modulation frequency of the signal of the specified channel.

- **For example:**

  :AWG1:MODUlate:FREQuency 2000

  Set the modulation frequency of the signal of channel 1 to 2kHz.

:AWG1:MODUlate:FREQuency ?

The query returns 2.0000000000000E+003.

:AWG<n>:FM:FREQuency:DEV

- **Command format:**

  :AWG<n>:FM:FREQuency DEV <freq>

  :AWG<n>:FM:FREQuency ?

- **Function description:**

  Set or query the frequency offset of the waveform of the specified channel.

- **Parameter:**

  freq: Real number type, 100mHz to 12.5MHz, unit Hz

- **Return format:**

  The query returns the frequency offset of the waveform of the specified channel.

- **For example:**

  :AWG1:FM:FREQuency:DEV 2

  Set the frequency offset of channel 1 to 2Hz.

  :AWG1:FM:FREQuency:DEV ?

  The query returns that the frequency offset of channel 1 is 2.0000000000000E+000.

:AWG<n>:MODUlate:DEPTh

- **Command format:**

  :AWG<n>:MODUlate:DEPTh <depth>

  :AWG<n>:MODUlate:DEPTh ?

- **Function description:**

  Set or query the signal modulation depth of the specified channel.

- **Parameter:**

  Depth: Real type, representing the modulation depth, with the unit %. The AM modulation depth is 1% to 120%.

- **Return format:**

  The query returns the modulation depth of the signal of the specified channel.

- **For example:**

  :AWG1:MODUlate:DEPTh 50

  Set the signal modulation depth of channel 1 to 50%.

  :AWG1:MODUlate:DEPTh ?

The query returns 5.00000E+001.

:AWG<n>:PM:PHASe:DEV

- **Command format:**

  :AWG<n>:PM:PHASe:DEV <value>

  :AWG<n>:PM:PHASe:DEV ?

- **Function description:**

  Set or query the phase modulation phase deviation of the signal of the specified channel.

- **Parameter:**

  Value: Real number type, range 0° to 359.99°, unit ° (degree).

- **Return format:**

  The query returns the phase modulation phase deviation of the signal of the specified channel.

- **For example:**

  :AWG1:PM:PHASe:DEV 50

  Set the signal modulation phase deviation of channel 1 to 50°.

  :AWG1:PM:PHASe:DEV ?

  The query returns 5.00000E+001.

Sweep

:AWG<n>:SWEep:TYPe

- **Command format:**

  :AWG<n>:SWEep:TYPe <type>

  :AWG<n>:SWEep:TYPe ?

- **Function description:**

  Set or query the sweep type for the specified channel.

- **Parameter:**

  type:{LINear | LOGarithmic}

  LINear:Linear

  LOGarithmic:Logarithmic

- **Return format:**

  The query returns "LINear ", "LOGarithmic".

- **For example:**

:AWG1:SWEep:TYPe LOGarithmic Set the sweep type of channel 1 to logarithmic.

:AWG1:SWEep:TYPe ? The query returns LOGarithmic.

:AWG<n>:SWEep:DURatio

- **Command format:**

  :AWG<n>:SWEep:DURatio <time>

  :AWG<n>:SWEep:DURatio ?

- **Function description:**

  Set or query the sweep time of the specified channel.

- **Parameter:**

  time: Real type, in units of ms, s, the default unit is s.

- **Return format:**

  Query and return the sweep time of the specified channel.

- **For example:**

  :AWG1:SWEep:DURatio 1 Set the sweep time for channel 1 to 1 second.

  :AWG1:SWEep:DURatio ? Query and return 1.0000000000000E+000.

:AWG<n>:SWEep:FREQuency:STARt

- **Command format:**

  :AWG<n>:SWEep:FREQuency:STARt <freq>

  :AWG<n>:SWEep:FREQuency:STARt ?

- **Function description:**

  Set or query the starting sweep frequency for the specified channel.

- **Parameter:**

  freq: Real number type, unit mHz, Hz, kHz, MHz, default unit Hz.

- **Return format:**

  Query and return the starting sweep frequency for the specified channel.

- **For example:**

  :AWG1:SWEep:FREQuency:STARt 1

  Set the starting sweep frequency for channel 1 to 1Hz.

  :AWG1:SWEep:FREQuency:STARt ?

  Query and return 1.0000000000000E+000.

:AWG<n>:SWEep:FREQuency:END

- **Command format:**

  :AWG<n>:SWEep:FREQuency:END <freq>

  :AWG<n>:SWEep:FREQuency:END ?

- **Function description:**

  Set or query the end frequency of the sweep for the specified channel.

- **Parameter:**

  freq: Real number type, in units of mHz, Hz, kHz, MHz, with the default unit being Hz.

- **Return format:**

  Query and return the end frequency of the sweep for the specified channel.

- **For example:**

  :AWG1:SWEep:FREQuency:END 1

  Set the end frequency of the sweep for channel 1 to 1Hz.

  :AWG1:SWEep:FREQuency:END ?

  Query and return 1.0000000000000E+000.

:AWG<n>:SWEep:TRIGger:SOURce

- **Command format:**

  :AWG<n>:SWEep:TRIGger:SOURce <source>

  :AWG<n>:SWEep:TRIGger:SOURce ?

- **Function description:**

  Set or query the sweep trigger source of the specified channel.

- **Parameter:**

  source:{INSide | MANual | OUTSide}

  INSide:Inside

  MANual:Manual

  OUTSide:Outside

- **Return format:**

  Query and return the sweep trigger source of the specified channel.

- **For example:**

  :AWG1:SWEep:TRIGger:SOURce OUTSide

  Set the sweep trigger source of channel 1 to outside.

  :AWG1:SWEep:TRIGger:SOURce ?

Query and return Outside.

:AWG<n>:SWEep:TRIGger:ENABle

- **Command format:**

  :AWG<n>:SWEep:TRIGger:ENABle <enable>

  :AWG<n>:SWEep:TRIGger:ENABle ?

- **Function description:**

  Set or query the sweep trigger output state for the specified channel.

- **Parameter:**

  enable: Boolean type {ON | OFF} or {1 | 0}

- **Return format:**

  The query returns that "1" and "0" represent "ON" and "OFF" respectively.

- **For example:**

  :AWG1:SWEep:TRIGger:ENABle OFF

  Set the sweep trigger output of channel 1 to be off.

  :AWG1:SWEep:TRIGger:ENABle ?

  The query returns 0.

:AWG<n>:SWEep:TRIGger:MANualtrig

- **Command format:**

  :AWG<n>:SWEep:TRIGger:MANualtrig

- **Function description:**

  When setting the sweep frequency trigger source of the specified channel as "manual", trigger it once.

Channel

:CHANnel<n>:SELect

- **Command format:**

  :CHANnel<n>:SELect

  :CHANnel<n>:SELect ?

- **Function description:**

  Set or query the selected state of the channel

- **Parameter:**

  <n>:Number of analog channels, range from 1 to 4

- **Return format:**

  Query and return the selected status of the analog channels. Returning "1" and "0" represent selected and unselected respectively.

- **For example:**

  :CHANnel1:SELect Set channel 1 to the selected state.

  :CHANnel1:SELect? The query returns 1

:CHANnel<n>:BWLimit

- **Command format:**

  :CHANnel<n>:BWLimit <bwlimit>

  :CHANnel<n>:BWLimit ?

- **Function description:**

  Set or query the bandwidth limit for the channel.

- **Parameter:**

  bwlimit: Discrete type {FULL | 1GHz | 500MHz | 20MHz}

  FULL: The bandwidth limit is full bandwidth

  1GHz: The bandwidth limit is 1GHz, valid only for low resistance

  500MHz: The bandwidth limit is 500MHz, valid only for low resistance

  20MHz: The bandwidth limit is 20MHz

- **Return format:**

  The query returns "FULL", "1GHz", "500MHz", "20MHz".

- **For example:**

  :CHANnel1:BWLimit FULL

  Set the bandwidth limit for channel 1 to full bandwidth.

  :CHANnel1:BWLimit ?

  The query returns FULL.

:CHANnel<n>:COUPling

- **Command format:**

  :CHANnel<n>:COUPling <coupling>

  :CHANnel<n>:COUPling ?

- **Function description:**

  Set or query the coupling mode for the channel input as "DC1MΩ", "AC1MΩ", "DC50Ω", "GND".

- **Parameter:**

  coupling: Discrete type {DC1M | AC1M | DC50 | GND}

  DC: The coupling mode is direct current

  AC: The coupling mode is alternating current

- **Return format:**

  The query returns "DC1M", "AC1M", "DC50", "GND".

- **For example:**

  :CHANnel1:COUPling DC1M Set the coupling mode of channel 1 to DC1MΩ.

  :CHANnel1:COUPling ? The query returns DC1M.

:CHANnel<n>:DISPlay

- **Command format:**

  :CHANnel<n>:DISPlay <active>

  :CHANnel<n>:DISPlay ?

- **Function description:**

  Turn off or on the display of the channel.

- **Parameter:**

  active: Boolean type {ON | OFF} or {1 | 0}

- **Return format:**

  The query returns that "1" and "0" represent "ON" and "OFF" respectively.

- **For example:**

  :CHANnel1:DISPlay ON Turn on the display of channel 1.

  :CHANnel1:DISPlay ? The query returns 1.

:CHANnel<n>:INVert

- **Command format:**

  :CHANnel<n>:INVert <invert>

  :CHANnel<n>:INVert ?

- **Function description:**

  Turn off or on the inverted display of the channel.

- **Parameter:**

  invert: Boolean type {ON | OFF} or {1 | 0}

- **Return format:**

  The query returns that "1" and "0" represent "ON" and "OFF" respectively.

- **For example:**

  :CHANnel1:INVert ON Turn on the inverted display of channel 1.

  :CHANnel1:INVert ? The query returns 1.

:CHANnel<n>:OFFSet

- **Command format:**

  :CHANnel<n>:OFFSet <offset>

  :CHANnel<n>:OFFSet ?

- **Function description:**

  Set or query the vertical offset for the waveform display of the specified channel.

- **Parameter:**

  offset: Real number type, -5.5 to 5.5, centered at 0, unit div

- **Return format:**

  The query returns the vertical offset of the specified channel.

- **For example:**

  :CHANnel1:OFFSet 1 Set the vertical offset for channel 1 to 1div.

  :CHANnel1:OFFSet ? The query returns 1.00000E+000.

:CHANnel<n>:PROBe:MODel

- **Command format:**

  :CHANnel<n>:PROBe:MODel ?

- **Function description:**

  Query the probe model.

:CHANnel<n>:PROBe:SERailnumber

- **Command format:**

  :CHANnel<n>:PROBe:SERailnumber ?

- **Function description:**

  Query the probe serial number.

:CHANnel<n>:PROBe:GAIN:CURRent

- **Command format:**

  :CHANnel<n>:PROBe:GAIN:CURRent <probe>

  :CHANnel<n>:PROBe:GAIN:CURRent ?

- **Function description:**

  Set or query the attenuation ratio for the probe

- **Parameter:**

  probe: Discrete {1 | 10 | 100 | CUSTom}

  1: The probe attenuation ratio is 1

  10: The probe attenuation ratio is 10

  100: The probe attenuation ratio is 100

  CUSTOM: Custom

- **Return format:**

  Query and return the probe attenuation ratio.

- **For example:**

  :CHANnel1:PROBe:GAIN:CURRent 10

  Set the probe attenuation ratio for channel 1 to 10.

  :CHANnel1:PROBe:GAIN:CURRent ?

  Query and return 10.

:CHANnel<n>:PROBe:GAIN:CUSTom

- **Command format:**

  :CHANnel<n>:PROBe:GAIN:CUSTom <count>

  :CHANnel<n>:PROBe:GAIN:CUSTom ?

- **Function description:**

  Set or query the custom magnification ratio for the probe.

- **Parameter:**

  count: Real number type, setting range: 0.001X - 1000X

- **Return format:**

  Query and return the custom probe magnification ratio, expressed in scientific notation.

- **For example:**

  :CHANnel1:PROBe:GAIN:CUSTom 10

Set the custom probe magnification ratio for channel 1 to 10.

:CHANnel1:PROBe:GAIN:CUSTom ?

Query and return 1.00000E+001.

:CHANnel<n>:PROBe:BUTTon

- **Command format:**

  :CHANnel<n>:PROBe:BUTTon <button>

  :CHANnel<n>:PROBe:BUTTon ?

- **Function description:**

  Set or query the probe button type.

- **Parameter:**

  Button: Discrete type {HEADlight | RUNorstop | CLEar | FORCetrig | NOOPs}

  HEADlight: Headlamp

  RUNorstop: Run/Stop

  CLEar: Clear

  FORCetrig: Force Trigger

  NOOPs: No Operation

- **Return format:**

  The query returns "HEADlight", "RUNorstop", "CLEar", "FORCetrig", "NOOPs".

- **For example:**

  :CHANnel1:PROBe:BUTTon NOOPs

  Set the probe button type for channel 1 as 'No Operation.

  :CHANnel1:PROBe:BUTTon ?

  The query returns NOOPs.

:CHANnel<n>:UNIT:ACTive

- **Command format:**

  :CHANnel<n>:UNIT:ACTive <unit>

  :CHANnel<n>:UNIT:ACTive ?

- **Function description:**

  Set or query the activation status of the standby unit.

- **Parameter:**

  active: Boolean type {ON | OFF} or {1 | 0}.

- **Return format:**

  The query returns that "1" and "0" represent "ON" and "OFF" respectively.

- **For example:**

  :CHANnel1:UNIT:ACTive ON Set the standby unit of Channel 1 to be turned on.

  :CHANnel1:UNIT:ACTive ? The query returns 1.

:CHANnel<n>:UNIT:RATio

- **Command format:**

  :CHANnel<n>:UNIT:RATio <value>

  :CHANnel<n>:UNIT:RATio ?

- **Function description:**

  Set or query the standby unit ratio.

- **Parameter:**

  value: Real number type, with a setting range from 0.001 to 1000

- **Return format:**

  Query and return the value of the standby unit ratio。

- **For example:**

  :CHANnel1:UNIT:RATio 3 Set the probe unit ratio of Channel 1 to 3A/V.

  :CHANnel1:UNIT:RATio ? Query and return 3

:CHANnel<n>:SCALe

- **Command format:**

  :CHANnel<n>:SCALe <scale>

  :CHANnel<n>:SCALe?

- **Function description:**

  Set or query the vertical scale for the waveform display of the specified channel.

- **Parameter:**

  scale: Real number type, unit: Custom, If no unit is entered, the default unit is V.

  When the input impedance is 1MΩ, the adjustable range is from 1mV to 10V.

  When the input impedance is 50Ω, the adjustable range is from 1mV to 1V.

- **Return format:**

The query returns the vertical scale value in scientific notation, and the default unit is V.

- **For example:**

  :CHANnel1:SCALe 1V

  Set the vertical scale of the waveform display of channel 1 to 1V.

  :CHANnel1:SCALe ?

  The query returns 1.00000E+000.

:CHANnel<n>:FINe

- **Command format:**

  :CHANnel<n>:FINe <active>

  :CHANnel<n>:FINe ?

- **Function description:**

  Set or query whether the fine adjustment of the vertical scale of the specified channel is turned on.

- **Parameter:**

  active:Boolean type {ON | OFF} or {1 | 0}

- **Return format:**

  The query returns that "1" and "0" represent "ON" and "OFF" respectively.

- **For example:**

  :CHANnel1:FINe ON

  Set the fine adjustment of the vertical scale of channel 1 to be on.

  :CHANnel1:FINe ?

  The query returns 1.

:CHANnel<n>:BIAS

- **Command format:**

  :CHANnel<n>:BIAS <bias>

  :CHANnel<n>:BIAS ?

- **Function description:**

  Set or query the bias voltage for the channel.

- **Parameter:**

  bias: Real number, Unit: Consistent with the vertical scale unit of the channel, and the default unit is V.

- **Return format:**

  The query returns the bias voltage value in scientific notation, and the default unit is V.

- **For example:**

  :CHANnel1:BIAS 1V

  Set the bias voltage for channel 1 to 1V.

  :CHANnel1:BIAS ?

  The query returns 1.00000E+000.

:CHANnel<n>:LABel

- **Command format:**

  :CHANnel<n>:LABel <label>

  :CHANnel<n>:LABel ?

- **Function description:**

  Set or query the channel labels.

- **Parameter:**

  label: Channel name string

- **Return format:**

  The query returns the channel name string.

- **For example:**

  :CHANnel1:LABel “abcd” Set the label of channel 1 as abcd.

  :CHANnel1:LABel ? The query returns abcd.

Cursor

:CURSor:ACTive

- **Command format:**

  :CURSor:ACTive <active>

  :CURSor:ACTive ?

- **Function description:**

  Set or query whether the cursor is enabled.

- **Parameter:**

  active: Boolean type {ON | OFF} or {1 | 0}

- **Return format:**

  The query returns that "1" and "0" represent "ON" and "OFF" respectively.

- **For example:**

  :CURSor:ACTive ON Set to open the cursor.

  :CURSor:ACTive ? The query returns 1.

:CURSor:TYPe

- **Command format:**

  :CURSor:TYPe <type>

  :CURSor:TYPe ?

- **Function description:**

  Set or query the cursor type in the cursor mode.

- **Parameter:**

  Type: Discrete {AMPLitude | TIMe | SCReen | CLOSe}

  AMPLitude: Amplitude (Horizontal)

  TIMe: Time (Vertical)

  SCReen: Screen (Horizontal & Vertical)

  CLOSe: Close

- **Return format:**

  The query returns "AMPLitude", "TIMe", "SCReen", "CLOSe".

- **For example:**

  :CURSor:TYPe AMPLitude Set the cursor type as amplitude.

  :CURSor:TYPe ? The query returns AMPLitude.

:CURSor:SOURce

- **Command format:**

  :CURSor:SOURce <source>

  :CURSor:SOURce ?

- **Function description:**

  Set or query the source of the cursor mode.

- **Parameter:**

  Source: Discrete type {C1 | C2 | C3 | C4 | M1 | M2 | M3 | M4 | M5 | M6 | M7 | M8 | R1 | R2 | R3 | R4}

- **Return format:**

  The query returns "C1", "C2", "C3", "C4", "M1", "M2", "M3", "M4", "M5", "M6", "M7", "M8", "R1", "R2", "R3", "R4".

- **For example:**

  :CURSor:SOURce C1 Set the cursor source to Channel 1.

  :CURSor:SOURce ? The query returns C1.

:CURSor:MOVe:SYNC

- **Command format:**

  :CURSor:MOVe:SYNC <enable>

  :CURSor:MOVe:SYNC ?

- **Function description:**

  Set or query whether synchronous cursor movement is enabled.

- **Parameter:**

  enable: Boolean {ON | OFF} or {1 | 0}

- **Return format:**

  The query returns that "1" and "0" represent "ON" and "OFF" respectively.

- **For example:**

  :CURSor:MOVe:SYNC ON

  Enable synchronous cursor movement.

  :CURSor:MOVe:SYNC ?

  The query returns 1.

:CURSor:UNIT:VCURsor

- **Command format:**

  :CURSor:UNIT:VCURsor <unit>

  :CURSor:UNIT:VCURsor ?

- **Function description:**

  Set or query the vertical cursor unit.

- **Parameter:**

  unit:{SECond | DEGRee | PERCent}

  SECond:Second

  DEGRee:Degree

PERCent: Percent

- **Return format:**

  The query returns "SECond", "DEGRee", "PERCent".

- **For example:**

  :CURSor:UNIT:VCURsor SECond Set the vertical cursor unit to seconds.

  :CURSor:UNIT:VCURsor ? The query returns SECond.

:CURSor:UNIT:HCURsor

- **Command format:**

  :CURSor:UNIT:HCURsor <unit>

  :CURSor:UNIT:HCURsor ?

- **Function description:**

  Set or query the horizontal cursor unit.

- **Parameter:**

  unit:{BASe | PERCent}

  BASe:Be consistent with the unit of the analog channel

  PERCent:Percent

Note: BASE depends on the unit of the channel where the cursor is located.

- **Return format:**

  The query returns "BASe", "PERCent".

- **For example:**

  :CURSor:UNIT:HCURsor BASe Set the horizontal cursor unit to base.

  :CURSor:UNIT:HCURsor ? The query returns BASe.

:CURSor:POSition:MODe

- **Command format:**

  :CURSor:POSition:MODe <mode>

  :CURSor:POSition:MODe ?

- **Function description:**

  Set or query the display type of the cursor position.

- **Parameter:**

  active:Boolean type{ABSolute | RELative}

  ABSolute:The cursor position is an absolute value

RELative:The cursor position is an relative value

- **Return format:**

  The query returns “ABSolute”、“RELative”

- **For example:**

  :CURSor:POSition:MODe ABSolute Set to turn on the cursor position display type as absolute value display.

  :CURSor:POSition:MODe ? Query and return "ABSolute"

:CURSor:CAX

- **Command format:**

  :CURSor:CAX <value>

  :CURSor:CAX ?

- **Function description:**

  Set or query the position for vertical cursor A.

- **Parameter:**

  value: Real number type. When the cursor position type is relative value display, the unit is div. When the cursor position display type is absolute value, it is consistent with the actual horizontal coordinate unit, with s as the default unit.

- **Return format:**

  The query returns the position of the vertical cursor line A.

- **For example:**

  :CURSor:CAX 3 Set vertical cursor A to 3div.

  :CURSor:CAX ? The query returns 3.00000E+000.

:CURSor:CBX

- **Command format:**

  :CURSor:CBX <value>

  :CURSor:CBX ?

- **Function description:**

  Set or query the position for vertical cursor B.

- **Parameter:**

  value: Real number type. When the cursor position type is relative value display, the unit is div. When the cursor position display type is absolute value, it is consistent with the

actual horizontal coordinate unit, with s as the default unit.

- **Return format:**

  The query returns the position of the vertical cursor line B.

- **For example:**

  :CURSor:CBX 3 Set the position of the vertical cursor B to 3div.

  :CURSor:CBX ? The query returns 3.00000E+000.

:CURSor:CAY

- **Command format:**

  :CURSor:CAY <value>

  :CURSor:CAY ?

- **Function description:**

  Set or query the position for horizontal cursor A.

- **Parameter:**

  Value: Real number type. When the cursor position type is relative value display, the unit is div. When the cursor position display type is absolute value, it is consistent with the actual vertical coordinate unit, with V as the default unit.

- **Return format:**

  The query returns the position of the horizontal cursor line A.

- **For example:**

  :CURSor:CAY 3 Set the position of the horizontal cursor A to 3div.

  :CURSor:CAY ? The query returns 3.00000E+000.

:CURSor:CBY

- **Command format:**

  :CURSor:CBY <value>

  :CURSor:CBY ?

- **Function description:**

  Set or query the position for horizontal cursor B.

- **Parameter:**

  Value: Real number type. When the cursor position type is relative value display, the unit is div. When the cursor position display type is absolute value, it is consistent with the actual vertical coordinate unit, with V as the default unit.

- **Return format:**

  Query and return the position of the horizontal cursor line B.

- **For example:**

  :CURSor:CBY 3 Set the position of the horizontal cursor B to 3div.

  :CURSor:CBY ? Query and return 3.00000E+000.

:CURSor:AXValue

- **Command format:**

  :CURSor:AXValue ?

- **Function description:**

  Query the measured value for vertical cursor A.

- **Return format:**

  Query and return the measured value for vertical cursor A in scientific notation.

:CURSor:BXValue

- **Command format:**

  :CURSor:BXValue ?

- **Function description:**

  Query the measured value for vertical cursor B.

- **Return format:**

  The measured value of vertical cursor B is -1.600E-01V.

:CURSor:AYValue

- **Command format:**

  :CURSor:AYValue ?

- **Function description:**

  Query the measured value of horizontal cursor A.

- **Return format:**

  Query and return the measured value of the horizontal cursor A, expressed in scientific notation.

:CURSor:BYValue

- **Command format:**

  :CURSor:BYValue ?

- **Function description:**

  Query the measured value for horizontal cursor B.

- **Return format:**

  Query and return the measured value for horizontal cursor B, expressed in scientific notation.

:CURSor:DATA:MODe

- **Command format:**

  :CURSor:DATA:MODe <active>

  :CURSor:DATA:MODe ?

- **Function description:**

  Set or query whether the cursor data hover is turned on.

- **Parameter:**

  active: Boolean type {ON | OFF} or {1 | 0}

  ON: Suspend

  OFF: Fixed

- **Return format:**

  The query returns that "1" and "0" represent "ON" and "OFF" respectively.

- **For example:**

  :CURSor:DATA:MODe ON Turn on the cursor data hover.

  :CURSor:DATA:MODe ? The query returns 1.

Timebase

:TIMebase:SCALe

- **Command format:**

  :TIMebase:SCALe <scale>

  :TIMebase:SCALe ?

- **Function description:**

  Set or query the main time base scale position.

- **Parameter:**

  scale: Real number type, unit: ps, ns, us, ms, s, ks. If no unit is input, the default unit is s. Setting range: 100ps ~ 1ks.

- **Return format:**

  Query and return the horizontal scale position value in scientific notation. The default unit is s.

- **For example:**

  :TIMebase:SCALe 1s Set the horizontal time base scale to 1s.

  :TIMebase:SCALe ? Query and return 1.0000000000000E+000.

:TIMebase:DELay

- **Command format:**

  :TIMebase:DELay <delay>

  :TIMebase:DELay ?

- **Function description:**

  Set or query the delayed scan.

- **Parameter:**

  delay: Real number type, unit fs, ps, ns, us, ms, s, ks. If the unit is not input, the default unit is s.

- **Return format:**

  Query and return the delayed scan in scientific notation.

- **For example:**

  :TIMebase:DELay 1s

  Set the delayed scan to 1s.

  :TIMebase:DELay ?

  The delayed scan returning 1.0000000000000E+000 is queried.

Trigger

:TRIGger:TYPe

- **Command format:**

  :TRIGger:TYPe <type>

  :TRIGger:TYPe ?

- **Function description:**

  Set or query the type of trigger.

- **Parameter:**

  type: Discrete type {EDGe | PWIDth | VIDeo | PATTern | TIMeout | SETuphold | RUNt | TRANsition | DELay | NEDGe | SUSTaintime | SERial}

  EDGe: edge

  PWIDth: pulsewidth

  VIDeo: video

  PATTern: pattren

  TIMeout: Timeout

  SETuphold: setup&hold

  RUNt:runt

  TRANsition:transition

  DELay:delay

  NEDGe:Nth edge

  SUSTaintime:sustaintime

  SERial: serial bus

- **Return format:**

  The query returns "EDGe", "PWIDth", "VIDeo", "PATTern", "TIMeout", "SETuphold", "RUNt", "TRANsition", "DELay ", "NEDGe", "SUSTaintime", "SERial".

- **For example:**

  :TRIGger:TYPe EDGe Set the trigger type to edge trigger.

  :TRIGger:TYPe ? The query returns EDGe.

:TRIGger:HOLDoff

- **Command format:**

:TRIGger:HOLDoff <holdoff>

:TRIGger:HOLDoff ?

- **Function description:**

  Set or query the trigger holdoff time.

- **Parameter:**

  holdoff: Real type, unit: ns, us, ms, s, the default unit is s.

- **Return format:**

  Query and return the holdoff time, expressed in scientific notation.

- **For example:**

  :TRIGger:HOLDoff 1s Output selection trigger holdoff 1s.

  :TRIGger:HOLDoff ? Query and return 1.0000000000000E+000.

:TRIGger:MODe

- **Command format:**

  :TRIGger:MODe <mode>

  :TRIGger:MODe ?

- **Function description:**

  Set or query the trigger mode.

- **Parameter:**

  mode: Discrete type {AUTO | NORMal | SINGle}

  AUTO:Auto trigger

  NORMal:Normal trigger

  SINGle:Single trigger

- **Return format:**

  Query and return "AUTO", "NORMal", "SINGle".

- **For example:**

  :TRIGger:MODe NORMal Set the trigger mode to the normal mode.

  :TRIGger:MODe ? Query and return NORMal.

Trigger-Edge

:TRIGger:EDGe:COUPling

- **Command format:**

:TRIGger:EDGe:COUPling <coupling>

:TRIGger:EDGe:COUPling ?

- **Function description:**

  Set or query the trigger coupling.

- **Parameter:**

  coupling: discrete type {DC | AC | LFReject | HFReject | NREJect}

  DC: Direct Current

  AC: Alternating Current

  LFReject: Low Frequency Rejection

  HFReject: High Frequency Rejection

  NREJect: Noise Rejection

- **Return format:**

  Query and return "DC", "AC", "LFReject", "HFReject", "NREJect".

- **For example:**

  :TRIGger:EDGe:COUPling DC Set the trigger coupling to DC.

  :TRIGger:EDGe:COUPling ? Query and return DC.

:TRIGger:EDGe:SOURce

- **Command format:**

  :TRIGger:EDGe:SOURce <source>

  :TRIGger:EDGe:SOURce?

- **Function description:**

  Set or query the source for the edge trigger.

- **Parameter:**

  Source: Discrete type {C1 | C2 | C3 | C4 | D0 | D1 | D2 | D3 | D4 | D5 | D6 | D7 | D8 | D9 | D10 | D11 | D12 | D13 | D14 | D15 | Ext | Ext5 | AC | AUXin}

- **Return format:**

  The query returns "C1", "C2", "C3", "C4", "D0", "D1", "D2", "D3", "D4", "D5", "D6", "D7", "D8", "D9", "D10", "D11", "D12", "D13", "D14", "D15", "Ext", "Ext5", "AC", "AUXin".

- **For example:**

  :TRIGger:EDGe:SOURce C1 Set the edge trigger source to Channel 1.

  :TRIGger:EDGe:SOURce ? The query returns C1.

:TRIGger:EDGe:SLOPe

- **Command format:**

  :TRIGger:EDGe:SLOPe <type>

  :TRIGger:EDGe:SLOPe ?

- **Function description:**

  Set or query the edge type for the trigger.

- **Parameter:**

  Type: Discrete {RISe | FALL | ANY}

  RISe: Rising edge

  FALL: Falling edge

  ANY: Any edge

- **Return format:**

  The query returns "RISe", "FALL", "ANY".

- **For example:**

  :TRIGger:EDGe:SLOPe RISe Set the edge trigger edge type to rising edge.

  :TRIGger:EDGe:SLOPe ? The query returns RISe.

:TRIGger:EDGe:LEVel

- **Command format:**

  :TRIGger:EDGe:LEVel <level>

  :TRIGger:EDGe:LEVel ?

- **Function description:**

  Set or query the edge trigger level.

- **Parameter:**

  level: Real number, Unit: Consistent with the vertical scale unit of the channel. The default unit is V.

- **Return format:**

  The query returns the trigger level in scientific notation.

- **For example:**

  :TRIGger:EDGe:LEVel 1V Set the edge trigger level to 1V.

  :TRIGger:EDGe:LEVel ? The query returns 1.00000E+000.

:TRIGger:EDGe:EXTern:IMPedance

- **Command format:**

  :TRIGger:EDGe:EXTern:IMPedance <impedance >

  :TRIGger:EDGe:EXTern:IMPedance ?

- **Function description:**

  Set or query the input impedance of the external trigger.

- **Parameter:**

  impedance: Discrete type {HIGH | LOW}

  HIGH:1MΩ

  LOW:50Ω

- **Return format:**

  Query and return the input impedance of the external trigger.

- **For example:**

  :TRIGger:EDGe:EXTern:IMPedance HIGH

  Set the input impedance of the external trigger to 1MΩ.

  :TRIGger:EDGe:EXTern:IMPedance ?

  Query and return HIGH.

Trigger-Setup&Hold

:TRIGger:SHOLd:DSOUrce

- **Command format:**

  :TRIGger:SHOLd:DSOUrce <source>

  :TRIGger:SHOLd:DSOUrce ?

- **Function description:**

  Set or query the setup&hold data source.

- **Parameter:**

  Source: Discrete {C1 | C2 | C3 | C4}

- **Return format:**

  Query and return "C1", "C2", "C3", "C4".

- **For example:**

  :TRIGger:SHOLd:DSOUrce C1

Set the data source for setup and hold to channel 1.

:TRIGger:SHOLd:DSOUrce ?

Query returns C1.

Note: The data source and clock source must be different !

:TRIGger:SHOLd:CSOUrce

- **Command format:**

  :TRIGger:SHOLd:CSOUrce <source>

  :TRIGger:SHOLd:CSOUrce ?

- **Function description:**

  Set or query the setup&hold of the clock source.

- **Parameter:**

  source: Discrete{C1 | C2 | C3 | C4}

- **Return format:**

  The query returns "C1", "C2", "C3", "C4".

- **For example:**

  :TRIGger:SHOLd:CSOUrce C2

  Set the setup&hold clock source as Channel 2.

  :TRIGger:SHOLd:CSOUrce ?

  The query returns C2.

Note: The data source and the clock source cannot be the same !

:TRIGger:SHOLd:TYPe

- **Command format:**

  :TRIGger:SHOLd:TYPe <type>

  :TRIGger:SHOLd:TYPe ?

- **Function description:**

  Set or query the clock edge for the setup and hold trigger.

- **Parameter:**

  type: Discrete type {RISe | FALL}

  RISe: Rising edge

  FALL: Falling edge

- **Return format:**

  The query returns "RISe", "FALL".

- **For example:**

  :TRIGger:SHOLd:TYPe RISe

  Set the clock edge for setup&hold trigger to the rising edge.

  :TRIGger:SHOLd:TYPe ?

  The query returns RISE.

:TRIGger:SHOLd:CLEVel

- **Command format:**

  :TRIGger:SHOLd:CLEVel <value>

  :TRIGger:SHOLd:CLEVel?

- **Function description:**

  Set or query the clock level for setup&hold trigger.

- **Parameter:**

  Value: Real number,Unit: Consistent with the vertical scale unit of the channel, and the default unit is V.

- **Return format:**

  The query returns the clock level for setup&hold trigger.

- **For example:**

  :TRIGger:SHOLd:CLEVel 1V

  Set the clock level for the setup and hold trigger to 1V.

  :TRIGger:SHOLd:CLEVel ?

  The query returns 1.00000E+000.

:TRIGger:SHOLd:DLEVel

- **Command format:**

  :TRIGger:SHOLd:DLEVel <value>

  :TRIGger:SHOLd:DLEVel?

- **Function description:**

  Set or query the data level for the setup and hold trigger.

- **Parameter:**

  Value: Real number,Unit: Consistent with the vertical scale unit of the channel, and the

default unit is V.

- **Return format:**

  The query returns the data level for setup&hold trigger.

- **For example:**

  :TRIGger:SHOLd:DLEVel 1V

  Set the data level for setup&hold trigger to 1V.

  :TRIGger:SHOLd:DLEVel ?

  The query returns 1.00000E+000.

:TRIGger:SHOLd:PATTern

- **Command format:**

  :TRIGger:SHOLd:PATTern <type>

  :TRIGger:SHOLd:PATTern ?

- **Function description:**

  Set or query the pattern condition for the setup and hold trigger.

- **Parameter:**

  type: Discrete type {HIGH | LOW}

  HIGH:High level

  LOW:Low level

- **Return format:**

  The query returns "HIGH", "LOW".

- **For example:**

  :TRIGger:SHOLd:PATTern HIGH

  Set the pattern condition of setup&hold trigger as HIGH.

  :TRIGger:SHOLd:PATTern ?

  The query returns HIGH.

:TRIGger:SHOLd:VIOLation

- **Command format:**

  :TRIGger:SHOLd:VIOLation <type>

  :TRIGger:SHOLd:VIOLation ?

- **Function description:**

  Set or query the trigger condition of the setup&hold trigger.

- **Parameter:**

  type: Discrete type {SETup | HOLD | SETHold}

  SETup:Setup

  HOLD:Hold

  SETHold:Setup&Hold

- **Return format:**

  The query returns "SETup", "HOLD", "SETHold".

- **For example:**

  :TRIGger:SHOLd:VIOLation SETup

  The trigger condition for setting setup&hold trigger is SETUP.

  :TRIGger:SHOLd:VIOLation ?

  The query returns SETup.

:TRIGger:SHOLd:SETTime

- **Command format:**

  :TRIGger:SHOLd:SETTime <value>

  :TRIGger:SHOLd:SETTime ?

- **Function description:**

  Set or query the setup time of the setup&hold trigger.

- **Parameter:**

  Value: Real number, Unit: ns, us, ms, s. If no unit is input, the default unit is s.

- **Return format:**

  The query returns the setup time of the setup&hold trigger.

- **For example:**

  :TRIGger:SHOLd:SETTime 1s

  Set the setup time of the setup&hold trigger to 1s.

  :TRIGger:SHOLd:SETTime ?

  The query returns 1.0000000000000E+000.

:TRIGger:SHOLd:HOLDtime

- **Command format:**

  :TRIGger:SHOLd:HOLDtime <value>

  :TRIGger:SHOLd:HOLDtime ?

- **Function description:**

  Set or query the hold time of the hold trigger.

- **Parameter:**

  Value: Real number, unit: ns, us, ms, s. If no unit is input, the default unit is s.

- **Return format:**

  The query returns the hold time for the hold trigger.

- **For example:**

  :TRIGger:SHOLd:HOLDtime 1

  Set the hold time of the hold trigger to 1s.

  :TRIGger:SHOLd:HOLDtime ?

  The query returns 1.0000000000000E+000.

Trigger-Pattern

:TRIGger:PATTern:LOGic:ALLCondition

- **Command format:**

  :TRIGger:PATTern:LOGic:ALLCondition <condition>

- **Function description:**

  Set the trigger conditions for all channels in the pattern trigger logic.

- **Parameter:**

  condition: Discrete type {HIGH | LOW | ANY | RISe | FALL}

  HIGH: High

  LOW: Low

  ANY: Any

  RISe: Rising edge

  FALL: Falling edge

- **For example:**

  :TRIGger:PATTern:LOGic:ALLCondition HIGH

  Set the trigger conditions for all channels to HIGH.

:TRIGger:PATTern:LOGic:CONDition:C<n>

- **Command format:**

  :TRIGger:PATTern:LOGic:CONDition:C<n> <type>

:TRIGger:PATTern:LOGic:CONDition:C<n> ?

- **Function description:**

  Set or query the trigger condition of a single analog channel for the pattern trigger logic qualification.

- **Parameter:**

  type: Discrete type {HIGH | LOW | ANY | RISe | FALL}

  HIGH: High

  LOW: Low

  ANY: Any

  RISe: Rising edge

  FALL: Falling edge

- **Return format:**

  The query returns "HIGH", "LOW", "ANY", "RISe", "FALL".

- **For example:**

  :TRIGger:PATTern:LOGic:CONDition:C1 HIGH Set the trigger condition of C1 as HIGH.

  :TRIGger:PATTern:LOGic:CONDition:C1 ? The query returns HIGH.

:TRIGger:PATTern:LOGic:GATe:C<n>

- **Command format:**

  :TRIGger:PATTern:LOGic:GATe:C<n> <value>

  :TRIGger:PATTern:LOGic:GATe:C<n> ?

- **Function description:**

  Set or query the single analog channel threshold for the pattern trigger logic qualification.

- **Parameter:**

  Value: Real number, Unit: Consistent with the vertical scale unit of the channel, and the default unit is V.

- **Return format:**

  Query and return the single analog channel threshold of the pattern trigger logic qualification.

- **For example:**

  :TRIGger:PATTern:LOGic:GATe:C1 1V Set the trigger threshold of C1 to 1V.

  :TRIGger:PATTern:LOGic:GATe:C1 ? Query and return 1.00000E+000.

Trigger-Pulse width

:TRIGger:PWIDth:CONDition

- **Command format:**

  :TRIGger:PWIDth:CONDition <condition>

  :TRIGger:PWIDth:CONDition ?

- **Function description:**

  Set or query the condition for the pulse width trigger.

- **Parameter:**

  condition: Discrete type {GREaterthan | LESSthan | RANGe}

  GREaterthan: Greater than

  LESSthan: Less than

  RANGe: Within the range

- **Return format:**

  The query returns "GREaterthan", "LESSthan", "RANGe"

- **For example:**

  :TRIGger:PWIDth:CONDition GREaterthan

  Set the condition for the pulse width trigger to GREaterthan.

  :TRIGger:PWIDth:CONDition ?

  The query returns GREaterthan.

:TRIGger:PWIDth:TIMe:LOWer

- **Command format:**

  :TRIGger:PWIDth:TIMe:LOWer <time>

  :TRIGger:PWIDth:TIMe:LOWer ?

- **Function description:**

  Set or query the lower limit of the pulse width for the pulse width trigger.

- **Parameter:**

  time: Real number, unit: ns, us, ms, s. If the unit is not entered, the default unit is s.

- **Return format:**

  Query and return the lower limit for the pulse width, expressed in scientific notation.

- **For example:**

:TRIGger:PWIDth:TIMe:LOWer 1

Set the lower limit for the trigger pulse width to 1s.

:TRIGger:PWIDth:TIMe:LOWer ?

Query returns 1.0000000000000E+000.

:TRIGger:PWIDth:TIMe:UPPer

- **Command format:**

  :TRIGger:PWIDth:TIMe:UPPer <time>

  :TRIGger:PWIDth:TIMe:UPPer ?

- **Function description:**

  Set or query the upper limit for the pulse width trigger.

- **Parameter:**

  time: Real number, unit: ns, us, ms, s. If the unit is not input, the default unit is s.

- **Return format:**

  Query returns the upper limit value of the pulse width in seconds and expressed in scientific notation.

- **For example:**

  :TRIGger:PWIDth:TIMe:UPPer 1

  Set the upper limit for the pulse width trigger to 1 second.

  :TRIGger:PWIDth:TIMe:UPPer ?

  Query returns 1.0000000000000E+000.

:TRIGger:PWIDth:POLarity

- **Command format:**

  :TRIGger:PWIDth:POLarity <polarity>

  :TRIGger:PWIDth:POLarity ?

- **Function description:**

  Set or query the polarity of the pulse width trigger.

- **Parameter:**

  polarity: Discrete type {POSitive | NEGative}

  POSitive: Positive pulse width

  NEGative: Negative pulse width

- **Return format:**

The query returns "POSitive", "NEGative".

- **For example:**

  :TRIGger:PWIDth:POLarity POSitive

  Set the polarity of the pulse width trigger to positive.

  :TRIGger:PWIDth:POLarity ?

  The query returns POSitive.

:TRIGger:PWIDth:SOURce

- **Command format:**

  :TRIGger:PWIDth:SOURce <source>

  :TRIGger:PWIDth:SOURce ?

- **Function description:**

  Set or query the pulse width trigger source.

- **Parameter:**

  Source: Discrete type {C1 | C2 | C3 | C4 | D0 | D1 | D2 | D3 | D4 | D5 | D6 | D7 | D8 | D9 | D10 | D11 | D12 | D13 | D14 | D15}

- **Return format:**

  The query returns "C1", "C2", "C3", "C4", "D0", "D1", "D2", "D3", "D4", "D5", "D6", "D7", "D8", "D9", "D10", "D11", "D12", "D13", "D14", "D15 ".

- **For example:**

  :TRIGger:PWIDth:SOURce C1 Set the pulse width trigger source to C1.

  :TRIGger:PWIDth:SOURce ? The query returns C1.

:TRIGger:PWIDth:LEVel

- **Command format:**

  :TRIGger:PWIDth:LEVel <value>

  :TRIGger:PWIDth:LEVel ?

- **Function description:**

  Set or query the clock level of the pulse width trigger.

- **Parameter:**

  Value: Real number type, Unit: Consistent with the vertical scale unit of the channel, and the default unit is V.

- **Return format:**

  The query returns the clock level of the pulse width trigger.

- **For example:**

  :TRIGger:PWIDth:LEVel 10mV

  Set the clock level of the pulse width trigger to 10mV.

  :TRIGger:PWIDth:LEVel ?

  The query returns 1.00000E-002.

Trigger-Runt

:TRIGger:RUNT:SOURce

- **Command format:**

  :TRIGger:RUNT:SOURce <source>

  :TRIGger:RUNT:SOURce ?

- **Function description:**

  Set or query the source for the under-voltage trigger.

- **Parameter:**

  source: Discrete type {C1 | C2 | C3 | C4}

- **Return format:**

  The query returns "C1", "C2", "C3", "C4".

- **For example:**

  :TRIGger:RUNT:SOURce C1 Set the under-voltage trigger source to C1.

  :TRIGger:RUNT:SOURce ? The query returns C1.

:TRIGger:RUNT:CONDition

- **Command format:**

  :TRIGger:RUNT:CONDition <type>

  :TRIGger:RUNT:CONDition ?

- **Function description:**

  Set or query the under-voltage trigger conditions.

- **Parameter:**

  type: Discrete type {GREaterthan | LESSthan | RANGe | NRANge}

  GREaterthan : Greater than

  LESSthan: Less than

RANGe: Within the range

NRANge: Outside the range

- **Return format:**

  The query returns "GREaterthan", "LESSthan", "RANGe", "NRANge".

- **For example:**

  :TRIGger:RUNT:CONDition GREaterthan

  Set the under-voltage trigger condition to Greater than.

  :TRIGger:RUNT:CONDition ?

  The query returns Greater than.

:TRIGger:RUNT:LEVel:UPPer

- **Command format:**

  :TRIGger:RUNT:LEVel:UPPer <value>

  :TRIGger:RUNT:LEVel:UPPer ?

- **Function description:**

  Set or query the upper level for the under-voltage trigger.

- **Parameter:**

  Value: Real number type, Unit: Consistent with the vertical scale unit of the channel, and the default unit is V.

- **Return format:**

  The query returns the high level of the under-voltage trigger.

- **For example:**

  :TRIGger:RUNT:LEVel:UPPer 1

  Set the upper level for the under-voltage trigger to 1V.

  :TRIGger:RUNT:LEVel:UPPer ?

  The query returns 1.00000E+000.

:TRIGger:RUNT:LEVel:LOWer

- **Command format:**

  :TRIGger:RUNT:LEVel:LOWer <value>

  :TRIGger:RUNT:LEVel:LOWer ?

- **Function description:**

  Set or query the lower level for the under-voltage trigger.

- **Parameter:**

  Value: Real number type,Unit: Consistent with the vertical scale unit of the channel, and the default unit is V.

- **Return format:**

  The query returns the low level of the under-voltage trigger.

- **For example:**

  :TRIGger:RUNT:LEVel:LOWer 1

  Set the lower level for the under-voltage trigger to 1V.

  :TRIGger:RUNT:LEVel:LOWer ?

  The query returns 1.00000E+000.

:TRIGger:RUNT:POLarity

- **Command format:**

  :TRIGger:RUNT:POLarity <polarity>

  :TRIGger:RUNT:POLarity ?

- **Function description:**

  Set or query the polarity of the under-voltage trigger.

- **Parameter:**

  Polarity: Discrete type {POSitive | NEGative}

  POSitive: Positive electrode

  NEGative: Negative electrode

- **Return format:**

  The query returns "POSitive", "NEGative".

- **For example:**

  :TRIGger:RUNT:POLarity POSitive

  Set the under-voltage trigger polarity to positive polarity.

  :TRIGger:RUNT:POLarity ?

  The query returns POSitive.

:TRIGger:RUNT:TIMe:UPPer

- **Command format:**

  :TRIGger:RUNT:TIMe:UPPer <time>

  :TRIGger:RUNT:TIMe:UPPer ?

- **Function description:**

  Set or query the upper time limit for the under-voltage trigger.

- **Parameter:**

  time: Real number type, unit: ns, us, ms, s. If the unit is not input, the default unit is s.

- **Return format:**

  Query and return the upper limit of the under-voltage time of the under-voltage trigger.

- **For example:**

  :TRIGger:RUNT:TIMe:UPPer 1

  Set the upper time limit for the under-voltage trigger to 1s.

  :TRIGger:RUNT:TIMe:UPPer ?

  Query and return 1.0000000000000E+000.

:TRIGger:RUNT:TIMe:LOWer

- **Command format:**

  :TRIGger:RUNT:TIMe:LOWer <time>

  :TRIGger:RUNT:TIMe:LOWer ?

- **Function description:**

  Set or query the lower time limit for the under-voltage trigger.

- **Parameter:**

  time: Real number type, unit: ns, us, ms, s. If the unit is not input, the default unit is s.

- **Return format:**

  Query and return the lower limit of the under-voltage time of the under-voltage trigger.

- **For example:**

  :TRIGger:RUNT:TIMe:LOWer 1

  Set the lower time limit for the under-voltage trigger to 1s.

  :TRIGger:RUNT:TIMe:LOWer ?

  Query and return 1.0000000000000E+000.

Trigger-Slope

:TRIGger:SLOPe:SOURce

- **Command format:**

  :TRIGger:SLOPe:SOURce <source>

:TRIGger:SLOPe:SOURce ?

- **Function description:**

  Set or query the source for the slope trigger.

- **Parameter:**

  source: Discrete {C1 | C2 | C3 | C4}

- **Return format:**

  Query and return "C1", "C2", "C3", "C4".

- **For example:**

  :TRIGger:SLOPe:SOURce C1 Set the slope trigger source to C1.

  :TRIGger:SLOPe:SOURce ? Query returns C1.

:TRIGger:SLOPe:CONDition

- **Command format:**

  :TRIGger:SLOPe:CONDition <type>

  :TRIGger:SLOPe:CONDition ?

- **Function description:**

  Set or query the slope trigger conditions.

- **Parameter:**

  type: Discrete type {GREaterthan | LESSthan | RANGe}

  GREaterthan: Greater than

  LESSthan: Less than

  RANGe: Within the range

- **Return format:**

  The query returns "GREaterthan", "LESSthan", "RANGe".

- **For example:**

  :TRIGger:SLOPe:CONDition GREaterthan

  Set the slope trigger condition as GREaterthan.

  :TRIGger:SLOPe:CONDition ?

  The query returns GREaterthan.

:TRIGger:SLOPe:LEVel:UPPer

- **Command format:**

  :TRIGger:SLOPe:LEVel:UPPer <value >

:TRIGger:SLOPe:LEVel:UPPer ?

- **Function description:**

  Set or query the upper level for the slope trigger.

- **Parameter:**

  value : Real number type, Unit: Consistent with the vertical scale unit of the channel, and the default unit is V.

- **Return format:**

  The query returns the high level of the slope trigger.

- **For example:**

  :TRIGger:SLOPe:LEVel:UPPer 1

  Set the upper level for the slope trigger to 1V.

  :TRIGger:SLOPe:LEVel:UPPer ?

  The query returns 1.00000E+000.

:TRIGger:SLOPe:LEVel:LOWer

- **Command format:**

  :TRIGger:SLOPe:LEVel:LOWer <value>

  :TRIGger:SLOPe:LEVel:LOWer ?

- **Function description:**

  Set or query the low level of the slope trigger.

- **Parameter:**

  Value: Real number type, Unit: Consistent with the vertical scale unit of the channel, and the default unit is V.

- **Return format:**

  Query and return the low level of the slope trigger.

- **For example:**

  :TRIGger:SLOPe:LEVel:LOWer 1

  Set the low level of the slope trigger to 1V.

  :TRIGger:SLOPe:LEVel:LOWer ?

  Query and return 1.00000E+000.

:TRIGger:SLOPe:SLOPe

- **Command format:**

:TRIGger:SLOPe:SLOPe <type>

:TRIGger:SLOPe:SLOPe ?

- **Function description:**

  Set or query the trigger slope type.

- **Parameter:**

  type: Discrete type {RISe | FALL}

  RISe: Rising edge

  FALL: Falling edgel

- **Return format:**

  Query and return "RISe", "FALL".

- **For example:**

  :TRIGger:SLOPe:SLOPe RISe Set the slope trigger type to RISe.

  :TRIGger:SLOPe:SLOPe ? Query and return RISe.

:TRIGger:SLOPe:TIMe:UPPer

- **Command format:**

  :TRIGger:SLOPe:TIMe:UPPer <time>

  :TRIGger:SLOPe:TIMe:UPPer ?

- **Function description:**

  Set or query the upper time limit for the slope trigger.

- **Parameter:**

  time: Real number type, unit: ns, us, ms, s. If no unit is entered, the default unit is s.

- **Return format:**

  Query and return the upper limit of the slope time for the slope trigger.

- **For example:**

  :TRIGger:SLOPe:TIMe:UPPer 1

  Set the upper time limit for the slope trigger to 1s.

  :TRIGger:SLOPe:TIMe:UPPer ?

  Query and return 1.0000000000000E+000.

:TRIGger:SLOPe:TIMe:LOWer

- **Command format:**

  :TRIGger:SLOPe:TIMe:LOWer <time>

:TRIGger:SLOPe:TIMe:LOWer ?

- **Function description:**

  Set or query the lower limit of the slope time for the slope trigger.

- **Parameter:**

  time: Real number type, unit: ns, us, ms, s. If no unit is input, the default unit is s.

- **Return format:**

  Query and return the lower limit of the slope time for the slope trigger.

- **For example:**

  :TRIGger:SLOPe:TIMe:LOWer 1

  Set the lower limit of the slope time for the slope trigger to 1s.

  :TRIGger:SLOPe:TIMe:LOWer ?

  Query and return 1.0000000000000E+000.

Trigger-Timeout

:TRIGger:TIMeout:SOURce

- **Command format:**

  :TRIGger:TIMeout:SOURce <source>

  :TRIGger:TIMeout:SOURce ?

- **Function description:**

  Set or query the source for the timeout trigger.

- **Parameter:**

  source: Discrete type {C1 | C2 | C3 | C4 | D0 | D1 | D2 | D3 | D4 | D5 | D6 | D7 | D8 | D9 | D10 | D11 | D12 | D13 | D14 | D15}

- **Return format:**

  The query returns "C1", "C2", "C3", "C4", "D0", "D1", "D2", "D3", "D4", "D5", "D6", "D7", "D8", "D9", "D10", "D11", "D12", "D13", "D14", "D15 ".

- **For example:**

  :TRIGger:TIMeout:SOURce C1 Set the timeout trigger source to C1.

  :TRIGger:TIMeout:SOURce ? The query returns C1.

:TRIGger:TIMeout:SLOPe

- **Command format:**

:TRIGger:TIMeout:SLOPe <type>

:TRIGger:TIMeout:SLOPe ?

- **Function description:**

  Set or query the timeout edge type.

- **Parameter:**

  Type: Discrete type {POSitive | NEGative | ANY}

  POSitive: Rising edge

  NEGative: Falling edge

  ANY: Any edge

- **Return format:**

  The query returns "POSitive", "NEGative", "ANY".

- **For example:**

  :TRIGger:TIMeout:SLOPe POSitive

  Set the timeout trigger type to the rising edge.

  :TRIGger:TIMeout:SLOPe ?

  The query returns POSitive.

TRIGger:TIMeout:LEVel

- **Command format:**

  :TRIGger:TIMeout:LEVel <level>

  :TRIGger:TIMeout:LEVel ?

- **Function description:**

  Set or query the timeout trigger level.

- **Parameter:**

  Level: Real number type, Unit: Consistent with the vertical scale unit of the channel, and the default unit is V.

- **Return format:**

  Query and return the trigger level of the timeout trigger.

- **For example:**

  :TRIGger:TIMeout:LEVel 1

  Set the timeout trigger level to 1V.

  :TRIGger:TIMeout:LEVel ?

  Query and return 1.00000E+000.

:TRIGger:TIMeout:TIMe

- **Command format:**

  :TRIGger:TIMeout:TIMe <time>

  :TRIGger:TIMeout:TIMe ?

- **Function description:**

  Set or query the duration of the timeout trigger.

- **Parameter:**

  time: Real number type, unit: ns, us, ms, s. If no unit is input, the default unit is s.

- **Return format:**

  Query and return the duration of the timeout trigger.

- **For example:**

  :TRIGger:TIMeout:TIMe 1

  Set the duration of the timeout trigger to 1s.

  :TRIGger:TIMeout:TIMe ?

  Query and return 1.0000000000000E+000.

Trigger-Video

:TRIGger:VIDeo:SOURce

- **Command format:**

  :TRIGger:VIDeo:SOURce <source>

  :TRIGger:VIDeo:SOURce ?

- **Function description:**

  Set or query the source for the video trigger.

- **Parameter:**

  source: Discrete type {C1 | C2 | C3 | C4}

- **Return format:**

  Query returns "C1", "C2", "C3", "C4".

- **For example:**

  :TRIGger:VIDeo:SOURce C1 Set the video trigger source to C1.

  :TRIGger:VIDeo:SOURce ? Query returns C1.

:TRIGger:VIDeo:STANdard

- **Command format:**

  :TRIGger:VIDeo:STANdard <standard>

  :TRIGger:VIDeo:STANdard ?

- **Function description:**

  Set or query the video standard type.

- **Parameter:**

  standard:{NTSC | PAL}

  NTSC:NTSC

  PAL:PAL

- **Return format:**

  The query returns "NTSC", "PAL".

- **For example:**

  :TRIGger:VIDeo:STANdard NTSC Set the video standard type to NTSC.

  :TRIGger:VIDeo:STANdard ? The query returns NTSC.

:TRIGger:VIDeo:SYNC

- **Command format:**

  :TRIGger:VIDeo:SYNC <sync>

  :TRIGger:VIDeo:SYNC ?

- **Function description:**

  Set or query the synchronization mode of video triggering.

- **Parameter:**

  sync:{ODD | EVEN | ALL | SPECified}

  ODD: Even field

  EVEN: Odd field

  ALL: All rows

  SPECified: Specified row

- **Return format:**

  Query and return the synchronization mode of video triggering.

- **For example:**

  :TRIGger:VIDeo:SYNC ALL

Set the synchronization mode of video triggering to all rows.

:TRIGger:VIDeo:SYNC ?

Query and return all rows.

:TRIGger:VIDeo:LINe

- **Command format:**

  :TRIGger:VIDeo:LINe <line>

  :TRIGger:VIDeo:LINe ?

- **Function description:**

  Set or query the number of video rows for the specified row in video triggering.

- **Parameter:**

  Line: Integer type, ranging from 1 to 625

- **Return format:**

  Query and return the number of video rows of the specified row for video triggering.

- **For example:**

  :TRIGger:VIDeo:LINe 10

  Set the number of video rows for the specified row to 10.

  :TRIGger:VIDeo:LINe ?

  Query and return 10.

:TRIGger:VIDeo:LEVel

- **Command format:**

  :TRIGger:VIDeo:LEVel <LEVel >

  :TRIGger:VIDeo:LEVel ?

- **Function description:**

  Set or query the level of video triggering.

- **Parameter:**

  LEVEL : Video trigger level, Unit: Consistent with the vertical scale unit of the channel, and the default unit is V.

- **Return format:**

  Query and return the level value.

- **For example:**

  :TRIGger:VIDeo:LEVel 1 Set the video trigger level to 1V.

:TRIGger:VIDeo:LEVel ?	Query and return `1.00000E+000.

Trigger-Nth edge

:TRIGger:NEDGe:SLOPe

- **Command format:**

  :TRIGger:NEDGe:SLOPe <type>

  :TRIGger:NEDGe:SLOPe ?

- **Function description:**

  Set or query the edge type for the Nth edge trigger.

- **Parameter:**

  type: Discrete type {RISe | FALL}

  RISe: Rising edge

  FALL: Falling edge

- **Return format:**

  Query and return "RISe", "FALL".

- **For example:**

:TRIGger:NEDGe:SLOPe RISe	Set the edge type of the N edge trigger to RISe.
:TRIGger:NEDGe:SLOPe ?	Query and return RISe.

:TRIGger:NEDGe:SOURce

- **Command format:**

  :TRIGger:NEDGe:SOURce <source>

  :TRIGger:NEDGe:SOURce ?

- **Function description:**

  Set or query the trigger source of the Nth edge trigger.

- **Parameter:**

  source: Discrete type {C1 | C2 | C3 | C4| D0 | D1 | D2 | D3 | D4 | D5 | D6 | D7 | D8 | D9 | D10 | D11 | D12 | D13 | D14 | D15}

- **Return format:**

  The query returns "C1", "C2", "C3", "C4", "D0", "D1", "D2", "D3", "D4", "D5", "D6", "D7", "D8", "D9", "D10", "D11", "D12", "D13", "D14", "D15".

- **For example:**

:TRIGger:NEDGe:SOURce C1	Set the Nth edge trigger source as channel 1.
:TRIGger:NEDGe:SOURce ?	The query returns C1.

:TRIGger:NEDGe:DURation

- **Command format:**

  :TRIGger:NEDGe:DURation <time>

  :TRIGger:NEDGe:DURation ?

- **Function description:**

  Set or query the idle time of the Nth edge trigger.

- **Parameter:**

  time: Real number, unit: ns, us, ms, s. If no unit is input, the default unit is s. Setting range: 3.1ns ~ 10s.

- **Return format:**

  The query returns the idle time of the Nth edge trigger.

- **For example:**

  :TRIGger:NEDGe:DURation 1s

  Set the idle time of the N edge trigger to 1s.

  :TRIGger:NEDGe:DURation ?

  The query returns 1.0000000000000E+000.

:TRIGger:NEDGe:LEVel

- **Command format:**

  :TRIGger:NEDGe:LEVel <level>

  :TRIGger:NEDGe:LEVel ?

- **Function description:**

  Set or query the trigger level for the Nth edge.

- **Parameter:**

  level: Real number,Unit: Consistent with the vertical scale unit of the channel, and the default unit is V.

- **Return format:**

  The query returns the Nth edge trigger level value.

- **For example:**

  :TRIGger:NEDGe:LEVel 1V

Set the trigger level for the Nth edge to 1V.

:TRIGger:NEDGe:LEVel ?

The query returns 1.00000E+000.

TRIGger:NEDGe:EDGe

- **Command format:**

  :TRIGger:NEDGe:EDGe <count>

  :TRIGger:NEDGe:EDGe ?

- **Function description:**

  Set or query the number of edges of the Nth edge trigger.

- **Parameter:**

  count: Integer type, the setting range is 1 to 65.536k

- **Return format:**

  Query and return the number of edges.

- **For example:**

  :TRIGger:NEDGe:EDGe 5

  Set the number of edges of the Nth edge trigger to 5.

  :TRIGger:NEDGe:EDGe ?

  Query and return 5.

Trigger-Delay

:TRIGger:DELay:SOURce<n>

- **Command format:**

  :TRIGger:DELay:SOURce<n> <source>

  :TRIGger:DELay:SOURce<n> ?

- **Function description:**

  Set or query the source for the delayed trigger.

- **Parameter:**

  n: Real number {1 | 2}, indicating source 1 and source 2 of the delayed trigger.

  source: Discrete type {C1 | C2 | C3 | C4}

- **Return format:**

  Query and return "C1", "C2", "C3", "C4".

- **For example:**

  :TRIGger:DELay:SOURce1 C1 Set the source for delayed trigger 1 to C1.

  :TRIGger:DELay:SOURce1 ? Query and return C1.

Note: Source 1 and Source 2 cannot be set to the same source.

:TRIGger:DELay:SLOPe<n>

- **Command format:**

  :TRIGger:DELay:SLOPe<n> <type>

  :TRIGger:DELay:SLOPe<n> ?

- **Function description:**

  Set or query the edge type for the delayed trigger.

- **Parameter:**

  n: Real number {1 | 2}, representing Source 1 and Source 2 of the delayed trigger.

  type: Discrete type {RISe | FALL}

  RISe: Rising edge

  FALL: Falling edge

- **Return format:**

  The query returns "RISe", "FALL".

- **For example:**

  :TRIGger:DELay:SLOPe1 FALL

  Set the edge type of the delayed trigger source 1 as FALL.

  :TRIGger:DELay:SLOPe1 ?

  The query returns FALL.

:TRIGger:DELay:LEVel<n>

- **Command format:**

  :TRIGger:DELay:LEVel<n> <level>

  :TRIGger:DELay:LEVel<n> ?

- **Function description:**

  Set or query the source level of the delayed trigger.

- **Parameter:**

  n: Real number {1 | 2}, representing the source 1 and source 2 of the delayed trigger.

  level: Real number, Unit: Consistent with the vertical scale unit of the channel, and the

default unit is V.

- **Return format:**

  Query and return the source level value.

- **For example:**

  :TRIGger:DELay:LEVel1 100mV

  Set the level of the delayed trigger source 1 to 100mV.

  :TRIGger:DELay:LEVel1 ?

  Query and return 1.00000E-001.

:TRIGger:DELay:TIMe:UPPer

- **Command format:**

  :TRIGger:DELay:TIMe:UPPer <time>

  :TRIGger:DELay:TIMe:UPPer ?

- **Function description:**

  Set or query the upper time limit for the delayed trigger.

- **Parameter:**

  time: Real number type, unit: ns, us, ms, s. If no unit is input, the default unit is s.

- **Return format:**

  Query and return the upper limit of the delayed trigger time.

- **For example:**

  :TRIGger:DELay:TIMe:UPPer 1

  Set the upper time limit for the delayed trigger to 1s.

  :TRIGger:DELay:TIMe:UPPer ?

  Query and return 1.0000000000000E+000.

:TRIGger:DELay:TIMe:LOWer

- **Command format:**

  :TRIGger:DELay:TIMe:LOWer <time>

  :TRIGger:DELay:TIMe:LOWer ?

- **Function description:**

  Set or query the lower limit of the delayed trigger time.

- **Parameter:**

  time: Real number type, unit: ns, us, ms, s. If the unit is not entered, the default unit is s.

- **Return format:**

  Query and return the lower limit of the delayed trigger time.

- **For example:**

  :TRIGger:DELay:TIMe:LOWer 1

  Set the lower limit of the delayed trigger time to 1s.

  :TRIGger:DELay:TIMe:LOWer ?

  Query and return 1.0000000000000E+000.

:TRIGger:DELay:CONDition

- **Command format:**

  :TRIGger:DELay:CONDition <type>

  :TRIGger:DELay:CONDition ?

- **Function description:**

  Set or query the condition for the delayed trigger.

- **Parameter:**

  Type: Discrete type {GREaterthan | LESSthan | RANGe | NRANge}

  GREaterthan: Greater than

  LESSthan: Less than

  RANGe: Within the range

  NRANge: Outside the range

- **Return format:**

  The query returns "GREaterthan", "LESSthan", "RANGe", "NRANge".

- **For example:**

  :TRIGger:DELay:CONDition GREaterthan

  Set the condition for the delayed trigger to GREaterthan.

  :TRIGger:DELay:CONDition ?

  The query returns GREaterthan.

Trigger - Duration

:TRIGger:SUSTaintime:CONDtion

- **Command format:**

  :TRIGger:SUSTaintime:CONDition <type>

:TRIGger:SUSTaintime:CONDition ?

- **Function description:**

  Set or query the duration trigger condition.

- **Parameter:**

  type: Discrete type {GREaterthan | LESSthan | RANGe}

  GREaterthan: Greater than

  LESSthan: Less than

  RANGe: Within the range

- **Return format:**

  The query returns "GREaterthan", "LESSthan", "RANGe".

- **For example:**

  :TRIGger:SUSTaintime:CONDition GREaterthan

  Set the duration trigger condition to GREaterthan.

  :TRIGger:SUSTaintime:CONDition ?

  The query returns GREaterthan.

:TRIGger:SUSTaintime:TIMe:UPPer

- **Command format:**

  :TRIGger:SUSTaintime:TIMe:UPPer <time>

  :TRIGger:SUSTaintime:TIMe:UPPer ?

- **Function description:**

  Set or query the upper time limit for the duration trigger.

- **Parameter:**

  time: Real number type, unit: ns, us, ms, s. If the unit is not entered, the default unit is s.

- **Return format:**

  Query and return the upper limit of the duration trigger time.

- **For example:**

  :TRIGger:SUSTaintime:TIMe:UPPer 1

  Set the upper time limit for the duration trigger to 1s.

  :TRIGger:SUSTaintime:TIMe:UPPer ?

  Query and return 1.0000000000000E+000.

:TRIGger:SUSTaintime:TIMe:LOWer

- **Command format:**

  :TRIGger:SUSTaintime:TIMe:LOWer <time>

  :TRIGger:SUSTaintime:TIMe:LOWer ?

- **Function description:**

  Set or query the lower limit of the duration trigger time.

- **Parameter:**

  time: Real number type, unit: ns, us, ms, s. If the unit is not entered, the default unit is s.

- **Return format:**

  Query and return the lower limit of the duration trigger time.

- **For example:**

  :TRIGger:SUSTaintime:TIMe:LOWer 1

  Set the lower limit of the duration trigger time to 1s.

  :TRIGger:SUSTaintime:TIMe:LOWer ?

  Query and return 1.0000000000000E+000.

:TRIGger:SUST:LOGic:ALLCondition

- **Command format:**

  :TRIGger:SUST:LOGic:ALLCondition <condition>

- **Function description:**

  Set the trigger conditions for all channels in duration trigger logic.

- **Parameter:**

  condition: Discrete type {HIGH | LOW | ANY}

  HIGH:High

  LOW:Low

  ANY:Any

- **For example:**

  :TRIGger:SUST:LOGic:ALLCondition HIGH

  Set the trigger conditions for all channels to HIGH.

:TRIGger:SUST:LOGic:CONDition:C<n>

- **Command format:**

:TRIGger:SUST:LOGic:CONDition:C<n> <type>

:TRIGger:SUST:LOGic:CONDition:C<n> ?

- **Function description:**

  Set or query the trigger condition for a single analog channel in duration trigger logic.

- **Parameter:**

  type: Discrete type {HIGH | LOW | ANY}

  HIGH:High

  LOW:Low

  ANY:Any

- **Return format:**

  The query returns "HIGH", "LOW", "ANY".

- **For example:**

  :TRIGger:SUST:LOGic:CONDition:C1 HIGH

  Set the trigger condition for a single analog channel to HIGH.

  :TRIGger:SUST:LOGic:CONDition:C1 ?

  The query returns HIGH.

:TRIGger:SUST:LOGic:GATe:C<n>

- **Command format:**

  :TRIGger:SUST:LOGic:GATe:C<n> <value>

  :TRIGger:SUST:LOGic:GATe:C<n> ?

- **Function description:**

  Set or query the single analog channel threshold for the duration trigger logic qualification.

- **Parameter:**

  Value: Real number, Unit: Consistent with the vertical scale unit of the channel, and the default unit is V.

- **Return format:**

  Query and return the single analog channel threshold for the duration trigger logic qualification.

- **For example:**

  :TRIGger:SUST:LOGic:GATe:C1 1V

  Set the trigger threshold of the single analog channel to 1V.

  :TRIGger:SUST:LOGic:GATe:C1 ?

Query and return 1.00000E+000.

BUS

:TRIGger:BUS

- **Command format:**

  :TRIGger:BUS <type>

  :TRIGger:BUS

- **Function description:**

  Set or query the bus channel of the bus trigger.

- **Parameter:**

  type: Discrete type {B1 | B2}

  B1:BUS1

  B2:BUS2

- **Return format:**

  The query returns the bus channel of the bus trigger.

- **For example:**

  :TRIGger:BUS B1

  Set the trigger channel of the bus trigger to B1.

  :TRIGger:BUS ?

  The query returns B1.

RS232

:TRIGger:RS232:CONDition

- **Command format:**

  :TRIGger:RS232:CONDition <type>

  :TRIGger:RS232:CONDition ?

- **Function description:**

  Set or query the RS232 trigger conditions.

- **Parameter:**

  type: Discrete type {START | END | ERROR | DATA}

  STARt: Start bit

END: Stop bit

ERRor: Checksum error

DATA: Data

- **Return format:**

  The query returns the trigger conditions of RS232.

- **For example:**

  :TRIGger:RS232:CONDition DATA

  Set the RS232 trigger condition to DATA.

  :TRIGger:RS232:CONDition ?

  The query returns DATA.

:TRIGger:RS232:DATA

- **Command format:**

  :TRIGger:RS232:DATA <value>

  :TRIGger:RS232:DATA ?

- **Function description:**

  Set or query the data value when the RS232 trigger condition is data.

- **Parameter:**

  value: The binary string data represented by the parameter of 0 or 1 is related to the value set by the :BUS<n>:RS232:WIDTh instruction, and the range is: 0 - 2^n - 1, where n is the current data width.

- **Return format:**

  The query returns binary string data.

- **For example:**

  :TRIGger:RS232:DATA "01111111"

  Set the RS232 trigger data value as 0x7F.

  :TRIGger:RS232:DATA ?

  The query returns 01111111.

I2C

:TRIGger:I2C:CONDition

- **Command format:**

:TRIGger:I2C:CONDition <type>

:TRIGger:I2C:CONDition ?

- **Function description:**

  Set or query the I2C trigger conditions.

- **Parameter:**

  type: Discrete type {START | RESTART | STOP | LOST | ADDRESS | DATA | ADDData}

  STARt:Start bit

  RESTart:Restart

  STOP:Stop bit

  LOST:Response failure

  ADDRess:Address

  DATA:Data

  ADDAta:Address and data

- **Return format:**

  The query returns the trigger conditions of I2C.

- **For example:**

  :TRIGger:I2C:CONDition DATA

  Set the trigger condition of RS232 as DATA.

  :TRIGger:I2C:CONDition ?

  The query returns DATA.

:TRIGger:I2C:DATA

- **Command format:**

  :TRIGger:I2C:DATA <value>

  :TRIGger:I2C:DATA ?

- **Function description:**

  Set or query the data value when the I2C trigger condition is data and address data.

- **Parameter:**

  value: The binary string data represented by the parameter of 0 or 1.

- **Return format:**

  The query returns binary string data.

- **For example:**

  :TRIGger:I2C:DATA "10001110"

Set the I2C trigger data value as 0x8E.

:TRIGger:I2C:DATA ?

The query returns 10001110.

:TRIGger:I2C:DIRection

- **Command format:**

  :TRIGger:I2C:DIRection <direction>

  :TRIGger:I2C:DIRection ?

- **Function description:**

  Set or query the data direction when the I2C trigger condition is address or address data.

- **Parameter:**

  direction: Discrete type {WRITe | READ}

  WRITe:Write

  READ:Read

- **Return forma:**

  The query returns the data direction when the I2C trigger condition is address and address data.

- **For example:**

  :TRIGger:I2C:DIRection WRITe

  Set the I2C trigger data direction as WRITe.

  :TRIGger:I2C:DIRection ?

  The query returns WRITe.

:TRIGger:I2C:ADDRess

- **Command format:**

  :TRIGger:I2C:ADDRess <value>

  :TRIGger:I2C:ADDRess ?

- **Function description:**

  Set or query the address value for the I2C trigger condition.

- **Parameter:**

  value: The binary string data represented by the parameter of 0 or 1.

- **Return format:**

  The query returns binary string data.

- **For example:**

  :TRIGger:I2C:ADDRess "100101"

  Set the I2C trigger address value to 0x25.

  :TRIGger:I2C:ADDRess ?

  The query returns 100101.

:TRIGger:I2C:BYTe

- **Command format:**

  :TRIGger:I2C:BYTe <value>

  :TRIGger:I2C:BYTe ?

- **Function description:**

  Set or query the number of decoded data bytes。

- **Parameter:**

  value:INTEGER range: 0-5。

- **Return format:**

  Query the number of decoded data bytes returned。

- **For example:**

  :TRIGger:I2C:BYTe 2 Set the number of I2C data bytes to 2

  :TRIGger:I2C:BYTe ? Query returns 2

SPI

:TRIGger:SPI:CONDition

- **Command format:**

  :TRIGger:SPI:CONDition <condition>

  :TRIGger:SPI:CONDition ?

- **Function description:**

  Set or query the trigger condition for SPI.

- **Parameter:**

  condition: Discrete type {START | DATA}

  START: start

DATA: Data

- **Return format:**

  The query returns the SPI trigger condition.

- **For example:**

  :TRIGger:SPI:CONDition DATA

  Set the SPI trigger condition to DATA.

  :TRIGger:SPI:CONDition ?

  The query returns DATA.

:TRIGger:SPI:DATA

- **Command format:**

  :TRIGger:SPI:DATA <value>

  :TRIGger:SPI:DATA ?

- **Function description:**

  Set or query the data value when the SPI trigger condition is data.

- **Parameter:**

  value: The binary string data represented by the parameter of 0 or 1, its range is related to the number of frames set by the :TRIGger:SPI:FRAMecount instruction.

- **Return format:**

  The query returns binary string data.

- **For example:**

  :TRIGger:SPI:DATA "10001"

  Set the SPI trigger data value as 0x001.

  :TRIGger:SPI:DATA ?

  The query returns 10001.

:TRIGger:SPI:FRAMecount

- **Command format:**

  :TRIGger:SPI:FRAMecount <value>

  :TRIGger:SPI:FRAMecount ?

- **Function description:**

  Set or query the frame number when the SPI trigger condition is data.

- **Parameter:**

value: Integer type

- **Return format:**

  The query returns the frame number when the SPI trigger condition is data.

- **For example:**

  :TRIGger:SPI:FRAMecount 2

  Set the SPI trigger frame number as 2.

  :TRIGger:SPI:FRAMecount ?

  The query returns 2.

:TRIGger:SPI:WIDTh

- **Command format:**

  :TRIGger:SPI:WIDTh <value>

  :TRIGger:SPI:WUDTh ?

- **Function description:**

  Set or query the data bit width when the SPI trigger condition is data。

- **Parameter:**

  value:INTEGER range: 1-64

- **Return format:**

  Query the data bit width when the SPI trigger condition is data。

- **For example:**

  :TRIGger:SPI:WIDTh 2 When setting the SPI trigger condition to data, the data bit width is 2

  :TRIGger:SPI:WIDTh ? Query returns 2

CAN

:TRIGger:CAN:CONDition

- **Command format:**

  :TRIGger:CAN:CONDition <condition>

  :TRIGger:CAN:CONDition ?

- **Function description:**

  Set or query the trigger mode for CAN.

- **Parameter:**

  condition: Discrete type {STARtframe | TYPeframe | ID | DATA | IDDAta | ENDFrame | ERRor}

  STARtframe: Frame start

  TYPeframe: Frame type

  ID: ID

  DATA: Data

  IDDAta: ID and data

  ENDFrame: Frame end

  ERRor: Error

- **Return format:**

  The query returns the CAN trigger mode.

- **For example:**

  :TRIGger:CAN:CONDition DATA

  Set the CAN trigger mode to DATA.

  :TRIGger:CAN:CONDition ?

  The query returns DATA.

:TRIGger:CAN:DATA

- **Command format:**

  :TRIGger:CAN:DATA <value>

  :TRIGger:CAN:DATA ?

- **Function description:**

  Set or query the data value when the CAN trigger condition is data or ID and data.

- **Parameter:**

  value: The binary string data represented by the parameter of 0 or 1.

- **Return format:**

  The query returns binary string data.

- **For example:**

  :TRIGger:CAN:DATA "1010100"

  Set the CAN trigger data value as 54h.

  :TRIGger:CAN:DATA ?

  The query returns 1010100.

:TRIGger:CAN:BYTecount

- **Command format:**

  :TRIGger:CAN:BYTecount <value>

  :TRIGger:CAN:BYTecount ?

- **Function description:**

  Set or query the CAN trigger condition as data or ID and the number of data bytes when the data is triggered。

- **Parameter:**

  value:INTEGER range: 1-8

- **Return format:**

  Query the number of data bytes when the CAN trigger condition is data or ID and data。

- **For example:**

  :TRIGger:CAN:BYTecount 2 When setting the CAN trigger condition to data or ID and data, the number of data bytes is 2

  :TRIGger:CAN:BYTecount ? Query return2

:TRIGger:CAN:DIRection

- **Command format:**

  :TRIGger:CAN:DIRection <direction>

  :TRIGger:CAN:DIRection ?

- **Function description:**

  Set or query the frame direction when the CAN trigger condition is ID or ID and data.

- **Parameter:**

  direction: Discrete type {READ | WRITe | BOTH}

  WRITe:Write

  READ:Read

  BOTH:Write&Read

- **Return format:**

  The query returns the frame direction when the CAN trigger condition is ID or ID and data.

- **For example:**

  :TRIGger:CAN:DIRection WRITe

  Set the CAN frame direction to WRITe.

:TRIGger:CAN:DIRection ?

The query returns WRITe.

:TRIGger:CAN:IDSTandard

- **Command format:**

  :TRIGger:CAN:IDSTandard <standard>

  :TRIGger:CAN:IDSTandard ?

- **Function description:**

  Set or query the ID standard when the CAN trigger condition is ID or ID and data.

- **Parameter:**

  standard: Discrete type {STANdard | EXTended}

  STANdard:Standard

  EXTended:Extended

- **Return format:**

  The query returns the ID standard when the CAN trigger condition is ID or ID and data.

- **For example:**

  :TRIGger:CAN:IDSTandard EXTended

  Set the CAN trigger ID standard as EXTended.

  :TRIGger:CAN:IDSTandard ?

  The query returns Extended.

:TRIGger:CAN:STANdardid

- **Command format:**

  :TRIGger:CAN:STANdardid <value>

  :TRIGger:CAN:STANdardid ?

- **Function description:**

  Set or query the standard ID when the CAN trigger condition is ID or ID and data.

- **Parameter:**

  value: The binary string data represented by the parameter of 0 or 1.

- **Return format:**

  The query returns binary string data.

- **For example:**

  :TRIGger:CAN:STANdardid "101000"

Set the CAN trigger standard ID to 028h.

:TRIGger:CAN:STANdardid ?

The query returns 101000.

:TRIGger:CAN:EXTendedid

- **Command format:**

  :TRIGger:CAN:EXTendedid <value>

  :TRIGger:CAN:EXTendedid ?

- **Function description:**

  Set or query the extended ID when the CAN trigger condition is ID or ID and data.

- **Parameter:**

  Value: Binary string data represented by the parameter 0 or 1.

- **Return format:**

  Query and return binary string data.

- **For example:**

:TRIGger:CAN:EXTendedid "101000"	Set the CAN trigger extended ID to 00028h.
:TRIGger:CAN:EXTendedid ?	Query and return 101000

:TRIGger:CAN:FTYPe

- **Command format:**

  :TRIGger:CAN:FTYPe <type>

  :TRIGger:CAN:FTYPe ?

- **Function description:**

  Set or query the frame type when the CAN trigger condition is the frame type.

- **Parameter:**

  type: Discrete type {DATA | REMote | ERRor | OVERload}

  DATA: Data frame

  REMote: Remote frame

  ERRor: Error frame

  OVERload: Overload frame

- **Return format:**

  The query returns the frame type when the CAN trigger condition is the frame type.

- **For example:**

:TRIGger:CAN:FTYPe DATA

Set the CAN trigger frame type to DATA.

:TRIGger:CAN:FTYPe ?

The query returns DATA.

:TRIGger:CAN:ETYPe

- **Command format:**

  :TRIGger:CAN:ETYPe <type>

  :TRIGger:CAN:ETYPe ?

- **Function description:**

  Set or query the error packet type when the CAN trigger condition is an error.

- **Parameter:**

  type: Discrete type {ANYerror | ACKLose | BITFillerror | CRCerror}

  ANYerrorAny error

  ACKLose:Ack lose

  BITFillerror:BIT fill error

  CRCerror:CRC error

- **Return format:**

  The query returns the error packet type when the CAN trigger condition is an error.

- **For example:**

  :TRIGger:CAN:ETYPe ACKLose

  Set the CAN trigger error type to ACKLose.

  :TRIGger:CAN:ETYPe ?

  The query returns ACKLose.

CAN-FD

:TRIGger:CAN_FD:CONDition

- **Command format:**

  :TRIGger:CAN_FD:CONDition <condition>

  :TRIGger:CAN_FD:CONDition ?

- **Function description:**

  Set or query the CAN_FD trigger mode.

- **Parameter:**

  condition: Discrete type {STARtframe | TYPeframe | ID | DATA | IDDAta | ENDFrame | ERRor}

  STARtframe: Frame start

  TYPeframe: Frame type

  ID: ID

  DATA: Data

  IDDAta: ID and data

  ENDFrame: Frame end

  ERRor: Loss of acknowledgment

  BITFILLERRor: Bit filling error

  CRCERRor: CRC error

  ALLERRor: All errors

- **Return format:**

  The query returns the CAN_FD trigger mode.

- **For example:**

  :TRIGger:CAN_FD:CONDition IDDAta

  Set the CAN_FD trigger mode as IDDAta.

  :TRIGger:CAN_FD:CONDition ?

  The query returns IDDAta.

:TRIGger:CAN_FD:DATA

- **Command format:**

  :TRIGger:CAN_FD:DATA <value>

  :TRIGger:CAN_FD:DATA ?

- **Function description:**

  Set or query the data value when the CAN_FD trigger condition is data or ID and data.

- **Parameter:**

  value: The binary string data represented by the parameter of 0 or 1.

- **Return format:**

  The query returns binary string data.

- **For example:**

  :TRIGger:CAN_FD:DATA "1010100"

  Set the CAN_FD data value to 54h.

:TRIGger:CAN_FD:DATA ?

The query returns 1010100.

:TRIGger:CAN_FD:DIRection

- **Command format:**

  :TRIGger:CAN_FD:DIRection <direction>

  :TRIGger:CAN_FD:DIRection ?

- **Function description:**

  Set or query the frame direction when the CAN trigger condition is ID or ID and data.

- **Parameter:**

  direction: Discrete type { WRITe | READ | BOTH}

  WRITe:Write

  READ:Read

  BOTH:Write&Read

- **Return format:**

  The query returns the frame direction when the CAN_FD trigger condition is ID or ID and data.

- **For example:**

  :TRIGger:CAN_FD:DIRection WRITe

  Set the CAN_FD trigger frame direction as WRITe.

  :TRIGger:CAN_FD:DIRection ?

  The query returns WRITe.

:TRIGger:CAN_FD:IDSTandard

- **Command format:**

  :TRIGger:CAN_FD:IDSTandard <standard>

  :TRIGger:CAN_FD:IDSTandard ?

- **Function description:**

  Set or query the ID standard when the CAN_FD trigger condition is ID or ID and data.

- **Parameter:**

  standard: Discrete type {STANdard | EXTended}

  STANdard:Standard

  EXTended:Extended

- **Return format:**

  The query returns the ID standard when the CAN_FD trigger condition is ID or ID and data.

- **For example:**

  :TRIGger:CAN_FD:IDSTandard EXTended

  Set the CAN_FD trigger ID standard to EXTended.

  :TRIGger:CAN_FD:IDSTandard ?

  The query returns EXTended.

:TRIGger:CAN_FD:STANdardid

- **Command format:**

  :TRIGger:CAN_FD:STANdardid <value>

  :TRIGger:CAN_FD:STANdardid ?

- **Function description:**

  Set or query the standard ID when the CAN_FD trigger condition is ID or ID and data.

- **Parameter:**

  value: The binary string data represented by the parameter of 0 or 1.

- **Return format:**

  The query returns binary string data.

- **For example:**

  :TRIGger:CAN_FD:STANdardid "101000"

  Set the CAN_FD trigger standard ID as 0x028.

  :TRIGger:CAN_FD:STANdardid ?

  The query returns 101000.

:TRIGger:CAN_FD:EXTendedid

- **Command format:**

  :TRIGger:CAN_FD:EXTendedid <value>

  :TRIGger:CAN_FD:EXTendedid ?

- **Function description:**

  Set or query the extended ID when the CAN_FD trigger condition is ID or ID and data.

- **Parameter:**

  Value: Binary string data represented by the parameter 0 or 1。

- **Return format:**

Query and return binary string data。

- **For example:**

  :TRIGger:CAN_FD:EXTendedid "1111" Set the CAN_FD trigger extended ID to 0x0000000F.

  :TRIGger:CAN_FD:EXTendedid ? Query and return 1111

:TRIGger:CAN_FD:FDSTandardid

- **Command format:**

  :TRIGger:CAN_FD:FDSTandardid <value>

  :TRIGger:CAN_FD:FDSTandardid?

- **Function description:**

  Set or query the FD standard ID when the CAN_FD trigger condition is ID or ID and data.

- **Parameter:**

  Value: Binary string data represented by the parameter 0 or 1.

- **Return format:**

  Query and return binary string data。

- **For example:**

  :TRIGger:CAN_FD:FDSTandardid "10001" Set the FD standard ID for CAN_FD triggering to 0x011.

  :TRIGger:CAN_FD:FDSTandardid ? The query returns 10001

:TRIGger:CAN_FD:FDEXtendedid

- **Command format:**

  :TRIGger:CAN_FD:FDEXtendedid <value>

  :TRIGger:CAN_FD:FDEXtendedid ?

- **Function description:**

  Set or query the FD extended ID when the CAN_FD trigger condition is ID or ID and data.

- **Parameter:**

  Value: Binary string data represented by the parameter 0 or 1

- **Return format:**

  Query and return binary string data

- **For example:**

  :TRIGger:CAN_FD:FDEXtendedid "10001" Set the FD extended ID for CAN_FD

triggering to 0x011.

:TRIGger:CAN_FD:FDEXtendedid ? The query returns 10001

:TRIGger:CAN_FD:FTYPe

- **Command format:**

  :TRIGger:CAN_FD:FTYPe <type>

  :TRIGger:CAN_FD:FTYPe ?

- **Function description:**

  Set or query the frame type when the CAN_FD trigger condition is the frame type.

- **Parameter:**

  type: Discrete type {DATA | REMote | ERRor | OVERload}

  DATA: Data frame

  REMote: Remote frame

  ERRor: Error frame

  OVERload: Overload frame

- **Return format:**

  The query returns the frame type when the CAN_FD trigger condition is the frame type.

- **For example:**

  :TRIGger:CAN_FD:FTYPe DATA

  Set the CAN_FD trigger frame type to DATA.

  :TRIGger:CAN_FD:FTYPe ?

  The query returns DATA.

:TRIGger:CAN_FD:ETYPe

- **Command format:**

  :TRIGger:CAN_FD:ETYPe <type>

  :TRIGger:CAN_FD:ETYPe ?

- **Function description:**

  Set or query the error packet type when the CAN_FD trigger condition is lost confirmation.

- **Parameter:**

  type: Discrete type {ANYerror | ACKLose | BITFillerror | CRCerror}

  ANYerrorAny error

  ACKLose:Ack lose

BITFillerror:BIT fill error

CRCerror:CRC error

- **Return format:**

  The query returns the error packet type when the CAN_FD trigger condition is lost confirmation.

- **For example:**

  :TRIGger:CAN_FD:ETYPe ACKLose

  Set the CAN_FD trigger error packet type as ACKLose.

  :TRIGger:CAN_FD:ETYPe ?

  The query returns ACKLose.

:TRIGger:CAN_FD:DLENgth

- **Command format:**

  :TRIGger:CAN_FD:DLENgth <value>

  :TRIGger:CAN_FD:DLENgth ?

- **Function description:**

  Set or query the number of data bytes when the CAN_FD trigger condition is data or ID and data.

- **Parameter:**

  value:Number of data bytes, with a range of 1 to 8

- **Return format:**

  Query and return the number of data bytes when the CAN_FD trigger condition is data or ID and data.

- **For example:**

  :TRIGger:CAN_FD:DLENgth1 Set the number of data bytes to 1 when the CAN_FD trigger mode is set to data.

  :TRIGger:CAN_FD:DLENgth ? The query returns 1.

LIN

:TRIGger:LIN:CONDition

- **Command format:**

:TRIGger:LIN:CONDition <condition>

:TRIGger:LIN:CONDition ?

- **Function description:**

  Set or query the trigger mode for LIN.

- **Parameter:**

  condition: Discrete type {STARtframe | ID | DATA | IDDAta | WAKeup | SLEep | SYERror | IDCRcerror | SUMCrcerror}

  STARtframe: Frame start

  ID: ID

  DATA: Data

  IDDAta: ID and data

  WAKeup: Wake-up frame

  SLEep: Sleep frame

  SYERror: Synchronization error

  IDCRcerror: ID checksum error

  SUMCrcerror: Checksum error

- **Return format:**

  The query returns the LIN trigger mode.

- **For example:**

  :TRIGger:LIN:CONDition IDDAta

  Set the LIN trigger mode to IDDAta.

  :TRIGger:LIN:CONDition ?

  The query returns IDDAta.

:TRIGger:LIN:DATA

- **Command format:**

  :TRIGger:LIN:DATA <value>

  :TRIGger:LIN:DATA ?

- **Function description:**

  Set or query the data value when the LIN trigger condition is data or ID and data.

- **Parameter:**

  value: The binary string data represented by the parameter of 0 or 1.

- **Return format:**

The query returns binary string data.

- **For example:**

  :TRIGger:LIN:DATA "1010100"

  Set the LIN data value as 54h.

  :TRIGger:LIN:DATA ?

  The query returns 1010100.

:TRIGger:LIN:ID

- **Command format:**

  :TRIGger:LIN:ID <value>

  :TRIGger:LIN:ID ?

- **Function description:**

  Set or query the standard ID when the LIN trigger condition is ID or ID and data.

- **Parameter:**

  value: The binary string data represented by the parameter of 0 or 1.

- **Return format:**

  The query returns binary string data.

- **For example:**

  :TRIGger:LIN:ID "101000"

  Set the LIN standard ID to 0x28.

  :TRIGger:LIN:ID ?

  The query returns 101000.

FLEXRAY

:TRIGger:FR:CONDition

- **Command format:**

  :TRIGger:FR:CONDition <condition>

  :TRIGger:FR:CONDition ?

- **Function description:**

  Set or query the trigger mode for FLEXRAY.

- **Parameter:**

  condition:Discrete type{HEADframe | INDicator | ID | CIRCulate | HEADer | DATA | IDDAta |

TAILframe | ERRor}

HEADframe:frame header

INDicator:Indicator bit

ID:ID

CIRCulate:Number of cycles

HEADer:Header fields

DATA:data

IDDAta:ID and Data

TAILframe:End of frame

ERRor:error

- **Return format:**

  The query returns the FLEXRAY trigger mode.

- **For example:**

  :TRIGger:FR:CONDition DATA

  Set the FLEXRAY trigger mode of TRIGger to DATA.

  :TRIGger:FR:CONDition ?

  The query returns DATA.

:TRIGger:FR:DATA

- **Command format:**

  :TRIGger:FR:DATA <value>

  :TRIGger:FR:DATA ?

- **Function description:**

  Set or query the data value when the FLEXRAY trigger condition is data or ID and data.

- **Parameter:**

  value: The binary string data represented by the parameter of 0 or 1.

- **Return format:**

  The query returns binary string data.

- **For example:**

  :TRIGger:FR:DATA "1010100"

  Set the FLEXRAY trigger data value to 54h.

  :TRIGger:FR:DATA ?

  The query returns 1010100.

:TRIGger:FR:DCOUnt

- **Command format:**

  :TRIGger:FR:DCOUnt <value>

  :TRIGger:FR:DCOUnt ?

- **Function description:**

  Set or query the data length when the FLEXRAY trigger condition is data or ID and data.

- **Parameter:**

  value: Integer type

- **Return format:**

  The query returns the data length when the FLEXRAY trigger condition is data or ID and data.

- **For example:**

  :TRIGger:FR:DCOUnt 3

  Set the FLEXRAY trigger data length as 3.

  :TRIGger:FR:DCOUnt ?

  The query returns 3.

:TRIGger:FR:ID

- **Command format:**

  :TRIGger:FR:ID <value>

  :TRIGger:FR:ID ?

- **Function description:**

  Set or query the standard ID when the FLEXRAY trigger condition is ID or ID and data.

- **Parameter:**

  value: The binary string data represented by the parameter of 0 or 1.

- **Return format:**

  The query returns binary string data.

- **For example:**

  :TRIGger:FR:ID "101000"

  Set the FLEXRAY trigger data value as 28h.

  :TRIGger:FR:ID ?

  The query returns 101000.

:TRIGger:FR:INDicator

- **Command format:**

  :TRIGger:FR:INDicator <type>

  :TRIGger:FR:INDicator ?

- **Function description:**

  Set or query the indicator bit type when the FLEXRAY trigger condition is the indicator bit.

- **Parameter:**

  type: Discrete type {NMFRame | PLFRame | EPFRame | SNFRame | STFRame}

  NMFRame: Normal frame

  PLFRame: Payload frame

  EPFRame: Empty frame

  SNFRame: Synchronization frame

  STFRame: Startup frame

- **Return format:**

  The query returns the indicator bit type when the FLEXRAY trigger condition is the indicator bit.

- **For example:**

  :TRIGger:FR:INDicator NMFRame

  Set the FLEXRAY trigger indicator type to NMFRame.

  :TRIGger:FR:INDicator ?

  The query returns NMFRame.

:TRIGger:FR:CYCLedata

- **Command format:**

  :TRIGger:FR:CYCLedata <value>

  :TRIGger:FR:CYCLedata ?

- **Function description:**

  Set or query the loop data when the FLEXRAY trigger condition is the number of cycles.

- **Parameter:**

  value: The binary string data represented by the parameter of 0 or 1.

- **Return format:**

The query returns binary string data.

- **For example:**

  :TRIGger:FR:CYCLedata "100000"

  Set the FLEXRAY trigger data value to 0x20.

  :TRIGger:FR:CYCLedata ?

  The query returns 100000.

:TRIGger:FR:EDFType

- **Command format:**

  :TRIGger:FR:EDFType <type>

  :TRIGger:FR:EDFType ?

- **Function description:**

  Set or query the end frame type when the FLEXRAY trigger condition is the end frame.

- **Parameter:**

  type: Discrete type {STATic | DYNamic | ALL}

  STATic:Static

  DYNamic:Dynamic

  ALL:All

- **Return format:**

  The query returns the end frame type when the FLEXRAY trigger condition is the end frame.

- **For example:**

  :TRIGger:FR:EDFType DYNamic

  Set the FLEXRAY trigger end frame type as DYNamic.

  :TRIGger:FR:EDFType ?

  The query returns DYNamic.

:TRIGger:FR:ETYPe

- **Command format:**

  :TRIGger:FR:ETYPe <type>

  :TRIGger:FR:ETYPe ?

- **Function description:**

  Set or query the error packet type when the FLEXRAY trigger condition is an error.

- **Parameter:**

  type: Discrete type {PGHeadcrc | PGTail | SEFRame | DEFRame | SYFRame | STFRame}

  PGHeadcrc: Header CRC

  PGTail: Frame tail

  SEFRame: Empty frame, static

  DEFRame: Empty frame, dynamic

  SYFRame: Synchronization frame

  STFRame: Startup frame (without synchronization)

- **Return format:**

  The query returns the error packet type when the FLEXRAY trigger condition is an error.

- **For example:**

  :TRIGger:FR:ETYPe SEFRame

  Set the FLEXRAY trigger error packet type to SEFRame.

  :TRIGger:FR:ETYPe ?

  The query returns SEFRame.

:TRIGger:FR:BIAS

- **Command format:**

  :TRIGger:FR:BIAS <type>

  :TRIGger:FR:BIAS ?

- **Function description:**

  Set and query to return the bias switch state of the data for the FR trigger condition.

- **Parameter:**

  type:Boolean type{ON | OFF}

- **Return format:**

  Query and return "1" and "0", representing "ON" and "OFF" respectively.

- **For example:**

  :TRIGger:FR:BIAS ON Set FR trigger condition to enable data bias

  :TRIGger:FR:BIAS ? Query return 1

:TRIGger:FR:OFFSet

- **Command format:**

  :TRIGger:FR:OFFSet <value>

:TRIGger:FR:OFFSet ?

- **Function description:**

  Set or query FR trigger condition as the bias value of the data。

- **Parameter:**

  value:real range：0-253。

- **Return format:**

  The FR trigger condition for querying is the bias value of the data。

- **For example:**

  :TRIGger:FR:OFFSet 1　　Set the FR trigger condition to a bias value of 1 for the data

  :TRIGger:FR:OFFSet ?　　Query returns 1

:TRIGger:FR:INDicator:DATA

- **Command format:**

  :TRIGger:FR:INDicator:DATA <value>

  :TRIGger:FR:INDicator:DATA ?

- **Function description:**

  Set or query FR trigger condition as the indicator bit value of the header field。

- **Parameter:**

  value:Binary string data represented by Parameter 0 or 1。

- **Return format:**

  The query returns the FR trigger condition as the indicator bit value of the header field。

- **For example:**

  :TRIGger:FR:INDicator:DATA 100　　Set the FR trigger condition to have the indicator bit value of the header field set to 04h

  :TRIGger:FR:INDicator:DATA ?　　Query return 100

:TRIGger:FR:PAYLoad

- **Command format:**

  :TRIGger:FR:PAYLoad <value>

  :TRIGger:FR:PAYLoad ?

- **Function description:**

  Set or query FR trigger condition as static load length of header field。

- **Parameter:**

  value:Binary string data represented by Parameter 0 or 1。

- **Return format:**

  Query the static load length with FR trigger condition as the header field。

- **For example:**

  :TRIGger:FR:PAYLoad 100 Set the FR trigger condition to have the indicator bit value of the header field set to 04h

  :TRIGger:FR:PAYLoad ? Query return 100

:TRIGger:FR:HEAD

- **Command format:**

  :TRIGger:FR:HEAD <value>

  :TRIGger:FR:HEAD ?

- **Function description:**

  Set or query FR trigger condition as header CRC value of header field。

- **Parameter:**

  value:Binary string data represented by Parameter 0 or 1。

- **Return format:**

  The query returns the CRC value of the header field triggered by the FR condition。

- **For example:**

  :TRIGger:FR:HEAD 100 Set the FR trigger condition to have the indicator bit value of the header field set to 04h

  :TRIGger:FR:HEAD ? Query return100

AUDIOBUS

:TRIGger:AB:CONDition

- **Command format:**

  :TRIGger:AB:CONDition <type>

  :TRIGger:AB:CONDition ?

- **Function description:**

  Set or query the AUDIOBUS trigger method.

- **Parameter:**

type: Discrete type {SYNC | DATA}

SYNC:synchronization bit

DATA:Data

- **Return format:**

  The query returns the AUDIOBUS trigger method.

- **For example:**

  :TRIGger:AB:CONDition DATA

  Set the AUDIOBUS trigger method to DATA.

  :TRIGger:AB:CONDition ?

  The query returns DATA.

:TRIGger:AB:DATA

- **Command format:**

  :TRIGger:AB:DATA <value>

  :TRIGger:AB:DATA ?

- **Function description:**

  Set or query the data value when the AUDIOBUS trigger condition is data.

- **Parameter:**

  value: The binary string data represented by the parameter of 0 or 1.

- **Return format:**

  The query returns binary string data.

- **For example:**

  :TRIGger:AB:DATA "1010100"

  Set the AUDIOBUS data value to 0x00000054.

  :TRIGger:AB:DATA ?

  The query returns 1010100.

:TRIGger:AB:Type

- **Command format:**

  :TRIGger:AB:TYPe <TYPe>

  :TRIGger:AB:TYPe ?

- **Function description:**

  Set or query the channel type when AB trigger condition is data。

- **Parameter:**

  type:Discrete type {LR | LEFT | RIGHT}

  LR:Left/Right

  LEFT:Left

  RIGHT:Right

- **Return format:**

  Query the channel type when the AB trigger condition is data。

- **For example:**

  :TRIGger:AB:TYPe LEFT When setting AB trigger condition to data, the channel type is left channel

  :TRIGger:AB:TYPe ? Query returnLEFT

MIL_STD_1553

:TRIGger:MS:CONDition

- **Command format:**

  :TRIGger:MS:CONDition <condition>

  :TRIGger:MS:CONDition ?

- **Function description:**

  Set or query MIL-STD_1553 trigger conditions。

- **Parameter:**

  Condition Discrete type{CMD | ERRor | DATA | SYNC}

  CMD:Command/Status Word

  ERRor:error

  DATA:data

  SNYC:synchronous

- **Return format:**

  Query returns MIL-STD_1553 trigger condition。

- **For example:**

  :TRIGger:MS:CONDition DATA Set MIL-STD_1553 trigger condition to DATA

  :TRIGger:MS:CONDition ? Query returnDATA

:TRIGger:MS:ADDRess

- **Command format:**

  :TRIGger:MS:ADDRess <value>

  :TRIGger:MS:ADDRess ?

- **Function description:**

  Set or query the remote terminal address when the MS trigger condition is a command/status word。

- **Parameter:**

  Value:INTEGER Binary string data represented by Parameter 0 or 1。

- **Return format:**

  Query the remote terminal address when the MS trigger condition is a command/status word。

- **For example:**

  :TRIGger:MS:ADDRess 100 When setting the MS trigger condition to command/status word, the remote terminal address is set to 4

  :TRIGger:MS:ADDRess ? Query return100

:TRIGger:MS:DATA

- **Command format:**

  :TRIGger:MS:DATA <value>

  :TRIGger:MS:DATA ?

- **Function description:**

  When setting or querying MS trigger conditions as command/status words or data values。

- **Parameter:**

  Value:INTEGER Binary string data represented by Parameter 0 or 1。

- **Return format:**

  Query the data value when the MS trigger condition is a command/status word or data。

- **For example:**

  :TRIGger:MS:DATA 100 When setting the MS trigger condition to command/status word or data, the data value is set to 4

  :TRIGger:MS:DATA ? Query return100

:TRIGger:MS:PAIRty

- **Command format:**

  :TRIGger:MS:PAIRty <value>

  :TRIGger:MS:PAIRty ?

- **Function description:**

  When setting or querying MS trigger conditions as command/status words or data, check the value。

- **Parameter:**

  Value:INTEGER Binary string data represented by Parameter 0 or 1。

- **Return format:**

  Verify the value when the query returns MS trigger conditions as command/status words or data。

- **For example:**

  :TRIGger:MS:PAIRty 100 When setting the trigger condition for MS to command/status word or data, the check value is set to 4

  :TRIGger:MS:PAIRty ? Query return100

:TRIGger:MS:ERRor

- **Command format:**

  :TRIGger:MS:ERRor <type>

  :TRIGger:MS:ERRor ?

- **Function description:**

  Error type when setting or querying MS trigger condition as error。

- **Parameter:**

  type: Discrete type{ODDevent | SYNC | MANChester | NOTContinue}

  ODDevent:Parity check

  SYNC:synchronous

  MANChester:Manchester

  NOTContinue:Non continuous data

- **Return format:**

  Error type when the query returns MS trigger condition as error。

- **For example:**

:TRIGger:MS:ERRor SYNC	When setting the MS trigger condition to error, the error type is synchronous
:TRIGger:MS:ERRor ?	Query returnSYNC

ARINC429

:TRIGger:A429:CONDition

- **Command format:**

  :TRIGger:A429:CONDition <condition>

  :TRIGger:A429:CONDition ?

- **Function description:**

  Set or query the trigger method for ARINC429.

- **Parameter:**

  condition:Discrete type {HDFRame | FrameTail | LABel | SDI | DATA | SSM | LabelAndData | ERror}

  HDFRame:Frame Start

  FrameTail:End of frame

  LABel:label

  SDI:sdi

  DATA:data

  SSM:Signs and status bits

  LabelAndData:Tags and data

  ERror:error

- **Return format:**

  The query returns the ARINC429 triggering method.

- **For example:**

  :TRIGger:A429:CONDition DATA

  Set the ARINC429 trigger method to DATA.

  :TRIGger:A429:CONDition ?

  The query returns DATA.

:TRIGger:A429:DATA

- **Command format:**

:TRIGger:A429:DATA <value>

:TRIGger:A429:DATA ?

- **Function description:**

  Set or query the data value when the ARINC429 trigger condition is data.

- **Parameter:**

  value: The binary string data represented by the parameter of 0 or 1.

- **Return format:**

  The query returns binary string data.

- **For example:**

  :TRIGger:A429:DATA "1010100"

  Set the triggering data value of ARINC429 as 0x54.

  :TRIGger:A429:DATA ?

  The query returns 1010100.

:TRIGger:A429:SDI

- **Command format:**

  :TRIGger:A429:SDI <value>

  :TRIGger:A429:SDI ?

- **Function description:**

  Set or query the value when the ARINC429 trigger condition is the source or destination identifier.

- **Parameter:**

  value: The binary string data represented by the parameter of 0 or 1.

- **Return format:**

  The query returns binary string data.

- **For example:**

  :TRIGger:A429:SDI "100011"

  Set the ARINC429 trigger SDI value as 0x23.

  :TRIGger:A429:SDI ?

  The query returns 100011.

:TRIGger:A429:SSM

- **Command format:**

:TRIGger:A429:SSM <value>

:TRIGger:A429:SSM ?

- **Function description:**

  Set or query the value when the ARINC429 trigger condition is the sign and status bit.

- **Parameter:**

  value: The binary string data represented by the parameter of 0 or 1.

- **Return format:**

  The query returns binary string data.

- **For example:**

  :TRIGger:A429:SSM "100011"

  Set the ARINC429 trigger SSM value to 0x23.

  :TRIGger:A429:SSM ?

  The query returns 100011.

:TRIGger:A429:LABel

- **Command format:**

  :TRIGger:A429:LABel <value>

  :TRIGger:A429:LABel ?

- **Function description:**

  Set or query the label value when the ARINC429 trigger condition is the label.

- **Parameter:**

  value:Binary string data represented by Parameter 0 or 1

- **Return format:**

  The query returns the label value when the ARINC429 trigger condition is the label.

- **For example:**

  :TRIGger:A429:LABel "100011"

  Set the ARINC429 trigger label value as 0x03.

  :TRIGger:A429:LABel ?

  The query returns 100011

:TRIGger:A429:ERRor

- **Command format:**

  :TRIGger:A429:ERRor <value>

:TRIGger:A429:ERRor ?

- **Function description:**

  Error type when setting or querying A429 trigger condition as error。

- **Parameter:**

  type：Discrete type{CRC | WORD | GAP | ANY}

  CRC:Inspection position error

  WORD:Word error

  GAP:Frame gap error

  ANY:Any error

- **Return format:**

  Error type when query returns A429 trigger condition as error。

- **For example:**

  :TRIGger:A429:ERRor WORD When setting A429 trigger condition to error, the error type is word error

  :TRIGger:A429:ERRor ? Query return WORD

SENT

:TRIGger:SENT:CONDition

- **Command format:**

  :TRIGger:SENT:CONDition <condition>

  :TRIGger:SENT:CONDition ?

- **Function description:**

  Set or query SENT trigger conditions。

- **Parameter:**

  condition:Discrete type{SYNC | STATus | DATA | CRC | STD | STDC | ERRor}

  SYNC:synchronous

  STATus:state

  DATA:data

  CRC:Verification error

  STD:Status+Data

  STDC:Status+Data+CRC

  ERRor:error

- **Return format:**

  Query returns SENT trigger condition。

- **For example:**

  :TRIGger:SENT:CONDition DATA Set SENT trigger condition to DATA

  :TRIGger:SENT:CONDition ? Query return DATA

:TRIGger:SENT:DATA

- **Command format:**

  :TRIGger:SENT:DATA <value>

  :TRIGger:SENT:DATA ?

- **Function description:**

  Set or query the data value when the SENT trigger condition is data。

- **Parameter:**

  value:Binary string data represented by Parameter 0 or 1。

- **Return format:**

  Query returns binary string data。

- **For example:**

  :TRIGger:SENT:DATA 100 Set SENT trigger data value to 0x4h

  :TRIGger:SENT:DATA ? Query return 100

:TRIGger:SENT:STATus

- **Command format:**

  :TRIGger:SENT:STATus <value>

  :TRIGger:SENT:STATus ?

- **Function description:**

  Set or query the state value when the SENT trigger condition is set to state。

- **Parameter:**

  value:Binary string data represented by Parameter 0 or 1。

- **Return format:**

  Query the state value when the SENT trigger condition is the state。

- **For example:**

  :TRIGger:SENT:STATus 011 Set SENT trigger status to 0x3h

  :TRIGger:SENT:STATus ? Query return 011

:TRIGger:SENT:DLEN

- **Command format:**

  :TRIGger:SENT:DLEN <value>

  :TRIGger:SENT:DLEN ?

- **Function description:**

  Set or query the data length when the SENT trigger condition is data。

- **Parameter:**

  value:INTEGER range：1-6。

- **Return format:**

  Query the data length when the SENT trigger condition is data。

- **For example:**

  :TRIGger:SENT:DLEN 1　　Set SENT trigger data length to 1

  :TRIGger:SENT:DLEN ?　　Query return 1

:TRIGger:SENT:CRC

- **Command format:**

  :TRIGger:SENT:CRC <value>

  :TRIGger:SENT:CRC ?

- **Function description:**

  Set or query the CRC value when the SENT trigger condition is CRC。

- **Parameter:**

  value:Binary string data represented by Parameter 0 or 1。

- **Return format:**

  Query the CRC value when the SENT trigger condition is CRC。

- **For example:**

  :TRIGger:SENT:CRC 011　　Set SENT trigger data length to 0x3h

  :TRIGger:SENT:CRC ?　　Query return 011

:TRIGger:SENT:ERR

- **Command format:**

  :TRIGger:SENT:ERR <type>

  :TRIGger:SENT:ERR ?

- **Function description:**

  Set or query the error option value when the SENT trigger condition is set to error。

- **Parameter:**

  condition:Discrete type{CRC | CONPuls}

  CRC:Fast CRC error

  CONPuls:Continuous pulse error

- **Return format:**

  Query the error option value when the SENT trigger condition is incorrect。

- **For example:**

  :TRIGger:SENT:ERR CRC When setting the SENT trigger condition to error, the error option value is set to fast CRC error

  :TRIGger:SENT:ERR ? Query returnCRC

Waveform Data

:WAVeform:SOURce

- **Command format:**

  :WAVeform:SOURce <source>

  :WAVeform:SOURce ?

- **Function description:**

  Set or query the channel source for waveform data reading.

- **Parameter:**

  source: Discrete type {C1 | C2 | C3 | C4 | M1 | M2 | M3 | M4 | M5 | M6 | M7 | M8}

- **Return format:**

  The query returns "C1", "C2", "C3", "C4", "M1", "M2", "M3", "M4", "M5", "M6", "M7", "M8".

- **For example:**

  :WAVeform:SOURce C1

  Set the channel source for waveform data reading as Channel 1.

  :WAVeform:SOURce ?

  Query and return C1.

:WAVeform:MODE

- **Command format:**

:WAVeform:MODE <mode>

:WAVeform:MODE ?

- **Function description:**

  Set or query the waveform data type.

- **Parameter:**

  mode: Discrete type {NORMal | RAW}

  NORMal: Screen display data

  RAW: Memory waveform data

- **Return format:**

  Query and return "NORMal", "RAW".

- **For example:**

  :WAVeform:MODE NORMal

  Set the waveform data type to NORMal.

  :WAVeform:MODE ?

  Query and return NORMal.

:WAVeform:FORMat

- **Command format:**

  :WAVeform:FORMat <format>

  :WAVeform:FORMat ?

- **Function description:**

  Set or query the Return format of the waveform data.

- **Parameter:**

  format:{BYTE | ASCII | WORD}

  BYTE: Single-byte data. If the AD value of a single waveform point occupies one byte and is only valid when there is an 8-bit ADC. Only "C1", "C2", "C3", and "C4" are supported. Return the waveform data of the analog channel.

  WORD: Double-byte data. If the AD value of a single waveform point occupies two bytes, when it is an 8-bit ADC, the lower 8 bits are valid and the upper 8 bits are 0.

  ASCII: Return the actual voltage values of each waveform point in scientific notation. The voltage values are separated by commas and conform to the data block format. When the channel source is a math channel, only ASCII format data is valid. Return the waveform

voltage data of the math channel displayed on the screen.

- **Return format:**

  Query returns "BYTE", “WORD”, "ASCII".

- **For example:**

  :WAVeform:FORMat BYTE

  Set the Return format of the waveform data as BYTE.

  :WAVeform:FORMat ?

  Query returns BYTE.

:WAVeform:STARt

- **Command format:**

  :WAVeform:STARt <value>

  :WAVeform:STARt ?

- **Function description:**
- Set or query the starting position for waveform reading.
- **Parameter:**

  value: Integer type

- **Return format:**
- Query and return the starting position of the waveform to be read.
- **For example:**

  :WAVeform:STARt 0

  Set the waveform reading start position to 0 (the first point).

  :WAVeform:STARt ?

  Query and return 0.

:WAVeform:POINts

- **Command format:**

  :WAVeform:POINts <point>

  :WAVeform:POINts ?

- **Function description:**

  Set or query the number of waveform points to be read.

- **Parameter:**

  point: Integer type.

- **Return format:**

  The query returns the number of waveform points to be read.

- **For example:**

  :WAVeform:POINts 10000

  Set the number of waveform points to be read as 10000.

  :WAVeform:POINts ?

  The query returns 10000.

:WAVeform:DATA

- **Command format:**

  :WAVeform:DATA ?

- **Function description:**

  Read the waveform data currently displayed on the screen; .

- **Return format:**

  The query returns the waveform data. The return format depends on the selected waveform data format.

- **For example:**

  The instruction sequence for obtaining the waveform data of the specified data source is executed as follows:

  The process of obtaining the screen waveform data.

  :WAVeform:SOURce C1

  Set the signal source for the waveform data to be queried currently as C1.

  :WAVeform: MODE NORMal

  Set to read the waveform data displayed on the screen.

  :WAVeform:FORMat WORD

  The return format of the waveform data is double-word mode.

  :WAVeform:STARt 1000

  The waveform data is read starting from the 1000th point.

  :WAVeform:POINts 10000

  Set the number of waveform points to be read as 10,000.

  :WAVeform:DATA ?

  Obtain the waveform data.

  The process of obtaining the memory waveform data. When the storage depth is too large,

it needs to be read in blocks. This process is only valid when the oscilloscope is in the stopped state.

:WAVeform:SOURce C1

Set the signal source of the waveform data to be queried currently as Channel 1.

:WAVeform: MODE RAW

Set the reading of memory waveform data.

:WAVeform:FORMat WORD

The return format of the waveform data is double-word mode.

:WAVeform:STARt 1000

Set the starting point for reading the memory waveform to the 1000th point

:WAVeform:POINts 5000

The number of waveform points read from the memory is 5000.

Send the read waveform command in a loop to obtain the entire memory data.

:WAVeform:DATA ?

Obtain the waveform data of a block in the memory.

:WAVeform: STARt ?

Until the starting position of the waveform data reading is equal to -1, it indicates that it has reached the last point.

Measurement

Measurement parameter table

Type	parameter	description	Multiple sources
Vertical	maximum value（MAX）	The maximum sampling point value, referring to the highest value in the waveform	
Vertical	minimum value（MIN）	The minimum sampling point value, referring to the lowest value in the	

		waveform	
Vertical	Peak-to-peak value (PKPK)	The difference between the maximum value and the minimum value	
Vertical	top value（TOP）	The value with the highest frequency above the midpoint of the waveform	
Vertical	base value（BASE ）	The value with the highest frequency below the midpoint of the waveform	
Vertical	Middle value（MID）	The average value between the top value and the bottom value	
Vertical	amplitude（AMPL）	The difference between the top value and the bottom value	
Vertical	average value（AVE）	The arithmetic mean of all sampling points in the waveform area	
Vertical	RMS value（RMS）	The root mean square (RMS) of all waveform data in the waveform area	
Vertical	Standard deviation (STDD)	The root mean square value of the voltage values of all the waveform data after removing the DC component in the waveform area	
Vertical	Positive overshoot (POST)	The distortion after the rising edge transition	

Vertical	Negative overshoot (NOST)	The distortion after the falling edge transition	
Vertical	cycle maximum value (CMAX)	The maximum value of the waveform within one cycle	
Vertical	cycle minimum value (CMIN)	The minimum value of the waveform within one cycle	
Vertical	cycle RMS value (CRMS)	The root mean square value of the waveform within one cycle	
Vertical	cycle average value (CAVE)	The average value of the waveform within one cycle	
Vertical	cycle Peak-to-peak value (CPK)	The peak-to-peak value of the waveform within one cycle	
Vertical	cycle Middle value (CMID)	The median value of the waveform within one cycle	
Vertical	positive pre-shoot (PPST)	The positive pre-shoot of the waveform refers to the distortion before the rising edge transition	
Vertical	negative pre-shoot (NPST)	The negative pre-shoot of the waveform refers to the distortion before the falling edge transition	
Horizontal	period（PRD）	The period is the time required to complete	

		one repetitive cycle	
Horizontal	frequency（FREQ）	The frequency is defined as the reciprocal of the period	
Horizontal	rise time（RISE）	The time required for the edge to rise from the lowest reference level (RB) to the highest reference level (RT)	
Horizontal	fall time（FALL）	The time required for the edge to fall from the highest reference level (RT) to the lowest reference level (RB)	
Horizontal	positive pulse width（PWID）	The time between the middle threshold of the rising edge and the middle threshold of the next falling edge	
Horizontal	negative pulse width（NWID）	The time from the middle threshold of the falling edge to the middle threshold of the next rising edge	
Horizontal	positive duty cycle（DUTY）	The ratio of the waveform's positive pulse width to the period	
Horizontal	negative duty cycle（NDTY）	The ratio of the negative pulse width of the waveform to the period	
Horizontal	time@Max（TMAX）	The horizontal axis value when the maximum value of the waveform first	

		occurs starting from the left side of the waveform area	
Horizontal	time@Min（TMIN）	The horizontal axis value when the minimum value of the waveform first occurs starting from the left side of the waveform area	
Horizontal	rise time@Lv（LRIS）	The rise time of the waveform at a specified level	
Horizontal	fall time@Lv（LFAL）	The fall time of the waveform at the specified level	
Horizontal	period@Lv（LPRD）	The period time of the waveform at the specified level	
Horizontal	frequency@Lv（LFRQ）	The waveform frequency at a specified level	
Horizontal	pulse width@Lv（LWID）	The pulse width of the waveform at the specified level	
Horizontal	duty cycle@Lv（LDUT）	The duty cycle of the waveform at the specified level	
Horizontal	phase difference@Lv（LPHA）	The phase difference at the 50% point of the first rising edge of the two sources at the specified level	√

Horizontal	RRD@Lv（LRRD）	Calculate the time difference at a specified level between the first rising edges of the two sources	√
Horizontal	FFD@Lv（LFFD）	Calculate the time difference at the specified level of the first falling edge of the two sources	√
Horizontal	RFD@Lv（LRFD）	Calculate the time difference at the specified level from the rising edge of the first source to the falling edge of the second source	√
Horizontal	FRD@Lv（LFRD）	Calculate the time difference at the specified level from the falling edge of the first source to the rising edge of the second source	√
Horizontal	skew（SKEW）	Calculate the time difference between the 50% points of the rising edges of the first and second sources.	√
Horizontal	waveform length（WLEN）	The number of samples of the waveform sampling points involved in the measurement	
Horizontal	number of periods	The number of cycles of the periodic	

	(CYCL)	waveform within the waveform area	
Horizontal	setup time (STPT)	The time from exceeding the specified middle reference level on the data source to the most recent time of exceeding the specified middle reference level on the clock source	√
Horizontal	hold time (HLDT)	The time from exceeding the specified middle reference level on the clock source to the most recent time of exceeding the specified middle reference level on the data source	√
Other	area (AREA)	The area of the waveform refers to the area between the measured waveform and the ground level	
Other	cycle area (CAR)	The area of the waveform within one cycle	

Quick Meas

:MEASure:QKMEasure:DISPlay

- **Command format:**

  :MEASure:QKMEasure:DISPlay <active>

  :MEASure:QKMEasure:DISPlay?

- **Function description:**

Set or query the status of the Quick Meas switch.

- **Parameter:**

  active: Boolean type {ON | OFF} or {1 | 0}

- **Return format:**

  The query returns that "1" and "0" represent "ON" and "OFF" respectively.

- **For example:**

  :MEASure:QKMEasure:DISPlay ON Turn Quick Meas on.

  :MEASure:QKMEasure:DISPlay ? The query returns 1.

:MEASure:QKMEasure:SOURce

- **Command format:**

  :MEASure:QKMEasure:SOURce <source>

  :MEASure:QKMEasure:SOURce ?

- **Function description:**

  Set or query the Quick Meas source.

- **Parameter:**

  source: Discrete type {C1 | C2 | C3 | C4}

- **Return format:**

  The query returns "C1", "C2", "C3", "C4".

- **For example:**

  :MEASure:QKMEasure:SOURce C1

  Set the quick meas source as Channel 1.

  :MEASure:QKMEasure:SOURce ?

  The query returns C1.

:MEASure:QKMEasure:DATA

- **Command format:**

  :MEASure:QKMEasure:DATA <source>

  :MEASure:QKMEasure:DATA ?

- **Function description:**

  Query all measurement results in the parameter snapshot.

  Among them, the parameter snapshot contains the following measurement parameters and is arranged as follows: Please refer to the measurement parameter table

"MAX", "MIN", "PKPK", "TOP", "BASE", "MID", "AMPL", "AVE", "RMS", "STDD", "POST", "NOST", "AREA", "PRD", "FREQ", "RISE", "FALL", "PWID", "NWID", "DUTY", "NDTY", "TMAX", "TMIN", "CMAX", "CMIN", "CRMS", "CAVE", "CPK", "CAR", "CMID", "WLEN", "CYCL", "PPST", "NPST"

- **Return format:**

  Query and return all results of the parameter snapshot measurement of the corresponding channel, in line with the data block format, expressed in scientific notation, and arranged in sequence separated by commas. Invalid values are represented by the maximum value of the real data type.

  For example: "#90000001001.20000E+000,2.00000E+002,1.20000E+003…."

Parameter measurement

:MEASure:ALL:ACTive

- **Command format:**

  :MEASure:ALL:ACTive <active>

- **Function description:**

  Set whether all measurement items are activated.

- **Parameter:**

  active: {ON | OFF} or {1 | 0}

- **For example:**

  :MEASure:ALL:ACTive ON

  Set all measurement items to be activated.

:MEASure:ULTRaacq

- **Command format:**

  :MEASure:ULTRaacq <mode>

- **Function description:**

  Set multiple measurement items of vertical type or horizontal type to be quickly enabled.

- **Parameter:**

  mode: Discrete {VERTical | HORizontal}

  VERTical: Vertical

  HORizontal: Horizontal

- **For example:**

:MEASure:ULTRaacq VERTical

Set multiple measurement items of the vertical type to be quickly enabled.

:MEASure:GATe

- **Command format:**

  :MEASure:GATe <type>

  :MEASure:GATe ?

- **Function description:**

  Set or query the threshold for measurements.

- **Parameter:**

  type: Discrete {SCReen | CURSor}

  SCReen: Screen

  CURSor: Cursor

- **Return format:**

  The query returns "SCReen", "CURSor".

- **For example:**

  :MEASure:GATe SCReen Set the measurement threshold to the screen.

  :MEASure:GATe ? The query returns SCReen.

:MEASure:INDicator

- **Command format:**

  :MEASure:INDicator <indicator>

  :MEASure:INDicator ?

- **Function description:**

  Set or query the indicator.

- **Parameter:**

  indicator: Discrete type {OFF | P1 | P2 | P3 | P4 | P5 | P6 | P7 | P8 | P9 | P10}

- **Return format:**

  The query returns "OFF", "P1", "P2", "P3", "P4", "P5", "P6", "P7", "P8", "P9", "P10".

- **For example:**

  :MEASure:INDicator P1 Set the indicator of P1 to be on.

  :MEASure:INDicator ? The query returns P1.

:MEASure:ITEM<n>:DISPlay

- **Command format:**

  :MEASure:ITEM<n>:DISPlay <active>

  :MEASure:ITEM<n>:DISPlay ?

- **Function description:**

  Set or query the activation status of a single measurement item.

- **Parameter:**

  active: {ON | OFF} or {1 | 0}

- **Return format:**

  The query returns that "1" and "0" represent "ON" and "OFF" respectively.

- **For example:**

  :MEASure:ITEM1:DISPlay ON

  Turn on measurement item P1.

  :MEASure:ITEM1:DISPlay ?

  The query returns 1.

:MEASure:STATistics:DISPlay

- **Command format:**

  :MEASure:STATistics:DISPlay <active>

  :MEASure:STATistics:DISPlay ?

- **Function description:**

  Set or query whether the statistics are enabled.

- **Parameter:**

  active: {ON | OFF} or {1 | 0}

- **Return format:**

  The query returns "1" and "0" representing "ON" and "OFF" respectively.

- **For example:**

  :MEASure:STATistics:DISPlay ON Turn on statistics.

  :MEASure:STATistics:DISPlay ? The query returns 1.

:MEASure:STATistics:RESet

- **Command format:**

:MEASure:STATistics:RESet

- **Function description:**

  Reset the statistics

:MEASure:ITEM<n>:TYPe

- **Command format:**

  :MEASure:ITEM<n>:TYPe <type>

  :MEASure:ITEM<n>:TYPe ?

- **Function description:**

  Set or query the measurement item type.

- **Parameter:**

  n: Discrete {1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 }. Indicates the nth measurement Parameter.

  type: Discrete type, represent the type of measurement parameter, Please refer to the measurement parameter table

  {MAX | MIN | PKPK | TOP | BASE | MID | AMPL | AVE | RMS | STDD | POST | NOST | PPST | NPST | AREA | PRD | FREQ | RISE | FALL | PWID | NWID | DUTY| NDTY | CYCL | TMAX | TMIN | CMAX | CMIN | CRMS | CAVE | CPK | CMID | CAR | LRIS | LFAL | LPRD | LFRQ | LWID | LDUT | LPHA | LRRD | LFFD | LRFD | LFRD | SKEW | WLEN | STPT | HLDT }and parameter calculation{+ | - | * | /}

- **Return format:**

  The query returns "MAX", "MIN", "PKPK", "TOP", "BASE", "MID", "AMPL", "AVE", "RMS", "STDD", "POST", "NOST", "PPST", "NPST", "AREA", "PRD", "FREQ", "RISE", "FALL", "PWID", "NWID", "DUTY", "NDTY", "CYCL", "TMAX", "TMIN", "CMAX", "CMIN", "CRMS", "CAVE", "CPK", "CMID", "CAR", "LRIS", "LFAL", "LPRD", "LFRQ", "LWID", "LDUT", "LPHA", "LRRD", "LFFD", "LRFD", "LFRD", "SKEW", "WLEN", "STPT", "HLDT"and parameter calculation.

- **For example:**

  :MEASure:ITEM3:TYPe "P2+P1"

  Set the measurement item of measurement position P3 as the"P2+P1"

  :MEASure:ITEM3:TYPe ?

  The query returns P2+P1

:MEASure:ITEM<n>:SOURce<ch>

- **Command format:**

:MEASure:ITEM<n>:SOURce<ch> <source>

:MEASure:ITEM<n>:SOURce<ch> ?

- **Function description:**

  Set or query the source for the measurement position.

- **Parameter:**

  n: Discrete {1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10}. It represents the nth measurement Parameter.

  source: Discrete {C1 | C2 | C3 | C4 | R1 | R2 | R3 | R4}

  ch: Discrete {1 | 2}. It represents source 1 or source 2.

  If the measurement parameter is a single source, only source 1 needs to be set, and source 2 can be omitted.

  If the measurement parameter is of multiple sources, both source 1 and source 2 need to be set, and source 2 cannot be omitted.

- **Return format:**

  The query returns "C1", "C2", "C3", "C4", "R1", "R2", "R3", "R4".

- **For example:**

  :MEASure:ITEM1:SOURce1 C1

  Set source 1 of measurement position P1 to channel 1.

  :MEASure:ITEM1:SOURce1 ?

  The query returns C1.

:MEASure:ITEM<n>:THRold:TYPe

- **Command format:**

  :MEASure:ITEM<n>:THRold:TYPe <type>

  :MEASure:ITEM<n>:THRold:TYPe ?

- **Function description:**

  Set or query the reference threshold of the measurement position - top-finding mode, and it can be set successfully only under a specific measurement item type.

- **Parameter:**

  n: Discrete {1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 }. Indicates the nth measurement Parameter.

  type: Top-finding mode {BASetop | MINMax | ZERMin | ZERMax}

  BASetop: Base value - Top value

  MINMax: Minimum - Maximum value

  ZERMin: Zero - Minimum value

ZERMax: Zero - Maximum value

- **Return format:**

  The query returns "BASetop", "MINMax", "ZERMin", "ZERMax".

- **For example:**

  :MEASure:ITEM1:THRold:TYPe BASetop

  Set the reference threshold - top-finding mode of the measurement position P1 to the top and bottom values.

  :MEASure:ITEM1:THRold:TYPe ?

  The query returns BASetop.

:MEASure:ITEM<n>:THRold:TOP

- **Command format:**

  :MEASure:ITEM<n>:THRold:TOP <top>

  :MEASure:ITEM<n>:THRold:TOP ?

- **Function description:**

  Set or query the high reference threshold for the measurement item. The valid value is 25% - 95%. It can be set successfully only under a specific measurement item type.

- **Parameter:**

  n: Discrete {1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 }, Indicates the nth measurement Parameter.

  top: Integer, high reference value

- **Return format:**

  The query returns the threshold.

- **For example:**

  :MEASure:ITEM1:THRold:TOP 50

  Set the high reference threshold of measurement item P1 to 50%.

  :MEASure:ITEM1:THRold:TOP ?

  The query returns 50.

:MEASure:ITEM<n>:THRold:MIDDle

- **Command format:**

  :MEASure:ITEM<n>:THRold:MIDDle <middle>

  :MEASure:ITEM<n>:THRold:MIDDle ?

- **Function description:**

Set or query the high reference value of the measurement item threshold. The valid value is 15% - 85%. It can be set successfully only under a specific measurement item type.

- **Parameter:**

  n: Discrete {1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 }, Indicates the nth measurement Parameter.

  middle: Integer, middle reference value

- **Return format:**

  The query returns the threshold.

- **For example:**

  :MEASure:ITEM1:THRold:MIDDle 50

  Set the threshold in the reference threshold of the measurement bit P1 - to 50%.

  :MEASure:ITEM1:THRold:MIDDle ?

  The query returns 50.

:MEASure:ITEM<n>:THRold:BOTTom

- **Command format:**

  :MEASure:ITEM<n>:THRold:BOTTom <bottom>

  :MEASure:ITEM<n>:THRold:BOTTom ?

- **Function description:**

  Set or query the low reference value of the measurement item threshold. The valid value is 5% - 75%. It can be set successfully only under the specific measurement item type.

- **Parameter:**

  n: Discrete {1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 }, Indicates the nth measurement Parameter.

  bottom: Integer, low reference value

- **Return format:**

  The query returns the threshold.

- **For example:**

  :MEASure:ITEM1:THRold:BOTTom 50

  Set the reference threshold - the bottom threshold of the measurement bit P1 to 50%.

  :MEASure:ITEM1:THRold:BOTTom ?

  The query returns 50.

:MEASure:ITEM<n>:VALue

- **Command format:**

:MEASure:ITEM<n>:VALue ?

- **Function description:**

  Query the measured values for the measurement items.

- **Parameter:**

  n: Discrete type {1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10}, Indicates the nth measurement Parameter.

- **Return format:**

  The query returns the measured values.

:MEASure:ITEM<n>:COUNter:MAX

- **Command format:**

  :MEASure:ITEM<n>:COUNter:MAX ?

- **Function description:**

  Query the maximum statistical value of the measurement item.

- **Parameter:**

  n: Discrete type {1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10}, Indicates the nth measurement Parameter.

- **Return format:**

  The query returns the maximum value of the statistical values of the measurement items.

:MEASure:ITEM<n>:COUNter:MIN

- **Command format:**

  :MEASure:ITEM<n>:COUNter:MIN ?

- **Function description:**

  Query the minimum statistical value of the measurement item.

- **Parameter:**

  n: Discrete {1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 }, Indicates the nth measurement Parameter.

- **Return format:**

  The query returns the minimum value of the statistical values of the measurement items.

:MEASure:ITEM<n>:COUNter:AVG

- **Command format:**

  :MEASure:ITEM<n>:COUNter:AVG ?

- **Function description:**

Query the statistical value of the measurement item - the average value.

- **Parameter:**

  n: Discrete {1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 }, It represents the nth measurement Parameter.

- **Return format:**

  The query returns the average value of the statistical values of the measurement items.

:MEASure:ITEM<n>:COUNter:DEV

- **Command format:**

  :MEASure:ITEM<n>:COUNter:DEV ?

- **Function description:**

  Query the standard deviation of the measurement item.

- **Parameter:**

  n: Discrete {1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 }, It indicates the nth measurement Parameter.

- **Return format:**

  The query returns the standard deviation of the statistical values of the measurement items.

:MEASure:ITEM<n>:COUNter:POP

- **Command format:**

  :MEASure:ITEM<n>:COUNter:POP?

- **Function description:**

  Query the statistics of the measurement items - count values.

- **Parameter:**

  n: Discrete {1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 }, It represents the nth measurement Parameter.

- **Return format:**

  The query returns the count values of the statistics of the measurement items.

:MEASure:ITEM<n>:HISTgram

- **Command format:**

  :MEASure:ITEM<n>:HISTgram <active>

  :MEASure:ITEM<n>:HISTgram ?

- **Function description:**

Set or query whether the histogram is enabled.

- **Parameter:**

  active: {ON | OFF} or {1 | 0}

- **Return format:**

  The query returns "1" and "0", representing "ON" and "OFF", respectively.

- **For example:**

  :MEASure:ITEM1:HISTgram ON Enable the histogram.

  :MEASure:ITEM1:HISTgram ? The query returns 1.

:MEASure:ITEM<n>:TRENd

- **Command format:**

  :MEASure:ITEM<n>:TRENd <active>

  :MEASure:ITEM<n>:TRENd ?

- **Function description:**

  Set or query whether the trend chart is enabled.

- **Parameter:**

  active: {ON | OFF} or {1 | 0}

- **Return format:**

  The query returns "1" and "0" representing "ON" and "OFF" respectively.

- **For example:**

  :MEASure:ITEM1:TRENd ON Enable the trend chart.

  :MEASure:ITEM1:TRENd ? The query returns 1.

:MEASure:ITEM<n>:TRACk

- **Command format:**

  :MEASure:ITEM<n>:TRACk <active>

  :MEASure:ITEM<n>:TRACk ?

- **Function description:**

  Set or query whether the tracking is enabled.

- **Parameter:**

  active: {ON | OFF} or {1 | 0}

- **Return format:**

  The query returns "1" and "0" representing "ON" and "OFF" respectively.

- **For example:**

  :MEASure:ITEM1:TRACk ON Turn on the tracking.

  :MEASure:ITEM1:TRACk ? The query returns 1.

Math

:MATH<n>:ACTive

- **Command format:**

  :MATH<n>:ACTive <active>

  :MATH<n>:ACTive ?

- **Function description:**

  Set or query the enabled status of the specified mathematical channel.

- **Parameter:**

  active: Boolean {ON | OFF} or {1 | 0}

- **Return format:**

  The query returns that "1" and "0" represent "ON" and "OFF" respectively.

- **For example:**

  :MATH1:ACTive ON Set mathematical channel 1 to be open.

  :MATH1:ACTive ? The query returns 1.

:MATH<n>:LABel:NAMe

- **Command format:**

  :MATH<n>:LABel:NAMe <label>

  :MATH<n>:LABel:NAMe ?

- **Function description:**

  Set or query the label for the mathematical channel.

- **Parameter:**

  Label: Channel name string

- **Return format:**

  The query returns the channel name string.

- **For example:**

  :MATH1:LABel:NAMe “abcd”

  Set the label for mathematical channel 1 to abcd.

:MATH1:LABel:NAMe ?

The query returns abcd.

:MATH<n>:UNIT:ENABle

- **Command format:**

  :MATH<n>:UNIT:ENABle <enable>

  :MATH<n>:UNIT:ENABle ?

- **Function description:**

  Set or query the enabled status of the custom unit of the mathematical channel.

- **Parameter:**

  enable: Boolean type {ON | OFF} or {1 | 0}

- **Return format:**

  The query returns that "1" and "0" represent "ON" and "OFF" respectively.

- **For example:**

  :MATH1:UNIT:ENABle ON

  Set the custom unit of mathematical channel 1 to be on.

  :MATH1:UNIT:ENABle ?

  The query returns 1.

:MATH<n>:UNIT:NAMe

- **Command format:**

  :MATH<n>:UNIT:NAMe <unit>

  :MATH<n>:UNIT:NAMe ?

- **Function description:**

  Set or query the display unit for the specified mathematical channel. It can be set successfully only when the custom unit is enabled.

- **Parameter:**

  unit: Unit string

- **Return format:**

  Query and return the display unit name of the specified mathematical channel.

- **For example:**

  :MATH1:UNIT:NAMe “A”

  Set the display unit for mathematical channel 1 to A.

:MATH1:UNIT:NAMe ?

Query and return A.

:MATH<n>:TYPe

- **Command format:**

  :MATH<n>:TYPe <type>

  :MATH<n>:TYPe ?

- **Function description:**

  Set or query the type of mathematical operation of the mathematical channel.

- **Parameter:**

  Type: Discrete {BINary | FFT | FILTer | CUSTom | ADVanced}

  BINary: Basic operation

  FFT: Fast Fourier Transform

  FILTer: Filter

  CUSTom: Custom

  ADVanced: Advanced operation

- **Return format:**

  Query and return "BINary", "FFT", "ERES", "CUStom", "ADVanced".

- **For example:**

  :MATH1:TYPe FFT

  Set the mathematical operation type of mathematical channel 1 to FFT.

  :MATH1:TYPe ?

  Query and return FFT.

:MATH<n>:VERTical:OFFSet

- **Command format:**

  :MATH<n>:VERTical:OFFSet <offset>

  :MATH<n>:VERTical:OFFSet ?

- **Function description:**

  Set or query the vertical position for the mathematical channel.

- **Parameter:**

  offset: Real number type, in units of div, ranging from -4div to 4div.

- **Return format:**

Query and return the vertical position of the mathematical channel.

- **For example:**

  :MATH1:VERTical:OFFSet 1

  Set the vertical position for mathematical channel 1 to 1div.

  :MATH1:VERTical:OFFSet ?

  Query and return 1.00000E+000.

:MATH<n>:VERTical:SCALe

- **Command format:**

  :MATH<n>:VERTical:SCALe <scale>

  :MATH<n>:VERTical:SCALe ?

- **Function description:**

  Set or query the vertical scale of the mathematical channel.

- **Parameter:**

  scale: Real number type, unit: Custom unit. If no unit is entered, the default unit is V.

- **Return format:**

  Query and return the vertical scale of the mathematical channel.

- **For example:**

  :MATH1:VERTical:SCALe 1

  Set the vertical scale of mathematical channel 1 to 1V.

  :MATH1:VERTical:SCALe ?

  Query and return 1.00000E+000.

:MATH<n>:HORizontal:OFFSet

- **Command format:**

  :MATH<n>:HORizontal:OFFSet <offset>

  :MATH<n>:HORizontal:OFFSet ?

- **Function description:**

  Set or query the horizontal position of the math channel.offset: Real number type, in units of div, and the range is 0div - 10div.

- **Parameter:**

  offset: Real number type, in units of div, and the range is 0div - 10div.

- **Return format:**

The query returns the horizontal displacement of the math channel.

- **For example:**

  :MATH1:HORizontal:OFFSet 1

  Set the horizontal position of math channel 1 to 1div.

  :MATH1:HORizontal:OFFSet ?

  The query returns 1.00000E+000.

:MATH<n>:HORizontal:SCALe

- **Command format:**

  :MATH<n>:HORizontal:SCALe <scale>

  :MATH<n>:HORizontal:SCALe ?

- **Function description:**

  Set or query the horizontal scale of the math channel.

- **Parameter:**

  scale: Real number type, unit: The default unit is s, range: Related to the size of the main time base.

- **Return format:**

  The query returns the horizontal scale of the math channel.

- **For example:**

  :MATH1:HORizontal:SCALe 1us

  Set the horizontal scale of math channel1 to 1us.

  :MATH1:HORizontal:SCALe ?

  The query returns 1.0000000000000E-006.

:MATH<n>:SOURce<ch>

- **Command format:**

  :MATH<n>:SOURce<ch> <source>

  :MATH<n>:SOURce<ch> <source> ?

- **Function description:**

  Set or query the operation source of the mathematical channel.

- **Parameter:**

  ch: {1 | 2}, indicating source 1 or source 2

  source: Discrete type {C1 | C2 | C3 | C4 | R1 | R2 | R3 | R4}

- **Return format:**

  The query returns "C1", "C2", "C3", "C4", "R1", "R2", "R3", "R4".

- **For example:**

  :MATH1:SOURce1 C1

  Set the operation source of mathematical channel 1 as channel 1.

  :MATH1:SOURce1 ?

  The query returns C1.

Note: If there is only one source, it can be written as source1, or directly as source.

Basic operation

:MATH<n>:OPERation

- **Command format:**

  :MATH<n>:OPERation <type>

  :MATH<n>:OPERation ?

- **Function description:**

  Set or query the operation type for the mathematical channel.

- **Parameter:**

  type: Discrete{ADD | SUBTract | MULTiply | DIVide}

  ADD:Add

  SUBTract:Subtract

  MULTiply:Multiply

  DIVide:Divide

- **Return format:**

  The query returns "ADD", "SUBTract", "MULTiply", "DIVide".

- **For example:**

  :MATH1:OPERation ADD

  Set the operation type for mathematical channel 1 to addition.

  :MATH1:OPERation ?

  The query returns ADD.

FFT

:MATH<n>:FFT:WINDow

- **Command format:**

  :MATH<n>:FFT:WINDow <type>

  :MATH<n>:FFT:WINDow ?

- **Function description:**

  When the operation type is Fast Fourier Transform (FFT), set or query the window function for the mathematical channel.

- **Parameter:**

  Type: Discrete {RECTangle | HANN | HAMMing | BLACkman | FLATtop}

  RECTangle: Rectangular window

  HANN: Hanning window

  HAMMing: Hamming window

  BLACkman: Blackman window

  FLATtop: Flat-top window

- **Return format:**

  The query returns "RECTangle", "HANN", "HAMMing", "BLACkman", "FLATtop".

- **For example:**

  :MATH1:FFT:WINDow RECTangle

  Set the window function for mathematical channel 1 to rectangular.

  :MATH1:FFT:WINDow ?

  The query returns RECTangle.

:MATH<n>:FFT:TYPe

- **Command format:**

  :MATH<n>:FFT:TYPe <type>

  :MATH<n>:FFT:TYPe ?

- **Function description:**

  When the operation type is Fast Fourier Transform (FFT), set or query the output type of the mathematical channel.

- **Parameter:**

type: Discrete type {AMPLtd | POWer | PSD | REAL | IMAGinary | PHASe}

AMPLtd : Amplitude spectrum

POWer: Power spectrum

PSD: Psd

REAL: Real part

IMAGinary: Imaginary part

PHASe: Phase spectrum

- **Return format:**

  The query returns "AMPLtd", "POWer", "PSD", "REAL", "IMAGinary", "PHASe".

- **For example:**

  :MATH1:FFT:TYPe AMPLtd

  Set the output type of mathematical channel 1 as amplitude spectrum.

  :MATH1:FFT:TYPe ?

  The query returns AMPLtd.

:MATH<n>:FFT:UNIT

- **Command format:**

  :MATH<n>:FFT:UNIT <type>

  :MATH<n>:FFT:UNIT ?

- **Function description:**

  When the output type of the Fast Fourier Transform (FFT) is the amplitude spectrum, set or query the FFT unit for the mathematical channel.

- **Parameter:**

  type: Discrete type {VRMS | DBM | DBUW | DBMV | DBUV | DBMA | DBUA}

  VRMS:Vrms

  DBM:dBm

  DBUW:dBuW

  DBMV:dBmV

  DBUV:dBuV

  DBMA:dBmA

  DBUA:dBuA

- **Return format:**

  The query returns "VRMS", "DBM", "DBUW", "DBMV", "DBUV", "DBMA", "DBUA".

- **For example:**

  :MATH1:FFT:UNIT VRMS

  Set the FFT unit for mathematical channel 1 to VRMS.

  :MATH1:FFT:UNIT ?

  The query returns VRMS.

:MATH<n>:FFT:POINt

- **Command format:**

  :MATH<n>:FFT:POINt <type>

  :MATH<n>:FFT:POINt ?

- **Function description:**

  When the operation type is Fast Fourier Transform (FFT), set or query the scan points for the mathematical channel.

- **Parameter:**

  Type: Discrete {1k | 2k | 4k | 8k | 16k | 32k | 64k | 128k | 256k | 512k | 1M}, the default unit is Pts.

- **Return format:**

  The query returns "1k", "2k", "4k", "8k", "16k", "32k", "64k", "128k", "256k", "512k", "1M".

- **For example:**

  :MATH1:FFT:POINt 1k

  Set the scan points for mathematical channel 1 to 1KPts.

  :MATH1:FFT:POINt ?

  The query returns 1K.

:MATH<n>:FFT:FREQuency:CENTer

- **Command format:**

  :MATH<n>:FFT:FREQuency:CENTer <freq>

  :MATH<n>:FFT:FREQuency:CENTer ?

- **Function description:**

  When the operation type is Fast Fourier Transform (FFT), set or query the center frequency for the specified FFT. The default unit is Hz.

- **Parameter:**

  freq: Real number type, with units of mHz, Hz, kHz, MHz

- **Return format:**

  The query returns the center frequency, which is expressed in scientific notation.

- **For example:**

  :MATH1:FFT:FREQuency:CENTer 100

  Set the center frequency for math channel 1 to 100Hz.

  :MATH1:FFT:FREQuency:CENTer ?

  The query returns 1.0000000000000E+002.

:MATH<n>:FFT:FREQuency:SPAN

- **Command format:**

  :MATH<n>:FFT:FREQuency:SPAN

  :MATH<n>:FFT:FREQuency:SPAN ?

- **Function description:**
- When the operation type is Fast Fourier Transform (FFT), set or query the FFT frequency span. The default unit is Hz.
- **Parameter:**
- span: Real type, unit: Hz, kHz, MHz
- **Return format:**
- The query returns the frequency span, which is expressed in scientific notation.
- **For example:**

  :MATH1:FFT:FREQuency:SPAN 100000

  Set the FFT frequency span for math channel 1 to 100 kHz.

  :MATH1:FFT:FREQuency:SPAN ?

  The query returns 1.0000000000000E+005.

Note: The actual configurable range of the frequency span is related to the current time base scale.

User-define operation

:MATH<n>:EXPRession

- **Command format:**

  :MATH<n>:EXPRession <expression>

  :MATH<n>:EXPRession ?

- **Function description:**

  Perform mathematical calculations using free combination expressions.

  The expression format can be seen in the custom type formula editing window under the MATH menu of the oscilloscope. <expression> belongs to the ASCII string Parameter.

- **Parameter:**

  Expression: Custom operation string.

  Custom operations only support analog channels and reference channels.

- **For example:**

  :MATH1:EXPRession “C1 * C2”

  It indicates multiplying Channel 1 and Channel 2 of MATH1.

  :MATH1:EXPRession ?

  The query returns C1 * C2.

Note: All function expressions contained in the formula need to be in lowercase (channel letters are fixed in uppercase).

Filter

:MATH<n>:FILTer:TYPe

- **Command format:**

  :MATH<n>:FILTer:TYPe <type>

  :MATH<n>:FILTer:TYPe ?

- **Function description:**

  When the operation type is Filter, set or query the type of filter.

- **Parameter:**

  type: Discrete {LOWPass | HIGHpass | BNDPass | BNDStop}

  LOWPass: Low-pass

  HIGHpass: High-pass

  BNDPass: Band-pass

  BNDStop: Band-stop

- **Return format:**

  The query returns "LOWPass", "HIGHpass", "BNDPass", "BNDStop".

- **For example:**

  :MATH1:FILTer:TYPe BNDStop

Set the filter type for mathematical channel 1 to band-stop.

:MATH1:FILTer:TYPe ?

The query returns BNDStop.

:MATH<n>:FILTer:FREQuency<n>:STOP

- **Command format:**

  :MATH<n>:FILTer:FREQuency<n>:STOP <freq>

  :MATH<n>:FILTer:FREQuency<n>:STOP ?

- **Function description:**

  When the operation type is Filter, set or query the cutoff frequency, and the default unit is Hz.

- **Parameter:**

  n: Discrete {1 | 2}, representing the cutoff frequency 1 and the cutoff frequency 2

  freq: Real number type, units mHz, Hz, kHz, MHz

- **Return format:**

  The query returns the cutoff frequency and is expressed in scientific notation.

- **For example:**

  :MATH1:FILTer:FREQuency1:STOP 100

  Set the filter cutoff frequency 1 of math channel 1 to 100Hz.

  :MATH1:FILTer:FREQuency1:STOP ?

  The query returns 1.0000000000000E+002.

:MATH<n>:FILTer:CUSTom:ENABle

- **Command format:**

  :MATH<n>:FILTer:CUSTom:ENABle <enable>

  :MATH<n>:FILTer:CUSTom:ENABle ?

- **Function description:**

  When the operation type is Filter, set or query the status of the custom filter.

- **Parameter:**

  enable: Boolean type {ON | OFF} or {1 | 0}

- **Return format:**

  The query returns that "1" and "0" represent "ON" and "OFF" respectively.

- **For example:**

:MATH1:FILTer:CUSTom:ENABle ON

Enable the custom filter for mathematical channel 1.

:MATH1:FILTer:CUSTom:ENABle ?

The query returns 1.

Frequency counter

:CYMometer:ACTive

- **Command format:**

  :CYMometer:ACTive <active>

  :CYMometer:ACTive ?

- **Function description:**

  Set or query the status of the frequency counter.

- **Parameter:**

  The query returns "1" and "0", representing "ON" and "OFF" respectively.

- **Return format:**

  The query returns "1" and "0", representing "ON" and "OFF" respectively.

- **For example:**

  :CYMometer:ACTive ON Turn on the frequency counter.

  :CYMometer:ACTive ? The query returns 1.

:CYMometer:FREQuency

- **Command format:**

  :CYMometer:FREQuency ?

- **Function description:**

  The query returns the value of the frequency counter.

- **Return format:**

  The query returns the measured value of the frequency counter in Hz.

Digital Voltmeter

:DVM:ACTive

- **Command format:**

  :DVM:ACTive <active>

  :DVM:ACTive ?

- **Function description:**

  Set or query the on state of the voltmeter.

- **Parameter:**

  active: Boolean type {ON | OFF} or {1 | 0}

- **Return format:**

  The query returns "1" and "0", representing "ON" and "OFF", respectively.

- **For example:**

  :DVM:ACTive ON Set the voltmeter on.

  :DVM:ACTive ? The query returns 1.

:DVM:SOURce

- **Command format:**

  :DVM:SOURce <source>

  :DVM:SOURce ?

- **Function description:**

  Set or query the source for the voltmeter.

- **Parameter:**

  source:{C1 | C2 | C3 | C4}

- **Return format:**

  The query returns "C1", "C2", "C3", "C4".

- **For example:**

  :DVM:SOURce C1 Set the voltmeter source to C1.

  :DVM:SOURce ? The query returns C1.

:DVM:MODe

- **Command format:**

:DVM:MODe <mode>

:DVM:MODe ?

- **Function description:**

  Set or query the voltmeter mode.

- **Parameter:**

  mode:{DC | ACRMs | DCACrms}

- **Return format:**

  The query returns "DC", "ACRMs", "DCACrms".

- **For example:**

  :DVM:MODe DC Set the voltmeter mode to DC.

  :DVM:MODe ? The query returns DC.

:DVM:AUTO

- **Command format:**

  :DVM:AUTO <active>

  :DVM:AUTO ?

- **Function description:**

  Set or query the enabled status of the voltmeter's automatic range.

- **Parameter:**

  active: Boolean type {ON | OFF} or {1 | 0}

- **Return format:**

  The query returns that "1" and "0" represent "ON" and "OFF" respectively.

- **For example:**

  :DVM:AUTO ON Set the automatic range of the voltmeter to be on.

  :DVM:AUTO ? The query returns 1.

:DVM:CURRent

- **Command format:**

  :DVM:CURRent ?

- **Function description:**

  The query returns the value of the voltmeter.

- **Return format:**

  The query returns the measured value of the voltmeter in volts.

Display

:DISPlay:PERSist

- **Command format:**

  :DISPlay:PERSist <persist>

  :DISPlay:PERSist ?

- **Function description:**

  Set or query the infinite persistence mode.

- **Parameter:**

  persist: Discrete type {CLOSe | AUTO | INFinity}

  CLOSe: Close

  AUTO: Automatic

  INFinity: Infinite

- **Return format:**

  The query returns the infinite afterglow states "CLOSe", "AUTO", and "INFinity".

- **For example:**

  :DISPlay:PERSist CLOSe Turn off the infinite persistence.

  :DISPlay:PERSist ? The query returns CLOSe.

:DISPlay:MARK

- **Command format:**

  :DISPlay:MARK <mark>

  :DISPlay:MARK?

- **Function description:**

  Set or query the display mark.

- **Parameter:**

  mark: Boolean type {ON | OFF} or {1 | 0}

- **Return format:**

  The query returns that "1" and "0" represent "ON" and "OFF" respectively.

- **For example:**

  :DISPlay:MARK ON Set the display mark to be on.

  :DISPlay:MARK ? The query returns 1.

:DISPlay:MKVert

- **Command format:**

  :DISPlay:MKVert <mkvert>

  :DISPlay:MKVert ?

- **Function description:**

  Set or query the position of the vertical mark.

- **Parameter:**

  mkvert:{LEFT | RIGHt}

  LEFT:Left

  RIGHt:Right

- **Return format:**

  The query returns the vertical mark positions "LEFT" and "RIGHt".

- **For example:**

  :DISPlay:MKVert LEFT Set the vertical mark to the left position.

  :DISPlay:MKVert ? The query returns LEFT.

:DISPlay:MKHori

- **Command format:**

  :DISPlay:MKHori <mkHori>

  :DISPlay:MKHori ?

- **Function description:**

  Set or query the horizontal mark position.

- **Parameter:**

  mkHori:{TOP | BOTTom}

  TOP:Top

  BOTTom:Bottom

- **Return format:**

  The query returns the horizontal mark positions "TOP" and "BOTTom".

- **For example:**

  :DISPlay:MKHori TOP Set the horizontal mark position to the top.

  :DISPlay:MKHori ? The query returns TOP.

:DISPlay:GRID:STYLe

- **Command format:**

  :DISPlay:GRID:STYLe <style>

  :DISPlay:GRID:STYLe ?

- **Function description:**

  Set or query the style of the grid.

- **Parameter:**

  style: {FULL | BRIef | NONe}

  FULL;Full

  BRIef:Brief

  NONe:None

- **Return format:**

  The query returns the grid styles "FULL", "BRIef", and "NONe".

- **For example:**

  :DISPlay:GRID:STYLe FULL Set the grid style to FULL.

  :DISPlay:GRID:STYLe ? The query returns FULL.

:DISPlay:GRID:BRIGtness

- **Command format:**

  :DISPlay:GRID:BRIGtness <value>

  :DISPlay:GRID:BRIGtness ?

- **Function description:**

  Set or query the grid brightness.

- **Parameter:**

  Value: Integer type, range: 0 - 100.

- **Return format:**

  The query returns the grid brightness value.

- **For example:**

  :DISPlay:GRID:BRIGtness 50 Set the grid brightness value to 50.

  :DISPlay:GRID:BRIGtness ? The query returns 50.

:DISPlay:WAVe:STYLe

- **Command format:**

  :DISPlay:WAVe:STYLe <style>

  :DISPlay:WAVe:STYLe ?

- **Function description:**

  Set or query the waveform style.

- **Parameter:**

  style:{VECTor | DOT}

  VECTor: vector

  DOT: point

- **Return format:**

  The query returns the waveform styles "VECTor" and "DOT".

- **For example:**

  :DISPlay:WAVe:STYLe VECTor Set the waveform style as vector.

  :DISPlay:WAVe:STYLe ? The query returns VECTor.

:DISPlay:WAVe:BRIGtness

- **Command format:**

  :DISPlay:WAVe:BRIGtness <value>

  :DISPlay:WAVe:BRIGtness ?

- **Function description:**

  Set or query the brightness of the waveform.

- **Parameter:**

  value: Integer type, range: 0 - 100.

- **Return format:**

  The query returns the waveform brightness value.

- **For example:**

  :DISPlay:WAVe:BRIGtness 50 Set the waveform brightness to 50.

  :DISPlay:WAVe:BRIGtness ? The query returns 50.

:DISPlay:SCReen:BRIGtness

- **Command format:**

:DISPlay:SCReen:BRIGtness <value>

:DISPlay:SCReen:BRIGtness ?

- **Function description:**

  Set or query the brightness of the screen.

- **Parameter:**

  value: Integer type, range: 5 - 100.

- **Return format:**

  The query returns the screen brightness value.

- **For example:**

  :DISPlay:SCReen:BRIGtness 50 Set the screen brightness to 50.

  :DISPlay:SCReen:BRIGtness ? The query returns 50.

:DISPlay:SCReen:CONTrast

- **Command format:**

  :DISPlay:SCReen:CONTrast <value>

  :DISPlay:SCReen:CONTrast ?

- **Function description:**

  Set or query the screen contrast.

- **Parameter:**

  value: Integer type, range: 50 - 100.

- **Return format:**

  The query returns the screen contrast value.

- **For example:**

  :DISPlay:SCReen:CONTrast 50 Set the screen contrast value to 50.

  :DISPlay:SCReen:CONTrast ? The query returns 50.

Save

Save-Waveform

:FILe:WAVe:FORMat

- **Command format:**

  :FILe:WAVe:FORMat <format>

:FILe:WAVe:FORMat ?

- **Function description:**

  Set or query the format for saving waveforms.

- **Parameter:**

  format:{BINary | TEXT | MATlab | EXCel | CSV | TSV | DAT}

  BINary:Binary

  TEXT:Text

  MATlab:Matlab

  EXCel:Excel

  CSV:CSV

  TSV:TSV

  DAT:DAT

- **Return format:**

  The query returns the waveform save formats "BINary", "TEXT", "MATlab", "EXCel", "CSV", "TSV", "DAT".

- **For example:**

:FILe:WAVe:FORMat TEXT	Set the waveform save format to TEXT.
:FILe:WAVe:FORMat ?	The query returns TEXT

:FILe:WAVe:TXTEncoding

- **Command format:**

  :FILe:WAVe:TXTEncoding <encoding>

  :FILe:WAVe:TXTEncoding ?

- **Function description:**

  Set or query the text encoding format when the waveform save format is text.

- **Parameter:**

  encoding:{ASCII | GB2312 | UTF8 | UTF32 | UINCode}

  ASCII:ASCII

  GB2312:

  UTF8:UTF8

  UTF32:UTF32

  UINCode:Uincode

- **Return format:**

The query returns the text encoding formats "ASCII", "GB2312", "UTF8", "UTF32", "UINCode".

- **For example:**

:FILe:WAVe:TXTEncoding ASCII	Set the text encoding format to ASCII.
:FILe:WAVe:TXTEncoding ?	The query returns ASCII.

:FILe:WAVe:SOURce

- **Command format:**

  :FILe:WAVe:SOURce <source>

  :FILe:WAVe:SOURce ?

- **Function description:**

  Set or query the source of the saved waveform.

- **Parameter:**

  source:{C1 | C2 | C3 | C4}

- **Return format:**

  The query returns the saved waveform sources "C1", "C2", "C3", "C4".

- **For example:**

:FILe:WAVe:SOURce C1	Set the waveform saved source C1.
:FILe:WAVe:SOURce ?	The query returns C1.

:FILe:WAVe:DATetime:ENABle

- **Command format:**

  :FILe:WAVe:DATetime:ENABle <enable>

  :FILe:WAVe:DATetime:ENABle ?

- **Function description:**

  Set or query whether the name of the saved waveform appends the date.

- **Parameter:**

  enable: Boolean type {ON | OFF} or {1 | 0}

- **Return format:**

  The query returns that "1" and "0" represent "ON" and "OFF" respectively.

- **For example:**

  :FILe:WAVe:DATetime:ENABle OFF

  Set the saved waveform name to exclude the date.

  :FILe:WAVe:DATetime:ENABle ?

The query returns 0.

:FILe:WAVe:NAMe

- **Command format:**

  :FILe:WAVe:NAMe <name>

  :FILe:WAVe:NAMe ?

- **Function description:**

  Set or query the name of the saved waveform.

- **Parameter:**

  name: Customizable input for saving the waveform name. If no name is entered, the default name will automatically have a serial number appended.

- **Return format:**

  Query and return the waveform name.

- **For example:**

  :FILe:WAVe:NAMe abcd Set the saved waveform name to abcd.

  :FILe:WAVe:NAMe ? Query and return abcd.

:FILe:WAVe:PATH

- **Command format:**

  :FILe:WAVe:PATH <path>

  :FILe:WAVe:PATH ?

- **Function description:**

  Set or query the waveform save path.

- **Return format:**

  Query and return the waveform path.

- **For example:**

  :FILe:WAVe:PATH "D:\\WaveForm" Set the waveform save path as D:\WaveForm

  :FILe:WAVe:PATH ? The query returned D:\\WaveForm

:FILe:WAVe:SAVe

- **Command format:**

  :FILe:WAVe:SAVe ?

- **Function description:**

  Save the waveform data and return whether the saving was successful.

- **Return format:**

  Query and return the result as "SUCCess" or "FAIL".

Screenshot

:FILe:PICTure:REGion

- **Command format:**

  :FILe:PICTure:REGion <region>

  :FILe:PICTure:REGion ?

- **Function description:**

  Set or query the area for saving images.

- **Parameter:**

  region:{WINDow | APPLication}

  WINDow: grid

  APPLication: screen

- **Return format:**

  The query returned "WINDow" or "APPLication".

- **For example:**

  :FILe:PICTure:REGion WINDow Set the grid area for image saving.

  :FILe:PICTure:REGion ? The query returned WINDow.

:FILe:PICTure:COLor

- **Command format:**

  :FILe:PICTure:COLor <color>

  :FILe:PICTure:COLor ?

- **Function description:**

  Set or query the color for image saving.

- **Parameter:**

  color: {STANdard | BLACkwhite | REVerse}

  STANdard: Standard

  BLACkwhite: Black and white

REVerse: Reverse

- **Return format:**

  The query returned "STANdard ", "BLACkwhite", "REVerse".

- **For example:**

  :FILe:PICTure:COLor STANdard Set the image save color to standard.

  :FILe:PICTure:COLor ? The query returned STANdard.

:FILe:PICTure:FORMat

- **Command format:**

  :FILe:PICTure:FORMat <format>

  :FILe:PICTure:FORMat ?

- **Function description:**

  Set or query the image saving format.

- **Parameter:**

  format:{BMP | TIFF | GIF | PNG | JPEG}

  BMP:Bmp

  TIFF:Tiff

  GIF:Gif

  PNG:Png

  JPEG:Jpeg

- **Return format:**

  The query returned "BMP", "TIFF", "GIF", "PNG", "JPEG".

- **For example:**

  :FILe:PICTure:FORMat JPEG Set the image saving format as JPEG.

  :FILe:PICTure:FORMat ? The query returned JPEG.

:FILe:PICTure:NAMe

- **Command format:**

  :FILe:PICTure:NAMe <name>

  :FILe:PICTure:NAMe ?

- **Function description:**

  Set or query the image saving name.

- **Parameter:**

Name: Discrete. Customizable input for saving the waveform name. If no name is entered, the default name will automatically have a superimposed serial number.

- **For example:**

:FILe:PICTure:NAMe abcd Set the image saving name as abcd.

:FILe:PICTure:NAMe ? Query returns abcd.

:FILe:PICTure:DATetime:ENABle

- **Command format:**

:FILe:PICTure:DATetime:ENABle <enable>

:FILe:PICTure:DATetime:ENABle ?

- **Function description:**

Set or query whether to include the date in the image save name.

- **Parameter:**

enable: Boolean type {ON | OFF} or {1 | 0}

- **Return format:**

The query returns that "1" and "0" represent "ON" and "OFF" respectively.

- **For example:**

:FILe:PICTure:DATetime:ENABle OFF

Set the image save name to exclude the date.

:FILe:PICTure:DATetime:ENABle ?

The query returns 0.

:FILe:PICTure:PATH

- **Command format:**

:FILe:PICTure:PATH <path>

:FILe:PICTure:PATH ?

- **Function description:**

Set or query the PICTure save path.

- **Return format:**

Query and return the PICTure path.

- **For example:**

:FILe:PICTure:PATH "D:\\PICTure" Set the PICTure save path as D:\PICTure

:FILe:PICTure:PATH ? The query returned D:\\PICTure

:FILe:PICTure:SAVe

- **Command format:**

  :FILe:PICTure:SAVe ?

- **Function description:**

  Save the image data and return whether the saving was successful.

- **Return format:**

  Query and return the result as "SUCCess" or "FAIL".

Print

:PRINt:REGion

- **Command format:**

  :PRINt:REGion <region>

  :PRINt:REGion ?

- **Function description:**

  Set or query the print area.

- **Parameter:**

  region:{SCReen | GRID}

  SCReen:Screen

  GRID:Grid

- **Return format:**

  The query returns "SCReen", "GRID".

- **For example:**

  :PRINt:REGion SCReen Set the printing area to the screen.

  :PRINt:REGion ? The query returns SCReen.

:PRINt:COLor

- **Command format:**

  :PRINt:COLor <color>

  :PRINt:COLor ?

- **Function description:**

  Set or query the printing color of the printer.

- **Parameter:**

  color: {STANdard | BLACkwhite | REVerse}.

  STANdard: Standard

  BLACkwhite: Black and white

  REVerse: Reverse

- **Return format:**

  The query returns "STANdard ", "BLACkwhite", "REVerse".

- **For example:**

  :PRINt:COLor STANdard Set the print color to STANdard.

  :PRINt:COLor ? The query returns STANdard.

:PRINt:ORIent

- **Command format:**

  :PRINt:ORIent <orient>

  :PRINt:ORIent ?

- **Function description:**

  Set or query the printer's printing direction.

- **Parameter:**

  orient: {HOR | VER}.

  HOR: Horizontal

  VER: Vertical

- **Return format:**

  The query returns "HOR ", "VER".

- **For example:**

  :PRINt:ORIent HOR Set the printing direction to Horizontal.

  :PRINt:ORIent ? The query returns HOR.

LA

:LA:ITEM<n>:DISPlay

- **Command format:**

  :LA:ITEM<n>:DISPlay <active>

  :LA:ITEM<n>:DISPlay ?

- **Function description:**

  Set or query the open state of the specified logical channel.

- **Parameter:**

  n: Integer type, with the setting range from 0 to 15, representing logical channels D0 to D15.

  active: Boolean type {ON | OFF} or {1 | 0}

- **Return format:**

  The query returns that "1" and "0" represent "ON" and "OFF" respectively.

- **For example:**

  :LA:ITEM1:DISPlay ON Enable logical channel 1.

  :LA:ITEM1:DISPlay ? The query returns 1.

:LA:ITEM<n>:OFFSet

- **Command format:**

  :LA:ITEM<n>:OFFSet <offset>

  :LA:ITEM<n>:OFFSet?

- **Function description:**

  Set or query the vertical position offset of the specified logical channel.

- **Parameter:**

  n: Integer type, the setting range is 0 to 15, representing logical channels D0 to D15.

  offset: Real number type, the setting range is -3.5div to 4div.

- **Return format:**

  Query and return the vertical offset of the specified logical channel in scientific notation.

- **For example:**

  :LA:ITEM1:OFFSet 3

  Set the vertical position offset of logical channel 1 to 3div.

  :LA:ITEM1:OFFSet ?

  Query and return 3.00000E+000.

:LA:ITEM<n>:LABel

- **Command format:**

  :LA:ITEM<n>:LABel <lable>

  :LA:ITEM<n>:LABel ?

- **Function description:**

  Set or query the label for the specified logical channel.

- **Parameter:**

  n: Integer type, the setting range is 0 to 15, representing logical channels D0 to D15.

  lable: String of channel name.

- **Return format:**

  Query and return the label of the specified logical channel.

- **For example:**

  :LA:ITEM1:LABel “abcd” Assign the label abcd to logical channel 1.

  :LA:ITEM1:LABel ? Query and return abcd.

:LA:ITEM<n>:THReshold

- **Command format:**

  :LA:ITEM<n>:THReshold <threshold>

  :LA:ITEM<n>:THReshold ?

- **Function description:**

  Set or query the threshold level for the specified logical channel.

- **Parameter:**

  n: Integer type, the setting range is 0 - 15, representing logical channels D0 - D15.

  threshold: {TTL | CMOS5000 | CMOS3300 | CMOS2500 | CMOS1800 | ECL | PECL | LVDS | USER}

  TTL:TTL

  CMOS5000:5.0V COMS

  CMOS3300:3.3V COMS

  CMOS2500:2.5V COMS

  CMOS1800:1.8V COMS

  ECL:ECL

  PECL:PECL

  LVDS:LVDS

  USER:USER

- **Return format:**

  Query and return "TTL", "CMOS5000", "CMOS3300", "CMOS2500", "CMOS1800", "ECL", "PECL", "LVDS", "USER".

- **For example:**

  :LA:ITEM1:THReshold CMOS3300

  Set the threshold level attribute of logical channel 1 to CMOS3300.

  :LA:ITEM1:THReshold ?

  Query and return CMOS3300.

:LA:ITEM<n>:LEVel

- **Command format:**

  :LA:ITEM<n>:LEVel <level>

  :LA:ITEM<n>:LEVel ?

- **Function description:**

  Set or query the threshold level of the specified logical channel.

- **Parameter:**

  n: Integer type, the setting range is 0 - 15, representing logical channels D0 - D15.

  level: Real number type, the units are mV, V, and the default unit is V.

- **Return format:**

  Query and return the threshold level of the logical channel, using scientific notation.

- **For example:**

  :LA:ITEM1:LEVel 3 Set the threshold for logical channel 1 to 3.

  :LA:ITEM1:LEVel ? Query and return 3.00000E+000.

:LA:HEIGht

- **Command format:**

  :LA:HEIGht <height>

  :LA:HEIGht ?

- **Function description:**

  Set or query the height of the logical channel.

- **Parameter:**

  height: {SMALl | MEDium | LARGe}

  SMALl:Small

  MEDium:Medium

  LARGe:Large

- **Return format:**

Query and return "SMALl", "MEDium", "LARGe".

- **For example:**

  :LA:HEIGht LARGe Set the height of the logical channel to LARGe.

  :LA:HEIGht ? Query and return LARGe.

:LA:AUTO:LOCate

- **Command format:**

  :LA:AUTO:LOCate <auto>

- **Function description:**

  Set the logical channel to auto-sort.

- **Parameter:**

  auto: Boolean type {ON | OFF} or {1 | 0}.

- **For example:**

  :LA:AUTO:LOCate ON Set the automatic sorting of the logical channel.

:LA:ALL:DISPlay

- **Command format:**

  :LA:ALL:DISPlay <active>

- **Function description:**

  Enable all logical channels.

- **Parameter:**

  active: Boolean type {ON | OFF} or {1 | 0}

- **For example:**

  :LA:ALL:DISPlay ON Set all logical channels open.

Sequence mode

:SEGement:ACTive

- **Command format:**

  :SEGement:ACTive <active>

  :SEGement:ACTive ?

- **Function description:**

  Set or query the enabling status of the sequence mode. When the sequence mode is

enabled, UltraAcq will be automatically turned on.

- **Parameter:**

  active: Boolean type {ON | OFF} or {1 | 0}

- **Return format:**

  The query returns "1" and "0", representing "ON" and "OFF" respectively.

- **For example:**

  :SEGement:ACTive ON Enable sequence mode.

  :SEGement:ACTive ? The query returns 1.

:SEGement:RSTacq

- **Command format:**

  :SEGement:RSTacq <acq>

- **Function description:**

  Whether to re-sample in the setting sequence mode.

- **Parameter:**

  acq: Boolean type {ON | OFF} or {1 | 0}

  ON or 1: Resample

  OFF or 0: Do not resample

- **For example:**

  :SEGement:RSTacq ON Set the sequence mode to resample.

:SEGement:MODe

- **Command format:**

  :SEGement:MODe <mode>

  :SEGement:MODe ?

- **Function description:**

  Set or query the working mode of the sequence mode.

- **Parameter:**

  mode: {SINGle | SEQuent}

  SINGle: Single frame

  SEQuent: Continuous frame

- **Return format:**

  The query returns "SINGle", "SEQuent".

- **For example:**

  :SEGement:MODe SEQuent

  Set the working mode of the sequence mode to the continuous frame mode.

  :SEGement:MODe ?

  The query returns SEQuent.

:SEGement:FRAMe:COUNt

- **Command format:**

  :SEGement:FRAMe:COUNt <count>

  :SEGement:FRAMe:COUNt ?

- **Function description:**

  Set or query the total number of frames of the sequence mode.

- **Parameter:**

  count: Integer type

- **Return format:**

  The query returns the total number of frames in sequence mode.

- **For example:**

  :SEGement:FRAMe:COUNt 100

  Set the total number of frames in sequence mode to 100.

  :SEGement:FRAMe:COUNt ?

  The query returns 100.

:SEGement:SEQuent:TYPe

- **Command format:**

  :SEGement:SEQuent:TYPe <mode>

  :SEGement:SEQuent:TYPe ?

- **Function description:**

  Set or query the waveform display type for sequence mode in consecutive frames.

- **Parameter:**

  mode:{ANGLe | STACk | SUPerimpose | SPLiced | NONe}

  ANGLe:Angle

  STACk:Stack

  SUPerimpose:Superimpose

SPLiced:Spliced

NONe:None

- **Return format:**

  The query returns "ANGLe", "STACk", "SUPerimpose", "SPLiced", "NONe".

- **For example:**

  :SEGement:SEQuent:TYPe ANGLe

  Set the waveform display type of the sequence mode under consecutive frames to 45°.

  :SEGement:SEQuent:TYPe ?

  The query returns ANGLe.

:SEGement:FRAMe:STARt

- **Command format:**

  :SEGement:FRAMe:STARt <count>

  :SEGement:FRAMe:STARt ?

- **Function description:**

  Set or query the starting frame in consecutive frames mode.

- **Parameter:**

  count: Integer type

- **Return format:**

  The query returns the starting frame in sequence mode with consecutive frames.

- **For example:**

  :SEGement:FRAMe:STARt 1

  Set the starting frame of the sequence mode to 1.

  :SEGement:FRAMe:STARt ?

  The query returns 1.

:SEGement:FRAMe:END

- **Command format:**

  :SEGement:FRAMe:END <count>

  :SEGement:FRAMe:END ?

- **Function description:**

  Set or query the ending frame in sequence mode with consecutive frames.

- **Parameter:**

count: Integer type

- **Return format:**

  The query returns the ending frame in sequence mode with consecutive frames.

- **For example:**

  :SEGement:FRAMe:END 15 Set the ending frame in sequence mode to 15.

  :SEGement:FRAMe:END ? The query returns 15.

:SEGement:FRAMe:CURRent

- **Command format:**

  :SEGement:FRAMe:CURRent <count>

  :SEGement:FRAMe:CURRent ?

- **Function description:**

  Set or query the selected frame of the sequence mode in the single frame mode.

- **Parameter:**

  count: Integer type

- **Return format:**

  The query returns the selected frame of the sequence mode in the single frame mode.

- **For example:**

  :SEGement:FRAMe:CURRent 1

  Set the selected frame 1 of the sequence mode.

  :SEGement:FRAMe:CURRent ?

  The query returns 1.

:SEGement:FRAMe:REF

- **Command format:**

  :SEGement:FRAMe:REF <count>

  :SEGement:FRAMe:REF ?

- **Function description:**

  Set or query the reference frame in single frame mode.

- **Parameter:**

  count: Integer type

- **Return format:**

  Query and return the reference frame in single frame mode.

- **For example:**

  :SEGement:FRAMe:REF 15.

  Set the reference frame of the sequence mode to 15

  :SEGement:FRAMe:REF ?

  Query and return 15.

:SEGement:REFactive

- **Command format:**

  :SEGement:REFactive <active>

  :SEGement:REFactive ?

- **Function description:**

  Set or query the enabled status of the reference frame in the sequence mode.

- **Parameter:**

  active: Boolean type {ON | OFF} or {1 | 0}

- **Return format:**

  The query returns that "1" and "0" represent "ON" and "OFF" respectively.

- **For example:**

  :SEGement:REFactive ON

  Set the reference frame of the sequence mode to be open.

  :SEGement:REFactive ?

  The query returns 1.

:SEGement:CALLback

- **Command format:**

  :SEGement:CALLback <callback>

  :SEGement:CALLback ?

- **Function description:**

  Set or query the playback status in single frame mode.

- **Parameter:**

  callback: Boolean type {ON | OFF} or {1 | 0}

  ON or 1: Start playback

  OFF or 0: Stop playback

- **Return format:**

The query returns that "1" and "0" represent "ON" and "OFF" respectively.

- **For example:**

  :SEGement:CALLback ON Enable sequence mode playback.

  :SEGement:CALLback ? The query returns 1.

X-Y

:XY<n>:ACTive

- **Command format:**

  :XY<n>:ACTive <active>

  :XY<n>:ACTive ?

- **Function description:**

  Set or query whether the specified XY is turned on.

- **Parameter:**

  n: Integer, range 1 - 4

  active: Boolean {ON | OFF} or {1 | 0}

- **Return format:**

  The query returns that "1" and "0" represent "ON" and "OFF" respectively.

- **For example:**

  :XY1:ACTive ON Set XY1 to be turned on.

  :XY1:ACTive ? The query returns 1.

:XY<n>:SOURce:X

- **Command format:**

  :XY<n>:SOURce:X <source>

  :XY<n>:SOURce:X ?

- **Function description:**

  Set or query source X for the specified XY.

- **Parameter:**

  n: Integer, range 1 - 4

  source: Discrete {C1 | C2 | C3 | C4}

- **Return format:**

  The query returns "C1", "C2", "C3", "C4".

- **For example:**

  :XY1:SOURce:X C1 Set source X for XY1 to C1.

  :XY1:SOURce:X ? The query returns C1.

:XY<n>:SOURce:Y

- **Command format:**

  :XY<n>:SOURce:Y <source>

  :XY<n>:SOURce:Y ?

- **Function description:**

  Set or query the source Y of the specified XY.

- **Parameter:**

  n: Integer, range 1 - 4

  source: Discrete {C1 | C2 | C3 | C4}

- **Return format:**

  The query returns "C1", "C2", "C3", "C4".

- **For example:**

  :XY1:SOURce:Y C2 Set the source Y of XY1 as C2.

  :XY1:SOURce:Y ? The query returns C2.

:XY<n>:CURSor:ACTive

- **Command format:**

  :XY<n>:CURSor:ACTive <active>

  :XY<n>:CURSor:ACTive ?

- **Function description:**

  Set or query whether the cursor for the specified XY is enabled.

- **Parameter:**

  n: Integer, range 1 - 4

  active: Boolean type {ON | OFF} or {1 | 0}

- **Return format:**

  The query returns that "1" and "0" represent "ON" and "OFF" respectively.

- **For example:**

  :XY1:CURSor:ACTive ON Enable the cursor for XY1.

  :XY1:CURSor:ACTive ? The query returns 1.

:XY<n>:TYPe

- **Command format:**

  :XY<n>:TYPe <type>

  :XY<n>:TYPe ?

- **Function description:**

  Set or query the marking type of the specified XY.

- **Parameter:**

  n: Integer, range 1 to 4

  type: Discrete {CURSor | WAVe}

  CURSor: Cursor

  WAVe: Wave

- **Return format:**

  The query returns "CURSor", "WAVe".

- **For example:**

  :XY1:TYPe CURSor Set the marking type of XY1 as cursor.

  :XY1:TYPe ? The query returns CURSor.

:XY<n>:DATA:TYPe

- **Command format:**

  :XY<n>:DATA:TYPe <type>

  :XY<n>:DATA:TYPe ?

- **Function description:**

  Set or query the data type of the specified XY.

- **Parameter:**

  n: Integer, range 1 - 4

  type: Discrete {RECTangle | POLar}

  RECTangle: Rectangle

  POLar: Polar

- **Return format:**

  The query returns "RECTangle", "POLar".

- **For example:**

  :XY1:DATA:TYPe POLar Set the data type for XY1 to POLar.

:XY1:DATA:TYPe ? The query returns POLar.

:XY<n>:CURSor:CAX

- **Command format:**

  :XY<n>:CURSor:CAX <value>

  :XY<n>:CURSor:CAX ?

- **Function description:**

  Set or query the position of the vertical cursor A of the specified XY marker in the cursor mode.

- **Parameter:**

  n: Integer, range 1 to 4

  value: Boolean, real number in units of div, range 0 to 10

- **Return format:**

  The query returns the position of the vertical cursor A of the specified XY marker in the cursor mode.

- **For example:**

  :XY1:CURSor:CAX 2

  Set the position of the vertical cursor A of XY1 in the cursor mode to 2div.

  :XY1:CURSor:CAX ?

  The query returns 2.00000E+000.

:XY<n>:CURSor:CBX

- **Command format:**

  :XY<n>:CURSor:CBX <value>

  :XY<n>:CURSor:CBX ?

- **Function description:**

  Set or query the position of the vertical cursor B of the specified XY marker in the cursor mode.

- **Parameter:**

  n: Integer, range 1 - 4

  value: Boolean, real number in units of div, range 0 - 10

- **Return format:**

  The query returns the position of vertical cursor B for the specified XY marker in cursor

mode.

- **For example:**

  :XY1:CURSor:CBX 2

  Set the vertical cursor B position for XY1 in cursor mode to 2div.

  :XY1:CURSor:CBX ?

  The query returns 2.00000E+000.

:XY<n>:CURSor:CAY

- **Command format:**

  :XY<n>:CURSor:CAY <value>

  :XY<n>:CURSor:CAY ?

- **Function description:**

  Set or query the position of the horizontal cursor A of the specified XY marker in the cursor mode.

- **Parameter:**

  n: Integer, range from 1 to 4

  value: Boolean, real number in div as the unit, range from -4 to 4

- **Return format:**

  The query returns the position of horizontal cursor A for the specified XY marker in cursor mode.

- **For example:**

  :XY1:CURSor:CAY 2

  Set the horizontal cursor A position for XY1 in cursor mode to 2div.

  :XY1:CURSor:CAY ?

  The query returns 2.00000E+000.

:XY<n>:CURSor:CBY

- **Command format:**

  :XY<n>:CURSor:CBY <value>

  :XY<n>:CURSor:CBY ?

- **Function description:**

  Set or query the position of the horizontal cursor B of the specified XY marker in the cursor mode.

- **Parameter:**

  n: Integer, ranging from 1 to 4

  value: Boolean, real number in units of div, ranging from -4 to 4

- **Return format:**

  Query and return the position of the horizontal cursor B of the specified XY marker in the cursor mode.

- **For example:**

  :XY1:CURSor:CBY 2

  Set the position of the horizontal cursor B of XY1 in the cursor mode to 2div.

  :XY1:CURSor:CBY ?

  Query and return 2.00000E+000.

:XY<n>:CURSor:AXValue

- **Command format:**

  :XY<n>:CURSor:AXValue ?

- **Function description:**

  Query and return the rectangular data value of vertical cursor A for the specified XY marker in cursor mode. The unit is consistent with the source X.

- **Parameter:**

  n: Integer type, range 1 to 4

- **Return format:**

  Query and return the rectangular data value of vertical cursor A for the specified XY marker in cursor mode, using scientific notation. The default unit is V.

:XY<n>:CURSor:BXValue

- **Command format:**

  :XY<n>:CURSor:BXValue ?

- **Function description:**

  Query the rectangular data measurement value of the vertical cursor B of the specified XY marker in the cursor mode. The unit matches source X.

- **Parameter:**

  n: Integer type, range 1 to 4

- **Return format:**

Query and return the rectangular data measurement value of the vertical cursor B of the specified X marker in the cursor mode, using scientific notation. The default unit is V.

:XY<n>:CURSor:AYValue

- **Command format:**

  :XY<n>:CURSor:AYValue ?

- **Function description:**

  Query the rectangular data value of horizontal cursor A for the specified XY marker in cursor mode. The unit matches source X.

- **Parameter:**

  n: Integer, range 1 to 4

- **Return format:**

  Query and return the rectangular data measurement value of the horizontal cursor A of the specified XY marker in the cursor mode, using scientific notation. The default unit is V.

:XY<n>:CURSor:BYValue

- **Command format:**

  :XY<n>:CURSor:BYValue ?

- **Function description:**

  Query the rectangular data value of horizontal cursor B for the specified XY marker in cursor mode. The unit is consistent with the source Y.

- **Parameter:**

  n: Integer, range 1 to 4

- **Return format:**

  Return the rectangular data measurement value of the horizontal cursor B of the specified XY marker in the cursor mode, using scientific notation. The default unit is V.

:XY<n>:WAVe:AXValue

- **Command format:**

  :XY<n>:WAVe:AXValue ?

- **Function description:**

  Query the rectangular data value of vertical cursor A for the specified XY marker in

waveform mode. The unit is consistent with source X.

- **Parameter:**

  n: Integer, range 1 to 4

- **Return format:**

  Query and return the rectangular data measurement value of the vertical cursor A of the specified XY marker in the waveform mode, using scientific notation.

:XY<n>:WAVe:BXValue

- **Command format:**

  :XY<n>:WAVe:BXValue ?

- **Function description:**

  Query the rectangular data measurement value of the vertical cursor B of the specified XY marker in the waveform mode. The unit remains consistent with the source X.

- **Parameter:**

  n: Integer, range 1 to 4

- **Return format:**

  Query and return the rectangular data measurement value of the vertical cursor B of the specified XY marker in the waveform mode, using scientific notation.

:XY<n>:WAVe:AYValue

- **Command format:**

  :XY<n>:WAVe:AYValue ?

- **Function description:**

  Query the rectangular data value of horizontal cursor A for the specified XY marker in waveform mode. The unit is consistent with source Y.

- **Parameter:**

  n: Integer, range 1 to 4

- **Return format:**

  Query and return the rectangular data measurement value of the horizontal cursor A of the specified XY marker in the waveform mode, using scientific notation.

:XY<n>:WAVe:BYValue

- **Command format:**

:XY<n>:WAVe:BYValue ?

- **Function description:**

  Query the rectangular data value of horizontal cursor B for the specified XY marker in waveform mode. The unit is consistent with source Y.

- **Parameter:**

  n: Integer, range 1 to 4

- **Return format:**

  To query and return the rectangular data measurement value of the horizontal cursor B of the specified XY marker in the waveform mode, using scientific notation.

:XY<n>:RADius:A

- **Command format:**

  :XY<n>:RADius:A ?

- **Function description:**

  Query the radius of the polar coordinate system formed by the voltage values of source X and source Y at cursor A for the specified XY marker. The unit matches source X.

- **Parameter:**

  n: Integer, range 1 to 4

- **Return format:**

  Query and return the radius of the polar coordinate system composed of the voltage values of the source X and source Y of the specified XY at cursor A, using scientific notation.

:XY<n>:RADius:B

- **Command format:**

  :XY<n>:RADius:B ?

- **Function description:**

  To query the radius of the polar coordinate system composed of the voltage values of the source X and source Y of the specified XY at cursor B, and the unit remains consistent with the source X.

- **Parameter:**

  n: Integer, range 1 to 4

- **Return format:**

Query and return the radius of the polar coordinate system composed of the voltage values of the source X and source Y of the specified XY at cursor B, using scientific notation.

:XY<n>:ANGLe:A

- **Command format:**

  :XY<n>:ANGLe:A ?

- **Function description:**

  Query the included angle of the polar coordinate system formed by the voltage values of source X and source Y at cursor A for the specified XY marker, with the unit in degrees.

- **Parameter:**

  n: Integer, ranging from 1 to 4

- **Return format:**

  Query and return the included angle of the polar coordinate system composed of the voltage values of the source X and source Y of the specified XY at cursor A, using scientific notation.

:XY<n>:ANGLe:B

- **Command format:**

  :XY<n>:ANGLe:B ?

- **Function description:**

  Query the included angle of the polar coordinate system formed by the voltage values of source X and source Y at cursor B for the specified XY marker, with the unit in degrees.

- **Parameter:**

  n: Integer, range 1 to 4

- **Return format:**

  Query and return the included angle of the polar coordinate system composed of the voltage values of the source X and source Y of the specified XY at cursor B, using scientific notation.

ZOOM

:ZOOM:ACTive

- **Command format:**

  :ZOOM:ACTive <active>

  :ZOOM:ACTive ?

- **Function description:**

  Set or query whether the extension is turned on.

- **Parameter:**

  active: Boolean type {ON | OFF} or {1 | 0}

- **Return format:**

  The query returns the extension status. "1" and "0" represent "ON" and "OFF" respectively.

- **For example:**

  :ZOOM:ACTive ON Set to open the extension.

  :ZOOM:ACTive ? The query returns 1.

:ZOOM:CENTer:X

- **Command format:**

  :ZOOM:CENTer:X <value>

  :ZOOM:CENTer:X ?

- **Function description:**

  Set or query the vertical position of the zoom center.

- **Parameter:**

  value: Real number type, range 0 - 10000

- **Return format:**

  The query returns the vertical position of the zoom center.

- **For example:**

  :ZOOM:CENTer:X 5000 Set the vertical zoom center to 5000.

  :ZOOM:CENTer:X ? The query returns 5.00000E+003.

:ZOOM:CENTer:Y

- **Command format:**

:ZOOM:CENTer:Y <value>

:ZOOM:CENTer:Y ?

- **Function description:**

  Set or query the horizontal position of the zoom center.

- **Parameter:**

  value: Real number type, range -4000 - 4000

- **Return format:**

  The query returns the horizontal position of the zoom center.

- **For example:**

  :ZOOM:CENTer:Y 0

  Set the open extended horizontal center point to 0.

  :ZOOM:CENTer:Y ?

  The query returns 0.00000E+000.

:ZOOM:SCALe:X

- **Command format:**

  :ZOOM:SCALe:X <value>

  :ZOOM:SCALe:X ?

- **Function description:**

  Set or query the horizontal zoom ratio.

- **Parameter:**

  value: Real number type, range 1 - 1000

- **Return format:**

  The query returns the horizontal zoom ratio.

- **For example:**

  :ZOOM:SCALe:X 10 Set the horizontal zoom ratio to 10.

  :ZOOM:SCALe:X ? The query returns 1.00000E+001.

:ZOOM:SCALe:Y

- **Command format:**

  :ZOOM:SCALe:Y <value>

  :ZOOM:SCALe:Y ?

- **Function description:**

Set or query the vertical zoom ratio.

- **Parameter:**

  value: Real number type, range 1 - 1000

- **Return format:**

  The query returns the vertical zoom ratio.

- **For example:**

  :ZOOM:SCALe:Y 10

  Set the horizontal zoom center to 10 when enabled.

  :ZOOM:SCALe:Y ?

  The query returns 1.00000E+001.

Jitter analysis

:JITTer:ACTive

- **Command format:**

  :JITTer:ACTive <active>

  :JITTer:ACTive ?

- **Function description:**

  Set or query whether the jitter analysis is turned on.

- **Parameter:**

  active: Boolean type {ON | OFF} or {1 | 0}

- **Return format:**

  The query returns the on state of the jitter analysis. "1" and "0" represent "ON" and "OFF" respectively.

- **For example:**

  :JITTer:ACTive ON Set to turn on the jitter analysis.

  :JITTer:ACTive ? The query returns 1.

:JITTer:SOURce

- **Command format:**

  :JITTer:SOURce <source>

  :JITTer:SOURce ?

- **Function description:**

Set or query the jitter analysis data source.

- **Parameter:**

  source: Discrete type {C1 | C2 | C3 | C4}

- **Return format:**

  The query returns "C1", "C2", "C3", "C4".

- **For example:**

  :JITTer:SOURce C1 Set the jitter analysis data source to C1.

  :JITTer:SOURce ? The query returns C1.

:JITTer:SIGNal:TYPe

- **Command format:**

  :JITTer:SIGNal:TYPe <type>

  :JITTer:SIGNal:TYPe ?

- **Function description:**

  Set or query the signal type of jitter analysis.

- **Parameter:**

  Type: Discrete type {CUSTom | CLOCk}

  CUSTom: Data signal

  CLOCk: Clock signal

- **Return format:**

  The query returns "CUSTom", "CLOCk".

- **For example:**

  :JITTer:SIGNal:TYPe CLOCk

  Set the signal type of jitter analysis as the clock signal.

  :JITTer:SIGNal:TYPe ?

  The query returns CLOCk.

:JITTer:DTALength

- **Command format:**

  :JITTer:DTALength <value>

  :JITTer:DTALength ?

- **Function description:**

  Set or query the data signal length.

- **Parameter:**

  Value: Integer type, range 0 - 2Gbit, default unit bit

- **Return format:**

  Query and return the data length under the data signal type.

- **For example:**

  :JITTer:DTALength 127 Set the data signal length to 127bits.

  :JITTer:DTALength ? Query and return 127.

:JITTer:THReshold

- **Command format:**

  :JITTer:THReshold <value>

  :JITTer:THReshold ?

- **Function description:**

  Set or query the comparison threshold for jitter analysis. The threshold range is 45% to 55%.

- **Parameter:**

  value: Real number type

- **Return format:**

  Query and return the comparison threshold of jitter analysis.

- **For example:**

  :JITTer:THReshold 50

  Set the comparison threshold of jitter analysis to 50%.

  :JITTer:THReshold ?

  Query and return 5.00000E+001.

:JITTer:HYSTeresis

- **Command format:**

  :JITTer:HYSTeresis <value>

  :JITTer:HYSTeresis ?

- **Function description:**

  Set or query the hysteresis of jitter analysis.

- **Parameter:**

value: Real number type

- **Return format:**

  Query and return the hysteresis of jitter analysis.

- **For example:**

  :JITTer:HYSTeresis 30

  Set the hysteresis of jitter analysis to 30%.

  :JITTer:HYSTeresis ?

  Query and return 3.00000E+001.

:JITTer:BITRate

- **Command format:**

  :JITTer:BITRate <bit>

  :JITTer:BITRate?

- **Function description:**

  Set or query the jitter analysis bit rate.

- **Parameter:**

  Value: Real number type, range 0 - 5 Gbps, default unit bps

- **Return format:**

  Query and return the bit rate of jitter analysis.

- **For example:**

  :JITTer:BITRate 20 Set the jitter analysis bit rate to 20.

  :JITTer:BITRate ? Query and return 2.00000E+001.

:JITTer:FNDBitrate

- **Command format:**

  :JITTer:FNDBitrate <active>

- **Function description:**

  Set to search for the bit rate.

- **Parameter:**

  active: Boolean type {ON | OFF} or {1 | 0}

- **For example:**

  :JITTer:FNDBitrate ON Set to search for the bit rate.

:JITTer:CLOCk:TYPe

- **Command format:**

  :JITTer:CLOCk:TYPe <type>

  :JITTer:CLOCk:TYPe ?

- **Function description:**

  Set or query the jitter analysis clock recovery method.

- **Parameter:**

  type: Discrete type {CONStant | PLL}

  CONStant: Constant clock

  PLL: Phase-locked loop

- **Return format:**

  The query returns either "CONStant" or "PLL".

- **For example:**

  :JITTer:CLOCk:TYPe PLL

  Set the clock recovery method for jitter analysis as PLL.

  :JITTer:CLOCk:TYPe ?

  The query returns PLL.

:JITTer:PLL:TYPe

- **Command format:**

  :JITTer:PLL:TYPe <type>

  :JITTer:PLL:TYPe ?

- **Function description:**

  Set or query the jitter analysis PLL mode.

- **Parameter:**

  type: Discrete type {GOLDen | SECondOrder}

  GOLDen: Golden phase-locked loop

  SECondOrder: Second-order phase-locked loop

- **Return format:**

  The query returns "GOLDen", "SECondOrder".

- **For example:**

  :JITTer:PLL:TYPe GOLDen Set the jitter analysis PLL mode to GOLDen.

:JITTer:PLL:TYPe ? The query returns GOLDen.

:JITTer:PLL:DIVisor:CUToff

- **Command format:**

  :JITTer:PLL:DIVisor:CUToff <value>

  :JITTer:PLL:DIVisor:CUToff ?

- **Function description:**

  Set or query the cut-off coefficient of the golden section frequency phase-locked loop for jitter analysis.

- **Parameter:**

  value: Real number type

- **Return format:**

  The query returns the cut-off coefficient of the golden section frequency phase-locked loop for jitter analysis.

- **For example:**

  :JITTer:PLL:DIVisor:CUToff 100

  Set the cut-off coefficient of the golden phase-locked loop for jitter analysis to 100.

  :JITTer:PLL:DIVisor:CUToff ?

  The query returns 1.00000E+002.

:JITTer:PLL:FREQ:CUToff

- **Command format:**

  :JITTer:PLL:FREQ:CUToff <value>

  :JITTer:PLL:FREQ:CUToff ?

- **Function description:**

  Set or query the cut-off frequency of the golden section PLL for jitter analysis. The default unit is Hz.

- **Parameter:**

  value: Real number type, unit mHz, Hz

- **Return format:**

  Query and return the cut-off frequency of the golden section PLL for jitter analysis.

- **For example:**

  :JITTer:PLL:FREQ:CUToff 10

Set the cut-off frequency of the golden PLL for jitter analysis to 10Hz.

:JITTer:PLL:FREQ:CUToff ?

Query and return 1.0000000000000E+001.

:JITTer:PLL:FREQ:NATural

- **Command format:**

  :JITTer:PLL:FREQ:NATural <value>

  :JITTer:PLL:FREQ:NATural ?

- **Function description:**

  Set or query the natural frequency of the second-order PLL for jitter analysis. The default unit is Hz.

- **Parameter:**

  Value: Real number type, units mHz, Hz

- **Return format:**

  Query and return the natural frequency of the second-order PLL for jitter analysis.

- **For example:**

  :JITTer:PLL:FREQ:NATural 10

  The natural frequency of the second-order phase-locked loop for jitter analysis is set to 10Hz.

  :JITTer:PLL:FREQ:NATural ?

  Query and return 1.0000000000000E+001.

:JITTer:PLL:FACTor

- **Command format:**

  :JITTer:PLL:FACTor <value>

  :JITTer:PLL:FACTor ?

- **Function description:**

  Set or query the damping factor of the second-order phase-locked loop for jitter analysis.

- **Parameter:**

  value: Real number type

- **Return format:**

  Query and return the damping factor of the second-order phase-locked loop.

- **For example:**

:JITTer:PLL:FACTor 0.6

Set the damping factor of the second-order phase-locked loop for jitter analysis to 0.6.

:JITTer:PLL:FACTor ?

Query and return 6.00000E-001.

:JITTer:PARams:JITTer

- **Command format:**

  :JITTer:PARams:JITTer <active>

  :JITTer:PARams:JITTer ?

- **Function description:**

  Set or query whether the jitter Parameter for jitter analysis is turned on.

- **Parameter:**

  active: Boolean type {ON | OFF} or {1 | 0}

- **Return format:**

  Query and return the on status of the jitter Parameter for jitter analysis. "1" and "0" represent "ON" and "OFF" respectively.

- **For example:**

  :JITTer:PARams:JITTer ON

  Set the jitter Parameter for jitter analysis to be on.

  :JITTer:PARams:JITTer ?

  Query and return 1.

:JITTer:PARams:EYE

- **Command format:**

  :JITTer:PARams:EYE <active>

  :JITTer:PARams:EYE ?

- **Function description:**

  Set or query whether the Parameter of jitter analysis and eye diagram is on.

- **Parameter:**

  active: Boolean type {ON | OFF} or {1 | 0}

- **Return format:**

  Query and return the on state of the Parameter of the jitter analysis and eye diagram. "1" and "0" represent "ON" and "OFF" respectively.

- **For example:**

  :JITTer:PARams:EYE ON

  Set the Parameter of the jitter analysis eye diagram to be on.

  :JITTer:PARams:EYE ?

  Query and return 1.

:JITTer:HISTogram:BINNum

- **Command format:**

  :JITTer:HISTogram:BINNum <value>

  :JITTer:HISTogram:BINNum ?

- **Function description:**

  Set or query the number of histogram bins for jitter analysis.

- **Parameter:**

  Value: Discrete {25 | 50 | 100 | 250 | 500 | 2000 | MAX}

- **Return format:**

  Query and return the number of histogram bars of the jitter analysis.

- **For example:**

  :JITTer:HISTogram:BINNum 25

  Set the histogram bin count for jitter analysis to 25.

  :JITTer:HISTogram:BINNum ?

  Query and return 25.

:JITTer:GRAPh

- **Command format:**

  :JITTer:GRAPh <type> <enable>

- **Function description:**

  Set whether the jitter analysis graph is enabled.

- **Parameter:**

  type: Discrete type {TRENd | SPECtrum | HISTogram | EYE | BATHtub }

  TRENd: Trend

  SPECtrum: Spectrum

  HISTogram: Histogram

  EYE: Eye diagram

BATHtub: Bathtub curve

enable: Boolean type {ON | OFF} or {1 | 0}

- **For example:**

  :JITTer:GRAPh EYE,ON Enable the jitter analysis eye diagram.

:JITTer:DATA:JITTer

- **Command format:**

  :JITTer:DATA:JITTer ?

- **Function description:**

  Query the result of jitter analysis parameters and arrange them as follows:

  "TIE", "TJ", "RJ", "DJ", "PJ", "DDJ", "DCD", "ISI"

  TIE: Time Interval Error

  TJ: Total Jitter at a Specific Bit Error Rate

  RJ: Random Jitter

  DJ: Deterministic Jitter

  PJ: Periodic Jitter

  DDJ: Data-Dependent Jitter

  DCD: Duty Cycle Distortion

  ISI: Inter-Symbol Interference

- **Return format:**

  The query returns the result value of the jitter analysis parameters, conforms to the data block format, is expressed in scientific notation, and is arranged in sequence and separated by commas. Invalid values are represented by the maximum value of the real data type.

  for example: “#90000001001.20000E+000,2.00000E+002,1.20000E+003.....”

:JITTer:DATA:EYE

- **Command format:**

  :JITTer:DATA:EYE ?

- **Function description:**

  Query and return the eye diagram parameters arranged as follows: "ZeroLevel", "OneLevel", "EyeAmplitude", "EyeHeight", "EyeWidth", "ExtinctionRatio", "QFactor", "EyeCrossRatio"

  ZeroLevel: 0 level

  OneLevel: 1 level

EyeAmplitude: Eye amplitude

EyeHeight: Eye height

EyeWidth: Eye width

ExtinctionRatio: Extinction ratio

QFactor: Q factor

EyeCrossRatio: Eye cross ratio

- **Return format:**

  The query returns the eye diagram parameter values in data block format, expressed in scientific notation and arranged sequentially, separated by commas. Invalid values are represented by the maximum value of the real data type.

  For example: "#90000001001.20000E+000, 2.00000E+002, 1.20000E+003...."

PASS/FAIL

:PF:ACTive

- **Command format:**

  :PF:ACTive <active>

  :PF:ACTive ?

- **Function description:**

  Set or query the pass/fail test status.

- **Parameter:**

  active: Boolean type {ON | OFF} or {1 | 0}

- **Return format:**

  The query returns the pass/fail test status. "1" represents "ON" and "0" represents "OFF".

- **For example:**

  :PF:ACTive ON Set to enable the pass/fail test.

  :PF:ACTive ? The query returns 1.

:PF:VISible

- **Command format:**

  :PF:VISible <visible>

  :PF:VISible ?

- **Function description:**

Set or query whether the test template of the pass/fail test is displayed.

- **Parameter:**

  visible: Boolean type {ON | OFF} or {1 | 0}

- **Return format:**

  The query returns whether the pass/fail test template is displayed. "1" represents "ON" and "0" represents "OFF".

- **For example:**

  :PF:VISible ON

  Set the display template for the pass/fail test to be displayed.

  :PF:VISible ?

  The query returns 1.

:PF:MODe

- **Command format:**

  :PF:MODe <mode>

  :PF:MODe ?

- **Function description:**

  Set or query the type of pass/fail test.

- **Parameter:**

  mode: Discrete type {LIMitMode | STDMaskmode}

  LIMitMode: Limit test

  STDMaskmode: Standard test

- **Return format:**

  The query returns the type of pass/fail test.

- **For example:**

  :PF:MODe LIMitMode Set the type of pass/fail test as limit test.

  :PF:MODe ? The query returns LIMitMode.

:PF:SOURce

- **Command format:**

  :PF:SOURce <source>

  :PF:SOURce ?

- **Function description:**

Set or query the pass/fail test data source.

- **Parameter:**

  source: Discrete {C1 | C2 | C3 | C4}

- **Return format:**

  The query returns the data source of the pass/fail test.

- **For example:**

  :PF:SOURce C1 Set the pass/fail data source to C1.

  :PF:SOURce ? The query returns C1.

:PF:LIMit:SOURce

- **Command format:**

  :PF:LIMit:SOURce <source>

  :PF:LIMit:SOURce ?

- **Function description:**

  Set or query the reference source for the pass/fail limit test.

- **Parameter:**

  Source: Discrete {C1 | C2 | C3 | C4}

- **Return format:**

  The query returns the reference source for the pass/fail limit test.

- **For example:**

  :PF:LIMit:SOURce C1

  Set the reference source for the pass/fail limit test to C1.

  :PF:LIMit:SOURce ?

  The query returns C1.

:PF:LIMit:VTOLerance

- **Command format:**

  :PF:LIMit:VTOLerance <value>

  :PF:LIMit:VTOLerance?

- **Function description:**

  Set or query the vertical tolerance for the pass/fail limit test. The default unit is mdiv.

- **Parameter:**

  Value: Integer type, range 1mdiv - 1div

- **Return format:**

  The query returns the vertical capacity of the pass/fail limit test template.

- **For example:**

  :PF:LIMit:VTOLerance 100

  Set the vertical tolerance for the pass/fail limit test to 100mdiv.

  :PF:LIMit:VTOLerance ?

  The query returns 100.

:PF:LIMit:HTOLerance

- **Command format:**

  :PF:LIMit:HTOLerance <value>

  :PF:LIMit:HTOLerance ?

- **Function description:**

  Set or query the horizontal capacity of the pass/fail limit test template. The default unit is mdiv.

- **Parameter:**

  Value: Integer type, with the unit range from 1mdiv to 500mdiv

- **Return format:**

  Query and return the horizontal capacity of the pass/fail limit test template.

- **For example:**

  :PF:LIMit:HTOLerance 100

  Set the horizontal capacity of the pass/fail limit test template to 100mdiv.

  :PF:LIMit:HTOLerance ?

  Query and return 100.

:PF:LIMit:CREate

- **Command format:**

  :PF:LIMit:CREate

- **Function description:**

  Create the pass/fail limit test template.

:PF:STANdard:TYPe

- **Command format:**

:PF:STANdard:TYPe <type>

:PF:STANdard:TYPe ?

- **Function description:**

  Set or query the pass/fail standard test type.

- **Parameter:**

  Type: Discrete {ANSI | ITU | USB}

  ANSI:ANSI T1.102

  ITU:ITU-T

  USB:USB

- **Return format:**

  Query and return the pass/fail standard test type.

- **For example:**

  :PF:STANdard:TYPe ANSI Set the pass/fail standard test type ANSI.

  :PF:STANdard:TYPe ? Query and return ANSI.

:PF:STANdard:LOCK

- **Command format:**

  :PF:STANdard:LOCK <lock>

  :PF:STANdard:LOCK ?

- **Function description:**

  Set or query whether the pass/fail standard test template is locked.

- **Parameter:**

  lock: Boolean type {ON | OFF} or {1 | 0}

  ON or 1: Locked

  OFF or 0: Unlocked

- **Return format:**

  Query and return the lock status of the pass/fail standard test template. "1" and "0" represent "ON" and "OFF" respectively.

- **For example:**

  :PF:STANdard:LOCK ON

  Set the lock of the pass/fail standard test template.

  :PF:STANdard:LOCK ?

  Query and return 1.

:PF:STANdard:INDex

- **Command format:**

  :PF:STANdard:INDex <index>

  :PF:STANdard:INDex ?

- **Function description:**

  Set or query the index of the pass/fail standard test.

- **Parameter:**

  index: Integer type

- **Return format:**

  Query and return the index of the pass/fail standard test template.

- **For example:**

  :PF:STANdard:INDex 10

  Set the index of the pass/fail standard test to 10.

  :PF:STANdard:INDex ?

  Query and return 10.

:PF:STOP:TYPe

- **Command format:**

  :PF:STOP:TYPe <type>

  :PF:STOP:TYPe ?

- **Function description:**

  Set or query the end conditions for fail tests.

- **Parameter:**

  type: Discrete type {WFMs | TIMe}

  WFMs: Total number of waveforms

  TIMe: Total time

- **Return format:**

  Query and return the end conditions for failed tests.

- **For example:**

  :PF:STOP:TYPe WFMs Set the end condition for failed tests to WFMs.

  :PF:STOP:TYPe ? Query and return WFMs.

:PF:WFMS:COUNt

- **Command format:**

  :PF:WFMS:COUNt <value>

  :PF:WFMS:COUNt ?

- **Function description:**

  Set or query the total number of waveforms when the end condition for failed tests is set to waveforms.

- **Parameter:**

  Value: Integer type, ranging from 1 to 100,000

- **Return format:**

  Query and return the total number of waveforms when the end condition for failed tests is waveforms.

- **For example:**

  :PF:WFMS:COUNt 1000

  Set the total waveform count for fail tests to 1000.

  :PF:WFMS:COUNt ?

  Query and return 1000.

:PF:TIMe:DURation

- **Command format:**

  :PF:TIMe:DURation <value>

  :PF:TIMe:DURation ?

- **Function description:**

  Set or query the total time when the end condition for failed tests is set to time. The default unit is seconds.

- **Parameter:**

  Value: Real number type, range 100ms - 1Ms

- **Return format:**

  Query and return the total time when the end condition for failed tests is time.

- **For example:**

  :PF:TIMe:DURation 100ms Set the total time for fail tests to 100ms.

  :PF:TIMe:DURation ? Query and return 1.0000000000000E-001.

:PF:VIOLations:COUNt

- **Command format:**

  :PF:VIOLations:COUNt <value>

  :PF:VIOLations:COUNt ?

- **Function description:**

  Set or query the total number of violations for failed tests.

- **Parameter:**

  value: Integer type, range 1 - 1000

- **Return format:**

  Query and return the total number of violations for failed tests.

- **For example:**

  :PF:VIOLations:COUNt 10 Set the total number of violations for failed tests to 10.

  :PF:VIOLations:COUNt ? Query and return 10.

:PF:OPTion:SAVe

- **Command format:**

  :PF:OPTion:SAVe <save>

  :PF:OPTion:SAVe ?

- **Function description:**

  Set or query whether the failed operations of the failed tests are saved.

- **Parameter:**

  save: Boolean type {ON | OFF} or {1 | 0}

  ON: Save

  OFF: Do not save

- **Return format:**

  Query whether the failed operations of the failed tests are saved. "1" and "0" represent "ON" and "OFF" respectively.

- **For example:**

  :PF:OPTion:SAVe ON Set the failed save of the failed tests.

  :PF:OPTion:SAVe ? Query returns 1.

:PF:OPTion:BEEP

- **Command format:**

  :PF:OPTion:BEEP <beep>

  :PF:OPTion:BEEP ?

- **Function description:**

  Set or query whether to enable the alarm for fail tests.

- **Parameter:**

  beep: Boolean type {ON | OFF} or {1 | 0}

  ON: Alarm

  OFF: No alarm

- **Return format:**

  Query whether to alarm for the failed operations of the failed tests. "1" and "0" represent "ON" and "OFF" respectively.

- **For example:**

  :PF:OPTion:BEEP ON Enable the alarm for fail test operations.

  :PF:OPTion:BEEP ? Query returns 1.

:PF:OPTion:PULSe

- **Command format:**

  :PF:OPTion:PULSe <pulse>

  :PF:OPTion:PULSe ?

- **Function description:**

  Set or query whether to output pulses for fail test operations.

- **Parameter:**

  pulse: Boolean type {ON | OFF} or {1 | 0}

  ON: Output pulse

  OFF:Do not output the pulse

- **Return format:**

  Query to return whether to output pulses for the failed operations of the failed tests. "1" and "0" represent "ON" and "OFF" respectively.

- **For example:**

  :PF:OPTion:PULSe ON

Enable pulse output for fail test operations.

:PF:OPTion:PULSe ?

Query and return 1.

:PF:OPTion:HARDcopy

- **Command format:**

  :PF:OPTion:HARDcopy <hardcopy>

  :PF:OPTion:HARDcopy ?

- **Function description:**

  Set or query whether to hard copy for the failed operations of the failed tests.

- **Parameter:**

  Hardcopy: Boolean type {ON | OFF} or {1 | 0}

  ON: Hard copy

  OFF: No hard copy

- **Return format:**

  Query and return whether the failed operations of the failed tests are hard copied. "1" and "0" represent "ON" and "OFF" respectively.

- **For example:**

  :PF:OPTion:HARDcopy ON

  Set to turn on hard copy for the failed operations of the failed tests.

  :PF:OPTion:HARDcopy ?

  Query and return 1.

:PF:RUN

- **Command format:**

  :PF:RUN <run>

  :PF:RUN ?

- **Function description:**

  Set or query the running status of fail tests.

- **Parameter:**

  run: Boolean type {ON | OFF} or {1 | 0}

  ON or 1: Running

  OFF or 0: Stopped

- **Return format:**

  Query and return the running status of the failed tests. "1" and "0" represent "ON" and "OFF" respectively.

- **For example:**

  :PF:RUN ON Set the running status of fail tests to ON.

  :PF:RUN ? Query and return 1.

:PF:DATA

- **Command format:**

  :PF:DATA ?

- **Function description:**

  Query the pass/fail test status table results.

- **Return format:**

  The query returns the pass/fail test result values, which conforms to the data block format and is arranged in sequence and separated by commas. Invalid values are represented by the maximum value of real data.

  For example: “#9000000147

  Test Status,OFF,

  Total Waveform,0,

  Total Violations,0,

  Testing Time,0 s,

  Number of Hits,0,

  Hits Per Segment:

  1,0

  2,0

  3,0

  4,0

  5,0

  6,0

  7,0”

Search

:SEARch<n>:ACTive

- **Command format:**

  :SEARch<n>:ACTive <active>

  :SEARch<n>:ACTive ?

- **Function description:**

  Set or query the status of the specified search.

- **Parameter:**

  n: Integer, range 0 - 10

  active: Boolean {ON | OFF} or {1 | 0}

- **Return format:**

  Query and return the on status of the specified search. "1" and "0" represent "ON" and "OFF" respectively.

- **For example:**

  :SEARch1:ACTive ON Turn on search1.

  :SEARch1:ACTive ? Query and return 1.

:SEARch<n>:SOURce

- **Command format:**

  :SEARch<n>:SOURce <source>

  :SEARch<n>:SOURce ?

- **Function description:**

  Set or query the data source of the specified search.

- **Parameter:**

  n: Integer, range 0 - 10

  source: Discrete {C1 | C2 | C3 | C4}

- **Return format:**

  Query and return the data source of the specified search.

- **For example:**

  :SEARch1:SOURce C1 Set the data source of search1 as C1.

  :SEARch1:SOURce ? Query and return C1.

:SEARch<n>:TYPe

- **Command format:**

  :SEARch<n>:TYPe <type>

  :SEARch<n>:TYPe ?

- **Function description:**

  Set or query the search type for the specified search.

- **Parameter:**

  n: Integer type, range 0 - 10

  type: Discrete type{EDGe | PULSe}

  EDGe: edge

  PULSe: pulse

- **Return format:**

  Query to return the search type of the specified search.

- **For example:**

  :SEARch1:TYPe EDGe Set the search type for search1 to EDGe.

  :SEARch1:TYPe ? Query to return EDGe.

:SEARch<n>:VISible

- **Command format:**

  :SEARch<n>:VISible <visible>

  :SEARch<n>:VISible ?

- **Function description:**

  Set or query the flag status of the specified search.

- **Parameter:**

  n: Integer, range 0 - 10

  visible: Boolean {ON | OFF} or {1 | 0}

- **Return format:**

  Query to return the on status of the flag of the specified search. "1" and "0" represent "ON" and "OFF" respectively.

- **For example:**

  :SEARch1:VISible ON Set the flag of search1 to ON.

  :SEARch1:VISible ? Query returns 1.

:SEARch<n>:EVENt

- **Command format:**

  :SEARch<n>:EVENt <enable>

  :SEARch<n>:EVENt ?

- **Function description:**

  Set or query whether the event list of the specified search is on.

- **Parameter:**

  n: Integer, range 0 - 10

  enable: Boolean {ON | OFF} or {1 | 0}

- **Return format:**

  Query the on state of the event list of the specified search returned. "1" and "0" represent "ON" and "OFF" respectively.

- **For example:**

  :SEARch1:EVENt ON Set the event list of search1 to be on.

  :SEARch1:EVENt ? Query returns 1.

:SEARch<n>:EDGe:SLOPe

- **Command format:**

  :SEARch<n>:EDGe:SLOPe <slope>

  :SEARch<n>:EDGe:SLOPe ?

- **Function description:**

  Set or query the edge type for the specified search.

- **Parameter:**

  n: Integer, range 0 - 10

  slope: Discrete {RISe | FALL | ANY}

  RISe : Rising edge

  FALL: Falling edge

  ANY: Any edge

- **Return format:**

  Query and return the edge type when the specified search is an edge search.

- **For example:**

  :SEARch1:EDGe:SLOPe FALL

Set the edge type for search1 to FALL.

:SEARch1:EDGe:SLOPe ?

Query and return FALL.

:SEARch<n>:EDGe:LEVel

- **Command format:**

  :SEARch<n>:EDGe:LEVel <level>

  :SEARch<n>:EDGe:LEVel ?

- **Function description:**

  Set or query the search level when the specified search is an edge search. The default unit is V.

- **Parameter:**

  n: Integer, range 0 - 10

  level: Real number type, units mV, V. The search level range is consistent with the edge trigger level range.

- **Return format:**

  Query and return the search level when the specified search is an edge search.

- **For example:**

  :SEARch1:EDGe:LEVel 100mV

  Set the search level of edge search for search1 to 100mV.

  :SEARch1:EDGe:LEVel ?

  Query and return 1.00000E-001.

:SEARch<n>:PULSe:POLarity

- **Command format:**

  :SEARch<n>:PULSe:POLarity <polarity>

  :SEARch<n>:PULSe:POLarity ?

- **Function description:**

  Set or query the polarity for the specified pulse width search.

- **Parameter:**

  n: Integer, range 0 - 10

  polarity: Discrete type {POSitive | NEGative}

- **Return format:**

Query and return the search polarity when the specified search is a pulse width search.

- **For example:**

  :SEARch1:PULSe:POLarity NEGative

  Set the pulse width search polarity for search1 to NEGative.

  :SEARch1:PULSe:POLarity ?

  Query and return NEGative.

:SEARch<n>:PULSe:CONDition

- **Command format:**

  :SEARch<n>:PULSe:CONDition <condition>

  :SEARch<n>:PULSe:CONDition ?

- **Function description:**

  Set or query the search conditions when the specified search is a pulse width search.

- **Parameter:**

  n: Integer, range 0 - 10

  condition: Discrete type {GREAterthan | LESSthan | RANGe}

  GREAterthan: Greater than

  LESSthan: Less than

  RANGe: Within the range

- **Return format:**

  Query and return the search conditions when the specified search is a pulse width search.

- **For example:**

  :SEARch1:PULSe:CONDition LESSthan

  Set the search condition of search1 pulse width search to LESSthan.

  :SEARch1:PULSe:CONDition ?

  Query and return LESSthan.

:SEARch<n>:PULSe:TIMe:UPPer

- **Command format:**

  :SEARch<n>:PULSe:TIMe:UPPer <width>

  :SEARch<n>:PULSe:TIMe:UPPer ?

- **Function description:**

  Set or query the upper pulse width limit for the specified search. The default unit is s.

- **Parameter:**

  n: Integer, range 0 - 10

  width: Real number type, units ns, us, ms, s

- **Return format:**

  Query and return the upper limit of the pulse width when the specified search is a pulse width search, using scientific notation.

- **For example:**

  :SEARch1:PULSe:TIMe:UPPer 2us

  Set the upper pulse width limit for search1 to 2us.

  :SEARch1:PULSe:TIMe:UPPer ?

  Query and return 2.0000000000000E-006.

:SEARch<n>:PULSe:TIMe:LOWer

- **Command format:**

  :SEARch<n>:PULSe:TIMe:LOWer <width>

  :SEARch<n>:PULSe:TIMe:LOWer ?

- **Function description:**

  Set or query the lower pulse width limit for the specified search. The default unit is s.

- **Parameter:**

  n: Integer, range 0 - 10

  width: Real number type, units ns, us, ms, s

- **Return format:**

  Query and return the lower limit of the pulse width when the specified search is a pulse width search, using scientific notation.

- **For example:**

  :SEARch1:PULSe:TIMe:LOWer 1us

  Set the lower pulse width limit for search1 to 1us.

  :SEARch1:PULSe:TIMe:LOWer ?

  Query and return 1.0000000000000E-006.

:SEARch<n>:PULSe:LEVel

- **Command format:**

  :SEARch<n>:PULSe:LEVel <level>

:SEARch<n>:PULSe:LEVel ?

- **Function description:**

  Set or query the threshold level when the specified search is a pulse width search. The default unit is V.

- **Parameter:**

  n: Integer, range 0 - 10

  level: Real number type, units mV, V. The threshold level range is consistent with the pulse width trigger level range.

- **Return format:**

  Query and return the threshold level when the specified search is a pulse width search.

- **For example:**

  :SEARch1:PULSe:LEVel 100mV

  Set the threshold level of the pulse width search of search1 to 100mV.

  :SEARch1:PULSe:LEVel ?

  Query and return 1.00000E-001.

:SEARch<n>:DATA

- **Command format:**

  :SEARch<n>:DATA ?

- **Function description:**

  Set the query to specify the result value of the search event list.

- **Return format:**

  The query returns all the result values of the specified search event list. conforming to the data block format, expressed in scientific notation, and arranged in sequence separated by commas. Invalid values are represented by the maximum value of the real data type.

  For example: "#9000000549index,type,position,delta,description,

  0,Edge,-2.4000000000000E-006,0.0000000000000E+000,Edge,

  1,Edge,-2.3000000000000E-006,1.0000000000000E-007,Edge,

  2,Edge,-2.2000000000000E-006,1.0000000000000E-007,Edge,"

Power analysis

:PWAS:ADD

- **Command format:**

  :PWAS:ADD <type>,<vsource>,<csource>

- **Function description:**

  Add a specified type of power analysis.

- **Parameter:**

  type:{QUALity | HARMonic | RIPPle | SWITchingloss | SAFeoperationarea | LOOPanalysis}

  QUALity: Power quality

  HARMonic: Harmonic analysis

  RIPPle: Ripple analysis

  SWITchingloss: Switching loss

  SAFeoperationarea: Safe operation area

  LOOPanalysis: Loop analysis

  Modulation：modulation analysis

  InrushCurrent：surge current

  PowerEfficency：Power Efficiency

  RDSon：On resistance

  TurnOnOff：Opening/closing time

  PSRR：Power Supply Rejection Ratio

  SlewRate：Slew rate

  vsource: Voltage source, discrete {C1 | C2 | C3 | C4}

  csource: Current source, discrete {C1 | C2 | C3 | C4}

- **For example:**

  :PWAS:ADD QUALity,C1,C2

  Add power quality analysis with voltage source C1 and current source C2.

:PWAS:ALLPoweranalysis

- **Command format:**

  :PWAS:ALLPoweranalysis ?

- **Function description:**

Query all the opened power analyses.

- **Return format:**

  The query returns all active power analyses, listing POWER1, its type, and additional active analyses, ..., POWER<n>, power analysis type.

- **For example:**

  :PWAS:ALLPoweranalysis ?

  The query returns POWER1, QUALity, POWER2, RIPPle.

:PWAS:DELete

- **Command format:**

  :PWAS:DELete <n>

- **Function description:**

  Delete a specified power analysis.

- **Parameter:**

  n: Integer. The value of n is less than or equal to the number of opened power analyses, within the range of 1 to 8.

- **For example:**

  :PWAS:DELete 1 Delete POWER1.

:POWer<n>:MODe

- **Command format:**

  :POWer<n>:MODe ?

- **Function description:**

  Query the power analysis type of the specified POWER.

- **Parameter:**

  n: Integer, range from 1 to 8

- **Return format:**

  Query and return the power analysis type of the specified POWER.

  QUALity: Power quality

  HARMonic: Harmonic analysis

  RIPPle: Ripple analysis

  SWITchingloss: Switching loss

  SAFeoperationarea: Safe operation area

LOOPanalysis: Loop analysis

Modulation: modulation analysis

InrushCurrent: surge current

PowerEfficency: Power Efficiency

RDSon: On resistance

TurnOnOff: Opening/closing time

PSRR: Power Supply Rejection Ratio

SlewRate: Slew rate

:POWer<n>:SOURce:VOLTage

- **Command format:**

  :POWer<n>:SOURce:VOLTage <source>

  :POWer<n>:SOURce:VOLTage ?

- **Function description:**

  Set or query the voltage source for the specified power analysis.

- **Parameter:**

  n: Integer, range 1 - 8

  source: Discrete {C1 | C2 | C3 | C4}

- **Return format:**

  Query and return the voltage source of the specified POWER.

- **For example:**

  :POWer1:SOURce:VOLTage C1 Set the voltage source for POWER1 to C1.

  :POWer1:SOURce:VOLTage ? Query and return C1.

:POWer<n>:SOURce:CURRent

- **Command format:**

  :POWer<n>:SOURce:CURRent <source>

  :POWer<n>:SOURce:CURRent ?

- **Function description:**

  Set or query the current source of the specified POWER.

- **Parameter:**

  n: Integer, range 1 to 8

  source: Discrete {C1 | C2 | C3 | C4}

- **Return format:**

  Query and return the current source of the specified POWER.

- **For example:**

  :POWer1:SOURce:CURRent C2 Set the current source of POWER1 to C2.

  :POWer1:SOURce:CURRent ? Query and return C2.

:POWer<n>:DATA

- **Command format:**

  :POWer<n>:DATA ?

- **Function description:**

  Query the measured result value for the specified POWER.

- **Parameter:**

  n: Integer, range from 1 to 8

- **Return format:**

  The query returns the measured result value of the specified POWER, conforming to the data block format, expressed in scientific notation, and arranged in sequence separated by commas. Invalid values are represented by the maximum value of the real data type.

  For example: "#9000000564QUALity,

  Value,Average,Maxumum,Minimum,

  1.78612E-003,1.99920E-003,2.47743E-003,1.53923E-003,

  1.95003E+000,2.51101E+000,4.00265E+000,1.22652E+000,

  1.79769E+308,1.79769E+308,1.79769E+308,1.79769E+308,

  1.78612E-003,1.99920E-003,2.47743E-003,1.53923E-003,

  1.95003E+000,2.51199E+000,4.00265E+000,1.22652E+000,

  4.02983E-006,4.00127E-006,4.18383E-006,3.83827E-006,

  3.19021E-006,4.01689E-006,6.13767E-006,2.36924E-006,

  NaN,1.90541E-006,4.68634E-006,8.00995E-008,

  1.26319E+000,1.01649E+000,1.65799E+000,6.45764E-001,

  NaN,2.47287E+001,4.97770E+001,1.16113E+000,"

Power quality

:QUALity<n>:FREQuency:REFerence

- **Command format:**

  :QUALity<n>:FREQuency:REFerence <type>

  :QUALity<n>:FREQuency:REFerence ?

- **Function description:**

  Set or query the frequency reference for the specified power quality analysis.

- **Parameter:**

  n: Integer, range 1 to 8

  type: Discrete {VOLTage | CURRent}

  VOLTage: Voltage source

  CURRent: Current source

- **Return format:**

  Query and return the frequency reference for the specified power quality analysis.

- **For example:**

  :QUALity1:FREQuency:REFerence CURRent

  Set the frequency reference of QUALity1 as the current source.

  :QUALity1:FREQuency:REFerence ?

  Query and return CURRent.

Harmonic analysis

:HARMonic<n>:COUNt

- **Command format:**

  :HARMonic<n>:COUNt <value>

  :HARMonic<n>:COUNt ?

- **Function description:**

  Set or query the harmonic count for the specified analysis.

- **Parameter:**

  value: Integer type, range 40 - 200

- **Return format:**

Query and return the number of harmonics of the specified harmonic analysis.

- **For example:**

  :HARMonic1:COUNt 50 Set the harmonic count for HARMonic1 to 50.

  :HARMonic1:COUNt ? Query and return 50.

:HARMonic<n>:FREQuency:REFerence

- **Command format:**

  :HARMonic<n>:FREQuency:REFerence <type>

  :HARMonic<n>:FREQuency:REFerence ?

- **Function description:**

  Set or query the frequency reference for the specified harmonic analysis.

- **Parameter:**

  n: Integer, range 1 to 8

  type: Discrete type {HARMonicsrc | VOLTage | CURRent | CONStant}

  HARMonicsrc: Harmonic source

  VOLTage: Voltage source

  CURRent: Current source

  CONStant: Constant

- **Return format:**

  Query and return the frequency reference of the specified harmonic analysis.

- **For example:**

  :HARMonic1:FREQuency:REFerence CURRent

  Set the frequency reference of HARMonic1 to the current source.

  :HARMonic1:FREQuency:REFerence ?

  Query and return CURRent.

:HARMonic<n>:FREQuency:CONStant

- **Command format:**

  :HARMonic<n>:FREQuency:CONStant <value>

  :HARMonic<n>:FREQuency:CONStant ?

- **Function description:**

  Set or query the frequency when the frequency reference of the specified harmonic analysis is fixed. The default unit is Hz.

- **Parameter:**

  n: Integer, range 1 to 8

  value: Real number type, units Hz, KHz, MHz, GHz.

- **Return format:**

  Query and return the frequency when the frequency reference of the specified harmonic analysis is fixed, and return it in scientific notation.

- **For example:**

  :HARMonic1:FREQuency:CONStant 120

  Set the fixed frequency reference of HARMonic1 to 120Hz.

  :HARMonic1:FREQuency:CONStant ?

  Query and return 1.2000000000000E+002.

:HARMonic<n>:SOURce

- **Command format:**

  :HARMonic<n>:SOURce <source>

  :HARMonic<n>:SOURce ?

- **Function description:**

  Set or query the harmonic source for the specified harmonic analysis.

- **Parameter:**

  n: Integer type, range 1 to 8

  source: Discrete type {VOLTage | CURRent}

  VOLTage: Voltage

  CURRent: Current

- **Return format:**

  Query and return the harmonic source of the specified harmonic analysis.

- **For example:**

  :HARMonic1:SOURce CURRent

  Set the harmonic source for HARMonic1 to current.

  :HARMonic1:SOURce ?

  Query and return CURRent.

Safe operating area

:SAFeoperationarea<n>:STOPonfail

- **Command format:**

  :SAFeoperationarea<n>:STOPonfail <active>

  :SAFeoperationarea<n>:STOPonfail ?

- **Function description:**

  Set or query whether to stop when a violation of the specified safe operating area occurs.

- **Parameter:**

  n: Integer, range 1 to 8

  active: Boolean type {ON | OFF} or {1 | 0}

  ON: Stop

  OFF: Do not stop

- **Return format:**

  Query whether to stop when a violation occurs in the specified safe operating area. "1" represents "ON" and "0" represents "OFF".

- **For example:**

  :SAFeoperationarea1:STOPonfail ON

  Set to stop when there is a violation in SAFeoperationarea1.

  :SAFeoperationarea1:STOPonfail ?

  Query and return 1.

:SAFeoperationarea<n>:AXIS:XMAX

- **Command format:**

  :SAFeoperationarea<n>:AXIS:XMAX <value>

  :SAFeoperationarea<n>:AXIS:XMAX ?

- **Function description:**

  Set or query the maximum X-axis value for the specified safe operating area. The default unit is V.

- **Parameter:**

  n: Integer, range 1 - 8

  value: Real number type, range -3.99V - 10kV

- **Return format:**

  Query and return the maximum X-axis value for the specified safe operating area.

- **For example:**

  :SAFeoperationarea1:AXIS:XMAX 5

  Set the maximum value of the X-axis of SAFeoperationarea1 to 5V.

  :SAFeoperationarea1:AXIS:XMAX ?

  Query and return 5.00000E+000.

:SAFeoperationarea<n>:AXIS:XMIN

- **Command format:**

  :SAFeoperationarea<n>:AXIS:XMIN <value>

  :SAFeoperationarea<n>:AXIS:XMIN ?

- **Function description:**

  Set or query the minimum value of the X-axis of the specified safe working area. The default unit is V.

- **Parameter:**

  n: Integer, range 1 - 8

  value: Real number type, range -10kV - 7.99V

- **Return format:**

  Query and return the minimum value of the X-axis of the specified safe working area.

- **For example:**

  :SAFeoperationarea1:AXIS:XMIN 1

  Set the minimum value of the X-axis of SAFeoperationarea1 to 1V.

  :SAFeoperationarea1:AXIS:XMIN ?

  Query and return 1.00000E+000.

:SAFeoperationarea<n>:AXIS:YMAX

- **Command format:**

  :SAFeoperationarea<n>:AXIS:YMAX <value>

  :SAFeoperationarea<n>:AXIS:YMAX ?

- **Function description:**

  Set or query the maximum Y-axis value for the specified safe operating area. The default unit is A.

- **Parameter:**

  n: Integer, range 1 - 8

  value: Real number type,range 10mA - 10kA

- **Return format:**

  Query and return the maximum Y-axis value for the specified safe operating area.

- **For example:**

  :SAFeoperationarea1:AXIS:YMAX 5

  Set the maximum value of the Y-axis of SAFeoperationarea1 to 5A.

  :SAFeoperationarea1:AXIS:YMAX ?

  Query and return 5.00000E+000.

:SAFeoperationarea<n>:AXIS:YMIN

- **Command format:**

  :SAFeoperationarea<n>:AXIS:YMIN <value>

  :SAFeoperationarea<n>:AXIS:YMIN ?

- **Function description:**

  Set or query the minimum value of the Y-axis of the specified safe working area. The default unit is A.

- **Parameter:**

  n: Integer, range 1 to 8

  value: Real number type, range -10kA to 5.29A

- **Return format:**

  Query and return the minimum value of the Y-axis of the specified safe working area.

- **For example:**

  :SAFeoperationarea1:AXIS:YMIN 1

  Set the minimum value of the Y-axis of SAFeoperationarea1 to 1A.

  :SAFeoperationarea1:AXIS:YMIN ?

  Query and return 1.00000E+000.

Loop analysis

:LOOPanalysis<n>:SOURce:AWG

- **Command format:**

:LOOPanalysis<n>:SOURce:AWG <source>

:LOOPanalysis<n>:SOURce:AWG ?

- **Function description:**

  Set or query the AWG source for the specified loop analysis.

- **Parameter:**

  n: Integer, range 1 - 8

  source: Discrete {G1 | G2}

- **Return format:**

  Query and return the AWG source of the specified loop analysis.

- **For example:**

  :LOOPanalysis1:SOURce:AWG G1

  Set the AWG source for LOOPanalysis1 to G1.

  :LOOPanalysis1:SOURce:AWG ?

  Query and return G1.

:LOOPanalysis<n>:IMPedance

- **Command format:**

  :LOOPanalysis<n>:IMPedance <type>

  :LOOPanalysis<n>:IMPedance ?

- **Function description:**

  Set or query the impedance of the specified loop analysis.

- **Parameter:**

  n: Integer, range 1 - 8

  type: Discrete {LOW | HIGH}

  LOW: 50Ω

  HIGH: High impedance

- **Return format:**

  Query and return the impedance of the specified loop analysis.

- **For example:**

  :LOOPanalysis1:IMPedance HIGH

  Set the impedance of LOOPanalysis1 to high impedance.

  :LOOPanalysis1:IMPedance ?

  Query and return HIGH.

:LOOPanalysis<n>:FREQuency:STARt

- **Command format:**

  :LOOPanalysis<n>:FREQuency:STARt <value>

  :LOOPanalysis<n>:FREQuency:STARt ?

- **Function description:**

  Set or query the start frequency for the specified loop analysis. The default unit is Hz.

- **Parameter:**

  n: Integer, range 1 - 8

  value: Real number type, unit Hz, kHz, MHz

- **Return format:**

  Query and return the start frequency for the specified loop analysis, and return it in scientific notation.

- **For example:**

  :LOOPanalysis1:FREQuency:STARt 100

  Set the start frequency of LOOPanalysis1 to 100Hz.

  :LOOPanalysis1:FREQuency:STARt ?

  Query and return 1.0000000000000E+002.

:LOOPanalysis<n>:FREQuency:STOP

- **Command format:**

  :LOOPanalysis<n>:FREQuency:STOP <value>

  :LOOPanalysis<n>:FREQuency:STOP ?

- **Function description:**

  Set or query the stop frequency of the specified loop analysis. The default unit is Hz.

- **Parameter:**

  n: Integer, range 1 - 8

  value: Real number type, units Hz, kHz, MHz

- **Return format:**

  Query and return the stop frequency of the specified loop analysis, returned in scientific notation.

- **For example:**

  :LOOPanalysis1:FREQuency:STOP 1MHz

Set the stop frequency of LOOPanalysis1 to 1MHz.

:LOOPanalysis1:FREQuency:STOP ?

Query and return 1.0000000000000E+006.

:LOOPanalysis<n>:SCANnum

- **Command format:**

  :LOOPanalysis<n>:SCANnum <value>

  :LOOPanalysis<n>:SCANnum ?

- **Function description:**

  Set or query the number of scan points of the specified loop analysis.

- **Parameter:**

  n: Integer, range 1 to 8

  value: Integer, range 1 to 1k

- **Return format:**

  Query and return the number of scan points of the specified loop analysis.

- **For example:**

  :LOOPanalysis1:SCANnum 30

  Set the number of scan points of LOOPanalysis1 to 30.

  :LOOPanalysis1:SCANnum ?

  Query and return 30.

:LOOPanalysis<n>:AMPMode

- **Command format:**

  :LOOPanalysis<n>:AMPMode <mode>

  :LOOPanalysis<n>:AMPMode ?

- **Function description:**

  Set or query the amplitude mode for the specified loop analysis.

- **Parameter:**

  n: Integer, range 1 - 8

  mode: Discrete {CONStant | VARiable}

  CONStant: Constant

  VARiable: Variable

- **Return format:**

Query and return the amplitude mode of the specified loop analysis.

- **For example:**

  :LOOPanalysis1:AMPMode CONStant

  Set the amplitude mode of LOOPanalysis1 as constant.

  :LOOPanalysis1:AMPMode ?

  Query and return CONStant.

:LOOPanalysis<n>:AMPValue

- **Command format:**

  :LOOPanalysis<n>:AMPValue <value>

  :LOOPanalysis<n>:AMPValue ?

- **Function description:**

  Set or query the amplitude for the specified loop analysis. The default unit is V.

- **Parameter:**

  n: Integer, range 1 to 8

  value: Real number type, units mV, V

- **Return format:**

  Query and return the amplitude of the specified loop analysis and return it in scientific notation.

- **For example:**

  :LOOPanalysis1:AMPValue 1 Set the amplitude of LOOPanalysis1 to 1V.

  :LOOPanalysis1:AMPValue ? Query and return 1.00000E+000.

Modulation analysis

:MODUlation<n>:SOURce

- **Command format:**

  :MODUlation<n>:SOURce <type>

  :MODUlation<n>:SOURce ?

- **Function description:**

  Set or query the specified modulation analysis source type。

- **Parameter:**

n:integer，range 1~8

type:Discrete type{VOLTage | CURRent}

VOLTage:Voltage source

CURRent:Current source

- **Return format:**

  Query and return the specified modulation analysis source type。

- **For example:**

  :MODUlation1:SOURce VOLTage

  Set the modulation source of MODULATION 1 to VOLTage

  :MODUlation1:SOURce ? Query and return VOLTage

:MODUlation<n>:HISTogram:SOURce

- **Command format:**

  :MODUlation<n>:HISTogram:SOURce <type>

  :MODUlation<n>:HISTogram:SOURce ?

- **Function description:**

  Set or query the type of modulation analysis histogram specified。

- **Parameter:**

  n:integer，range 1~8

  type:Discrete type{PERiod | FREQuency | PDuty | NDuty | PWidth | NWidth | RISEtime | FALLtime

  }

  PERiod:cycle

  FREQuency:frequency

  PDuty:Positive duty cycle

  NDuty:Negative duty cycle

  PWidth:Positive pulse width

  NWidth:Negative pulse width

  RISEtime:rise time

  FALLtime:fall time

- **Return format:**

  Query and return the histogram type of the specified modulation analysis。

- **For example:**

:MODUlation1:HISTogram:SOURce PERiod

Set the modulation source of MODULATION 1 to PERiod

:MODUlation1:HISTogram:SOURce ? Query returnPERiod

:MODUlation<n>:TRENd:SOURce

- **Command format:**

  :MODUlation<n>:TRENd:SOURce <type>

  :MODUlation<n>:TRENd:SOURce ?

- **Function description:**

  Set or query the type of modulation analysis trend chart specified。

- **Parameter:**

  n:integer，range 1~8

  type:Discrete type{PERiod | FREQuency | PDuty | NDuty | PWidth | NWidth | RISEtime | FALLtime

  }

  PERiod:cycle

  FREQuency:frequency

  PDuty:Positive duty cycle

  NDuty:Negative duty cycle

  PWidth:Positive pulse width

  NWidth:Negative pulse width

  RISEtime:rise time

  FALLtime:fall time

- **Return format:**

  Query and return the trend chart type of the specified modulation analysis。

- **For example:**

  :MODUlation1:TRENd:SOURce PERiod Set the modulation source of MODULATION 1 to PERiod

  :MODUlation1:TRENd:SOURce ? Query returnPERiod

:MODUlation<n>:HISTogram:ADD

- **Command format:**

  :MODUlation<n>:HISTogram:ADD

- **Function description:**

  Set modulation analysis to open histogram。

- **Parameter:**

  n:integer，range 1~8

- **For example:**

  :MODUlation1:HISTogram:ADD Open histogram

:MODUlation<n>:TRENd:ADD

- **Command format:**

  :MODUlation<n>:TRENd:ADD

- **Function description:**

  Set modulation analysis to open trend chart。

- **Parameter:**

  n:integer，range 1~8

- **For example:**

  :MODUlation1:TRENd:ADD Open the trend chart

Surge current

:INRUsh<n>:PEAKcurrent

- **Command format:**

  :INRUsh<n>:PEAKcurrent <value>

  :INRUsh<n>:PEAKcurrent ?

- **Function description:**

  Set or query the peak current value of the specified surge current。

- **Parameter:**

  n:integer，range 1~8

  value:real range ： -55V-55V unit mV、V

- **Return format:**

  Query and return the peak current value of the specified surge current。

- **For example:**

  :INRUsh1:PEAKcurrent 1 Set the peak current value of surge current to 1V

:INRUsh1:PEAKcurrent ? Query return 1.00000E+000

:INRUsh<n>:RUN

- **Command format:**

  :INRUsh<n>:RUN

- **Function description:**

  Run surge current。

- **Parameter:**

  n:integer，range 1~8

- **For example:**

  :INRUsh1:RUN Run surge current

Power Efficiency

:EFFIciency<n>:INPut:VOLTage

- **Command format:**

  :EFFIciency<n>:INPut:VOLTage <type>

  :EFFIciency<n>:INPut:VOLTage ?

- **Function description:**

  Set or query the input voltage source with specified power efficiency。

- **Parameter:**

  n:integer，range 1~8

  type:Discrete type{C1 | C2 | C3 | C4}

- **Return format:**

  Query and return the input voltage source with the specified power efficiency。

- **For example:**

  :EFFIciency1:INPut:VOLTage C1 Set the input voltage source C1 for power efficiency

  :EFFIciency1:INPut:VOLTage ? Query return C1

:EFFIciency<n>:INPut:CURRent

- **Command format:**

:EFFIciency<n>:INPut:CURRent <type>

:EFFIciency<n>:INPut:CURRent ?

- **Function description:**

  Set or query the input current source with specified power efficiency。

- **Parameter:**

  n:integer，range 1~8

  type:Discrete type{C1 | C2 | C3 | C4}

- **Return format:**

  Query returnInput current source with specified power efficiency。

- **For example:**

  :EFFIciency1:INPut:CURRent C2　　Set the input current source C2 for power efficiency

  :EFFIciency1:INPut:CURRent ?　　Query return C2

:EFFIciency<n>:OUTput:VOLTage

- **Command format:**

  :EFFIciency<n>:OUTput:VOLTage <type>

  :EFFIciency<n>:OUTput:VOLTage ?

- **Function description:**

  Set or query the output voltage source with specified power efficiency。

- **Parameter:**

  n:integer，range 1~8

  type:Discrete type{C1 | C2 | C3 | C4}

- **Return format:**

  Query return specifies the output voltage source for power efficiency。

- **For example:**

  :EFFIciency1:OUTput:VOLTage C3　Set the output voltage source C3 for power efficiency

  :EFFIciency1:OUTput:VOLTage ?　　Query return C3

:EFFIciency<n>:OUTput:CURRent

- **Command format:**

  :EFFIciency<n>:OUTput:CURRent <type>

  :EFFIciency<n>:OUTput:CURRent ?

- **Function description:**

Set or query the output current source with specified power efficiency。

- **Parameter:**

  n:integer，range 1~8

  type:Discrete type{C1 | C2 | C3 | C4}

- **Return format:**

  Query return specifies the output current source for power efficiency。

- **For example:**

  :EFFIciency1:OUTput:CURRent C1 Set the output current source C1 for power efficiency

  :EFFIciency1:OUTput:CURRent ? Query returnC1

:EFFIciency<n>:INPut:DIAGram

- **Command format:**

  :EFFIciency<n>:INPut:DIAGram

- **Function description:**

  Open the input power diagram。

- **Parameter:**

  n:integer，range 1~8

- **For example:**

  :EFFIciency1:INPut:DIAGram Open the input power diagram

:EFFIciency<n>:OUTput:DIAGram

- **Command format:**

  :EFFIciency<n>:OUTput:DIAGram

- **Function description:**

  Open the output power diagram。

- **Parameter:**

  n:integer，range 1~8

- **For example:**

  :EFFIciency1:OUTput:DIAGram Open the output power diagram

Rds（on)

:RDS<n>:WAVEform

- **Command format:**

  :RDS<n>:WAVEform

- **Function description:**

  Open RDS waveform diagram。

- **Parameter:**

  n:integer，range 1~8

- **For example:**

  :RDS1:WAVEform　　Open RDS waveform diagram

Turnon/Turnoff time

:TIMEonoff<n>:INPut:VOLTage

- **Command format:**

  :TIMEonoff<n>:INPut:VOLTage <value>

  :TIMEonoff<n>:INPut:VOLTage ?

- **Function description:**

  Set or query the input peak voltage value for the specified on/off time。

- **Parameter:**

  n:integer，range 1~8

  value:real range　-55V-55V unit mV、V

- **Return format:**

  Query return specifies the input peak voltage value for the on/off time。

- **For example:**

  :TIMEonoff1:INPut:VOLTage 1　　Set the input peak voltage value for the on/off time to 1V

  :TIMEonoff1:INPut:VOLTage ?　　Query return 1.00000E+000

:TIMEonoff<n>:OUTput:VOLTage

- **Command format:**

  :TIMEonoff<n>:OUTput:VOLTage <value>

  :TIMEonoff<n>:OUTput:VOLTage ?

- **Function description:**

  Set or query the peak output voltage value for a specified on/off time。

- **Parameter:**

  n:integer，range 1~8

  value:real range -55V-55V unit mV、V

- **Return format:**

  Query return specifies the peak voltage value of the output for the on/off time。

- **For example:**

  :TIMEonoff1:OUTput:VOLTage 1 Set the output peak voltage value for the on/off time to 1V

  :TIMEonoff1:OUTput:VOLTage ? Query return 1.00000E+000

:TIMEonoff<n>:TESTtype

- **Command format:**

  :TIMEonoff<n>:TESTtype <type>

  :TIMEonoff<n>:TESTtype ?

- **Function description:**

  Set or query test types with specified on/off times。

- **Parameter:**

  n:integer，range 1~8

  Type:Discrete type {Turnon | Turnoff}

  TurnOn:on-time

  TurnOff:closing time

- **Return format:**

  Query returnTest type specifying on/off time。

- **For example:**

  :TIMEonoff1:TESTtype Turnon The test type for setting the on/off time is the on time

:TIMEonoff1:TESTtype ? Query return Turnon

:TIMEonoff<n>:Typeconvert

- **Command format:**

  :TIMEonoff<n>:Typeconvert <type>

  :TIMEonoff<n>:Typeconvert ?

- **Function description:**

  Set or query the conversion type for specifying the on/off time。

- **Parameter:**

  n:integer，range 1~8

  Type:Discrete type {DC2DC | AC2DC}

  DC2DC:DC/DC

  AC2DC:AC/DC

- **Return format:**

  Query returnSpecify the conversion type for opening/closing time。

- **For example:**

  :TIMEonoff1:Typeconvert DC2DC The conversion type for setting the on/off time is DC/DC

  :TIMEonoff1:Typeconvert ? Query return DC2DC

:TIMEonoff<n>:ACQTime

- **Command format:**

  :TIMEonoff<n>:ACQTime <value>

  :TIMEonoff<n>:ACQTime ?

- **Function description:**

  Set or query the collection time value for the specified on/off time。

- **Parameter:**

  n:integer，range 1~8

  value:real 200ms-1000s unit ms,s

- **Return format:**

  Query return Specifies the collection time value of the open/close time, which is returned in Scientific notation。

- **For example:**

:TIMEonoff1:ACQTime 1	Set the collection time value for the on/off time to 1 second
:TIMEonoff1:ACQTime ?	Query return1.00000E-003

PSRR

:PSRR<n>:AWGSource

- **Command format:**

  :PSRR<n>:AWGSource <source>

  :PSRR<n>:AWGSource ?

- **Function description:**

  Set or query AWG source for specified PSRR.

- **Parameter:**

  n:integer，range 1~8

  source:Discrete type {G1 | G2}

- **Return format:**

  Query return AWG sources with a specified PSRR.

- **For example:**

:PSRR1:AWGSource G1	Query return specifies the AWG source of PSRR
:PSRR1:AWGSource ?	Query return G1

:PSRR<n>:IMPedance

- **Command format:**

  :PSRR<n>:IMPedance <type>

  :PSRR<n>:IMPedance ?

- **Function description:**

  Set or query the impedance type of the specified PSRR。

- **Parameter:**

  n:integer，range 1~8

  type:Discrete type {LOW | HIGH}

  LOW:Low

  HIGH:High

- **Return format:**

  Query return specifies the impedance type of PSRR。

- **For example:**

  :PSRR1:IMPedance LOW Set the AWG source for the specified PSRR to 50Ω

  :PSRR1:IMPedance ? Query return LOW

:PSRR<n>:TRIGger:STATus

- **Command format:**

  :PSRR<n>:TRIGger:STATus <type>

  :PSRR<n>:TRIGger:STATus ?

- **Function description:**

  Set or query the trigger status check for a specified PSRR。

- **Parameter:**

  n:integer，range 1~8

  type:Discrete type {OFF | ON}

- **Return format:**

  Query returnthe trigger status check for the specified PSRR.

- **For example:**

  :PSRR1:TRIGger:STATus ON Query return specifies the triggering state check of PSRR

  :PSRR1:TRIGger:STATus ? Query return ON

:PSRR<n>:STARtfrequency

- **Command format:**

  :PSRR<n>:STARtfrequency <value>

  :PSRR<n>:STARtfrequency ?

- **Function description:**

  Set or query the starting frequency value of the specified PSRR。

- **Parameter:**

  n:integer，range 1~8

  value:real range 10Hz-50MHz unit mHz、Hz

- **Return format:**

  Query return Specifies the starting frequency of PSRR, which is returned in Scientific

notation。

- **For example:**

  :PSRR1:STARtfrequency 50 Set the starting frequency value of PSRR to 50Hz

  :PSRR1:STARtfrequency ? Query return 5.0000000000000E+001

:PSRR<n>:ENDFrequency

- **Command format:**

  :PSRR<n>:ENDFrequency <value>

  :PSRR<n>:ENDFrequency ?

- **Function description:**

  Set or query the end frequency value of the specified PSRR。

- **Parameter:**

  n:integer，range 1~8

  value:real range 10Hz-50MHz unit mHz、Hz

- **Return format:**

  Query return Specifies the end frequency value of PSRR, which is returned in Scientific notation。

- **For example:**

  :PSRR1:STARtfrequency 50 Set the end frequency value of PSRR to 50Hz

  :PSRR1:STARtfrequency ? Query return5.0000000000000E+001

:PSRR<n>:SCANnumber

- **Command format:**

  :PSRR<n>:SCANnumber <value>

  :PSRR<n>:SCANnumber ?

- **Function description:**

  Set or query the number of scanning points for a specified PSRR。

- **Parameter:**

  n:integer，range 1~8

  value:real range 1-1K

- **Return format:**

  Query return specifies the number of scanning points for PSRR。

- **For example:**

:PSRR1:SCANnumber 10	Set the number of scanning points for PSRR to 10
:PSRR1:SCANnumber ?	Query return 10

:PSRR<n>:AMPMode

- **Command format:**

  :PSRR<n>:AMPMode <mode>

  :PSRR<n>:AMPMode ?

- **Function description:**

  Set or query the amplitude mode of the specified PSRR。

- **Parameter:**

  n:integer，range 1~8

  mode:Discrete type{CONStant | VARiable}

  CONStant:constant

  VARiable:variable

- **Return format:**

  Query return specifies the amplitude mode of PSRR。

- **For example:**

:PSRR1:AMPMode CONStant	Set the amplitude mode of PSRR to constant
:PSRR1:AMPMode ?	Query return CONStant

:PSRR<n>:AMPLitude

- **Command format:**

  :PSRR<n>:AMPLitude <value>

  :PSRR<n>:AMPLitude ?

- **Function description:**

  Set or query the amplitude of the specified PSRR, with the default unit being mV。

- **Parameter:**

  n:integer，range 1~8

  value:real，unit mV

- **Return format:**

  Query return Specifies the range of PSRR, which is returned in Scientific notation。

- **For example:**

:PSRR1:AMPLitude 1000 Set the amplitude of PSRR to 1V

:PSRR1:AMPLitude ? Query return 1.00000E+003

:PSRR<n>:BODepicture

- **Command format:**

  :PSRR<n>:BODepicture
- **Function description:**

  Open PSRR Bode plot。
- **Parameter:**

  n:integer，range 1~8
- **For example:**

  :PSRR1:BODepicture Open PSRR Bode plot

:PSRR<n>:RUN

- **Command format:**

  :PSRR<n>:RUN
- **Function description:**

  Run PSRR。
- **Parameter:**

  n:integer，range 1~8
- **For example:**

  :PSRR1:RUN Run PSRR

Slew rate

:SLEWrate<n>:DIAgram:DVDT

- **Command format:**

  :SLEWrate<n>:DIAgram:DVDT
- **Function description:**

Open the conversion rate dv/dt graph。

- **Parameter:**

  n:integer，range 1~8

- **For example:**

  :SLEWrate1:DIAgram:DVDT Open the conversion rate dv/dt graph

:SLEWrate<n>:DIAgram:DIDT

- **Command format:**

  :SLEWrate<n>:DIAgram:DIDT

- **Function description:**

  Open the conversion rate di/dt graph。

- **Parameter:**

  n:integer，range 1~8

- **For example:**

  :SLEWrate1:DIAgram:DIDT Open the conversion rate di/dt graph

REF

:REFerence<n>:ACTive

- **Command format:**

  :REFerence<n>:ACTive <active>，path

  :REFerence<n>:ACTive ?

- **Function description:**

  Set or query whether the specified reference channel is enabled.

- **Parameter:**

  n: Integer, range from 0 to 4

  active: Boolean type {ON | OFF} or {1 | 0}

- **Return format:**

  The query returns the status of the specified reference channel. "1" and "0" represent "ON" and "OFF" respectively.

- **For example:**

:REFerence1:ACTive ON,“C:\\Users\\Administrator\\Desktop\\Uni-t000.bin”

Set to open REFerence1.

:REFerence1:ACTive ?

The query returns 1.

:REFerence<n>:LABel

- **Command format:**

  :REFerence<n>:LABel <lable>

  :REFerence<n>:LABel ?

- **Function description:**

  Set or query the label for the specified reference channel.

- **Parameter:**

  n: Integer, range from 0 to 4

  lable: String

- **Return format:**

  The query returns the label for the specified reference channel.

- **For example:**

  :REFerence1:LABel “QWER”

  Set the label of REF1 as QWER.

  :REFerence1:LABel ?

  The query returns QWER.

:REFerence<n>:VERTical:OFFSet

- **Command format:**

  :REFerence<n>:VERTical:OFFSet <offset>

  :REFerence<n>:VERTical:OFFSet ?

- **Function description:**

  Set or query the vertical displacement of the specified reference channel.

- **Parameter:**

  n: Integer, range from 0 to 4

  offset: Real type, the unit is div, and the range is -4div - 4div

- **Return format:**

  The query returns the vertical displacement of the specified reference channel, using

scientific notation.

- **For example:**

  :REFerence1:VERTical:OFFSet 1.2

  Set the vertical displacement of REF1 to 1.2div.

  :REFerence1:VERTical:OFFSet ?

  The query returns 1.20000E+000.

:REFerence<n>:VERTical:SCALe

- **Command format:**

  :REFerence<n>:VERTical:SCALe <scale>

  :REFerence<n>:VERTical:SCALe ?

- **Function description:**

  Set or query the vertical scale of the specified reference channel.

- **Parameter:**

  n: Integer, range from 0 to 4

  scale: Real type, the unit is the unit when saving the waveform.

- **Return format:**

  The query returns the vertical scale of the specified reference channel, using scientific notation.

- **For example:**

  :REFerence1:VERTical:SCALe 1

  Set the vertical scale of REF1 to 1.

  :REFerence1:VERTical:SCALe ?

  The query returns 1.00000E+000.

:REFerence<n>:HORizontal:OFFSet

- **Command format:**

  :REFerence<n>:HORizontal:OFFSet <offset>

  :REFerence<n>:HORizontal:OFFSet ?

- **Function description:**

  Set or query the horizontal displacement for the specified reference channel.

- **Parameter:**

  n: Integer, range from 0 to 4

offset: Real type, the unit is div, and the range is 0div - 10div.

- **Return format:**

  The query returns the horizontal displacement of the specified reference channel, using scientific notation.

- **For example:**

  :REFerence1:HORizontal:OFFSet 1.2

  Set the horizontal displacement of REF1 to 1.2div.

  :REFerence1:HORizontal:OFFSet ?

  The query returns 1.20000E+000.

:REFerence<n>:HORizontal:SCALe

- **Command format:**

  :REFerence<n>:HORizontal:SCALe <scale>

  :REFerence<n>:HORizontal:SCALe ?

- **Function description:**

  Set or query the horizontal scale of the specified reference channel.

- **Parameter:**

  n: Integer, range from 0 to 4

  scale: Real type, unit: us, ms, s

- **Return format:**

  The query returns the horizontal scale of the specified reference channel, using scientific notation.

- **For example:**

  :REFerence1:HORizontal:SCALe 1

  Set the horizontal scale of REF1 to 1.

  :REFerence1:HORizontal:SCALe ?

  The query returns 1.00000E+000.

BUS

:BUS<n>:ACTive

- **Command format:**

  :BUS<n>:ACTive <active>

:BUS<n>:ACTive ?

- **Function description:**

  Set or query whether BUS is on.

- **Parameter:**

  n: Integer, within the range of 1 to 2

  active: Boolean type {ON | OFF} or {1 | 0}

- **Return format:**

  The query returns the on state of the specified BUS. "1" and "0" represent "ON" and "OFF" respectively.

- **For example:**

  :BUS1:ACTive 1 Turn on BUS1.

  :BUS1:ACTive ? The query returns 1.

:BUS<n>:TYPe

- **Command format:**

  :BUS<n>:TYPe <type>

  :BUS<n>:TYPe ?

- **Function description:**

  Set or query the decoding type of the specified BUS.

- **Parameter:**

  n: Integer, ranging from 1 to 2

  type: Discrete type {Close | RS232 | I2C | SPI | CAN | CANFd (CAN-FD Bus) | LIN | FLEXray | AUDioBus | MIL (MIL-STD-1553 Bus) | ARINc429 | SENT}

- **Return format:**

  The query returns the decoding type of the specified BUS.

- **For example:**

  :BUS1:TYPe RS232 Set the decoding type of BUS1 as RS232.

  :BUS1:TYPe ? The query returns RS232.

:BUS<n>:FORMat

- **Command format:**

  :BUS<n>:FORMat <type>

  :BUS<n>:FORMat ?

- **Function description:**

  Set or query the decoding display format of the specified BUS.

- **Parameter:**

  n: Integer, ranging from 1 to 2

  type: Discrete type {HEX | DEC | BINARY | ASCII | AUTO}

  HEX:Hex

  DEC:Dec

  BINARY:Binary

  ASCII:ASCII

  AUTO:Auto

- **Return format:**

  The query returns the decoding display format of the specified BUS.

- **For example:**

  :BUS1:FORMat HEX

  Set the decoding display format of BUS1 as HEX.

  :BUS1:FORMat ?

  The query returns HEX.

:BUS<n>:POSition

- **Command format:**

  :BUS<n>:POSition <value>

  :BUS<n>:POSition ?

- **Function description:**

  Set or query the decoding display position of the specified BUS. The default unit is div.

- **Parameter:**

  n: Integer, ranging from 1 to 2

  value: Real type, from -4div to 4div

- **Return format:**

  The query returns the decoding display position of the specified BUS, expressed in scientific notation.

- **For example:**

  :BUS1:POSition 2

  Set the decoding display position of BUS1 to 2div.

:BUS1:POSition ?

The query returns 2.00000E+000.

:BUS<n>:LABel

- **Command format:**

  :BUS<n>:LABel <lable>

  :BUS<n>:LABel ?

- **Function description:**

  Set or query the label of the specified BUS.

- **Parameter:**

  n: Integer, ranging from 1 to 2

  label: Discrete type

- **Return format:**

  The query returns the label of the specified BUS.

- **For example:**

  :BUS1:LABel abcd Set the label of BUS1 as abcd.

  :BUS1:LABel ? The query returns abcd.

:BUS<n>:DATA

- **Command format:**

  :BUS<n>:DATA ?

- **Function description:**

  Query the event list result of the specified BUS.

- **Parameter:**

  n: Integer, ranging from 1 to 2

- **For example:**

  The query returns all the result values of the specified BUS event list, conforming to the data block format, is expressed in scientific notation, and is arranged in sequence and separated by commas. Invalid values are represented by the maximum value of the real data. The return format is as follows, taking the CAN type as an example:

  For example: "#9000000272CAN,

  Index,Start Time,StandardID,ExpandID,DLC,Data,CRC,ACK,EOF,ERROR

  1,-1.89960E-004,NaN,00 02 73 E6,0F,NaN,NaN,NaN,NaN,NaN

2,2.10040E-004,NaN,NaN,NaN,NaN,NaN,NaN,NaN,NaN

3,3.10040E-004,01 8F,NaN,00,NaN,30 67,01,03,CRC

4,5.60040E-004,NaN,NaN,NaN,NaN,NaN,NaN,NaN,NaN"

RS232

:BUS<n>:RS232:SOURce

- **Command format:**

  :BUS<n>:RS232:SOURce <source>

  :BUS<n>:RS232:SOURce ?

- **Function description:**

  Set the RS232 source of BUS1 as C1.

- **Parameter:**

  n: Integer, ranging from 1 to 2

  source: Discrete type {C1 | C2 | C3 | C4}

- **Return format:**

  The query returns the decoding source of RS232.

- **For example:**

  :BUS1:RS232:SOURce C1 Set the RS232 source of BUS1 as C1.

  :BUS1:RS232:SOURce ? The query returns C1.

:BUS<n>:RS232:BAUDrate:CURRent

- **Command format:**

  :BUS<n>:RS232:BAUDrate:CURRent <value>

  :BUS<n>:RS232:BAUDrate ?

- **Function description:**

  Set or query the decoding baud rate of RS232. The default unit is bps.

- **Parameter:**

  n: Integer, ranging from 1 to 2

  value: Discrete type {CUSTOM | 2400 | 4800 | 9600 | 192000 | 38400 | 57600 | 115200}

- **Return format:**

  The query returns the decoding baud rate of RS232.

- **For example:**

:BUS1:RS232:BAUDrate:CURRent 115200

Set the RS232 baud rate of BUS1 to 115200bps.

:BUS1:RS232:BAUDrate:CURRent ?

The query returns 115200.

:BUS<n>:RS232:BAUdrate:CUSTom

- **Command format:**

  :BUS<n>:RS232:BAUdrate:CUSTom <value>

  :BUS<n>:RS232:BAUdrate:CUSTom ?

- **Function description:**

  Set or query the custom decoding baud rate of RS232. The default unit is bps.

- **Parameter:**

  n: Integer, ranging from 1 to 2

  value: Integer type, range from 1kbps to 1Gbps

- **Return format:**

  The query returns the custom decoding baud rate of RS232.

- **For example:**

  :BUS1:RS232:BAUdrate:CUSTom 5500

  Set the custom baud rate of RS232 of BUS1 to 5500bps.

  :BUS1:RS232:BAUdrate:CUSTom ?

  The query returns5.50000E+003.

:BUS<n>:RS232:LEVel

- **Command format:**

  :BUS<n>:RS232:LEVel <value>

  :BUS<n>:RS232:LEVel ?

- **Function description:**

  Set or query the decoding threshold of RS232. The default unit is V.

- **Parameter:**

  n: Integer, ranging from 1 to 2

  value: Real number type. The unit is consistent with the vertical scale unit of the channel. The default unit is V.

- **Return format:**

The query returns the decoding threshold of RS232.

- **For example:**

  :BUS1:RS232:LEVel 1

  Set the decoding threshold of RS232 of BUS1 to 1V.

  :BUS1:RS232:LEVel ?

  The query returns 1.00000E+000.

:BUS<n>:RS232:ORDer

- **Command format:**

  :BUS<n>:RS232:ORDer <order>

  :BUS<n>:RS232:ORDer ?

- **Function description:**

  Set or query the bit order for RS232 decoding.

- **Parameter:**

  n: Integer, ranging from 1 to 2

  order: Discrete type {MSB | LSB}

  MSB:Most Significant Bit

  LSB:Least Significant Bit

- **Return format:**

  The query returns the decoding bit order of RS232.

- **For example:**

  :BUS1:RS232:ORDer LSB

  Set the bit order for RS232 decoding on BUS1 to LSB.

  :BUS1:RS232:ORDer ?

  The query returns LSB.

:BUS<n>:RS232:WIDTh

- **Command format:**

  :BUS<n>:RS232:WIDTh <width>

  :BUS<n>:RS232:WIDTh ?

- **Function description:**

  Set or query the bit width for RS232 decoding.

- **Parameter:**

n: Integer, ranging from 1 to 2

width: Discrete type {5 | 6 | 7 | 8}

- **Return format:**

  The query returns the decoding bit width of RS232.

- **For example:**

  :BUS1:RS232:WIDTh 8

  Set the bit width for RS232 decoding on BUS1 to 8bits.

  :BUS1:RS232:WIDTh ?

  The query returns 8.

:BUS<n>:RS232:STOP

- **Command format:**

  :BUS<n>:RS232:STOP <stop>

  :BUS<n>:RS232:STOP ?

- **Function description:**

  Set or query the decoding stop bit of RS232.

- **Parameter:**

  n: Integer, ranging from 1 to 2

  width: Discrete type {1 | 2}

- **Return format:**

  The query returns the decoding stop bit of RS232.

- **For example:**

  :BUS1:RS232:STOP 1

  Set the decoding stop bit of RS232 of BUS1 to 1.

  :BUS1:RS232:STOP ?

  The query returns 1.

:BUS<n>:RS232:PARity

- **Command format:**

  :BUS<n>:RS232:PARity <type>

  :BUS<n>:RS232:PARity ?

- **Function description:**

  Set or query the parity bit for RS232 decoding.

- **Parameter:**

  n: Integer, ranging from 1 to 2

  type: Discrete type {NONE | ODD | EVEN}

  NONe: No parity

  ODD: Odd parity

  EVEN: Even Parity

- **Return format:**

  The query returns the decoding parity bit of RS232.

- **For example:**

  :BUS1:RS232:PARity ODD

  Set the parity bit for RS232 decoding on BUS1 to ODD.

  :BUS1:RS232:PARity ?

  The query returns ODD.

:BUS<n>:RS232:POLarity

- **Command format:**

  :BUS<n>:RS232:POLarity <type>

  :BUS<n>:RS232:POLarity ?

- **Function description:**

  Set or query the polarity for RS232 decoding.

- **Parameter:**

  n: Integer, ranging from 1 to 2

  type: Discrete type {NEGATIVE | POSITIVE}

  NEGative: Negative polarity

  POSitive: Positive polarity

- **Return format:**

  The query returns the decoding polarity of RS232.

- **For example:**

  :BUS1:RS232:POLarity POSitive

  Set the polarity for RS232 decoding on BUS1 to POSitive.

  :BUS1:RS232:POLarity ?

  The query returns POSitive.

I2C

:BUS<n>:I2C:SDA

- **Command format:**

  :BUS<n>:I2C:SDA <source>

  :BUS<n>:I2C:SDA ?

- **Function description:**

  Set or query the data source for I2C decoding.

- **Parameter:**

  n: Integer, ranging from 1 to 2

  source: Discrete type {C1 | C2 | C3 | C4}

- **Return format:**

  The query returns the data source of I2C decoding.

- **For example:**

  :BUS1:I2C:SDA C1

  Set the data source for I2C decoding on BUS1 to C1.

  :BUS1:I2C:SDA ?

  The query returns C1.

:BUS<n>:I2C:SCL

- **Command format:**

  :BUS<n>:I2C:SCL <source>

  :BUS<n>:I2C:SCL ?

- **Function description:**

  Set or query the clock source for I2C decoding.

- **Parameter:**

  n: Integer, ranging from 1 to 2

  source: Discrete type {C1 | C2 | C3 | C4}

- **Return format:**

  The query returns the clock source of I2C decoding.

- **For example:**

  :BUS1:I2C:SCL C2

Set the clock source for I2C decoding on BUS1 to C2.

:BUS1:I2C:SCL ?

The query returns C2.

:BUS<n>:I2C:DLEVel

- **Command format:**

  :BUS<n>:I2C:DLEVel <level>

  :BUS<n>:I2C:DLEVel ?

- **Function description:**

  Set or query the data threshold of I2C decoding. The default unit is V.

- **Parameter:**

  n: Integer, ranging from 1 to 2

  level: Real number type. The unit is consistent with the vertical scale unit of the channel. The default unit is V. The range is -8V to 8V.

- **Return format:**

  The query returns the data threshold of I2C decoding.0

- **For example:**

  :BUS1:I2C:DLEVel 1

  Set the data threshold of I2C decoding of BUS1 to 1V.

  :BUS1:I2C:DLEVel ?

  The query returns 1.00000E+000.

:BUS<n>:I2C:CLEVel

- **Command format:**

  :BUS<n>:I2C:CLEVel <level>

  :BUS<n>:I2C:CLEVel ?

- **Function description:**

  Set or query the clock threshold for I2C decoding. The default unit is V.

- **Parameter:**

  n: Integer, ranging from 1 to 2

  level: Real number type. The unit is consistent with the vertical scale unit of the channel. The default unit is V. The range is -8V to 8V.

- **Return format:**

The query returns the clock threshold of I2C decoding.

- **For example:**

  :BUS1:I2C:CLEVel 1

  Set the clock threshold for I2C decoding on BUS1 to 1V.

  :BUS1:I2C:CLEVel ?

  The query returns 1.00000E+000.

:BUS<n>:I2C:WIDth

- **Command format:**

  :BUS<n>:I2C:WIDTh <width>

  :BUS<n>:I2C:WIDTh ?

- **Function description:**

  Set or query the bit width for I2C decoding.

- **Parameter:**

  n: Integer, ranging from 1 to 2

  width: Discrete type {7 | 10}

- **Return format:**

  The query returns the bit width of I2C decoding.

- **For example:**

  :BUS1:I2C:WIDTh 7

  Set the bit width for I2C decoding on BUS1 to 7 bits.

  :BUS1:I2C:WIDTh ?

  The query returns 7.

SPI

:BUS<n>:SPI:MOD

- **Command format:**

  :BUS<n>:SPI:MOD <type>

  :BUS<n>:SPI:MOD ?

- **Function description:**

  Set or query SPI decoding mode。

- **Parameter:**

n:integer，range1~2

type:Discrete type{TIMEout | CS }

TIMEout:TIMEOUT

CS:CS

- **Return format:**

  The query returns SPI decoding mode。

- **For example:**

  :BUS1:SPI:MOD CS　　Set the SPI decoding mode of BUS1 to CS

  :BUS1:SPI:MOD ?　　The query returnsCS

:BUS<n>:SPI:TIMEout

- **Command format:**

  :BUS<n>:SPI:TIMEout <value>

  :BUS<n>:SPI:TIMEout ?

- **Function description:**

  Set or query the timeout time for SPI decoding mode。

- **Parameter:**

  n:integer，range1~2

  value:real，range：50ns-10s。

- **Return format:**

  The query returns the timeout of SPI decoding mode TIMEout。

- **For example:**

  :BUS1:SPI:TIMEout 1　　Set the decoding mode timeout to 1 second

  :BUS1:SPI:TIMEout ?　　The query returns 1.0000000000000E+000

:BUS<n>:SPI:SOURce:SCL

- **Command format:**

  :BUS<n>:SPI:SOURce:SCL <source>

  :BUS<n>:SPI:SOURce:SCL ?

- **Function description:**

  Set or query SPI decoding clock source。

- **Parameter:**

  n:integer，range1~2

source:Discrete type{C1 | C2 | C3 | C4}

- **Return format:**

  The query returns SPI decoding clock source。

- **For example:**

  :BUS1:SPI:SOURce:SCL C2 Set the SPI decoding clock source C2 for BUS1

  :BUS1:SPI:SOURce:SCL ? The query returnsC2

:BUS<n>:SPI:SOURce:BIT

- **Command format:**

  :BUS<n>:SPI:SOURce:BIT <source>

  :BUS<n>:SPI:SOURce:BIT ?

- **Function description:**

  Set or query the word selection source for SPI decoding。

- **Parameter:**

  n:integer，range1~2

  source:Discrete type{C1 | C2 | C3 | C4}

- **Return format:**

  The query returns the word selection source for SPI decoding。

- **For example:**

  :BUS1:SPI:SOURce:BIT C3 Set the SPI decoding word selection source C3 for BUS1

  :BUS1:SPI:SOURce:BIT ? The query returnsC3

:BUS<n>:SPI:SOURce:DAT

- **Command format:**

  :BUS<n>:SPI:SOURce:DAT <source>

  :BUS<n>:SPI:SOURce:DAT ?

- **Function description:**

  Set or query the data source for SPI decoding。

- **Parameter:**

  n:integer，range1~2

  source:Discrete type{C1 | C2 | C3 | C4}

- **Return format:**

The query returns the data source for SPI decoding。

- **For example:**

:BUS1:SPI:SOURce:DAT C3 Set the SPI decoding data source C3 for BUS1

:BUS1:SPI:SOURce:DAT ? The query returnsC3

:BUS<n>:SPI:LEVel:SCL

- **Command format:**

:BUS<n>:SPI:LEVel:SCL <level>

:BUS<n>:SPI:LEVel:SCL ?

- **Function description:**

Set or query the clock threshold for SPI decoding, default unit V。

- **Parameter:**

n:integer，range1~2

level:real，The unit is consistent with the vertical scale unit of the channel, and the default unit is V，range-8V~8V

- **Return format:**

The query returns the clock threshold for SPI decoding。

- **For example:**

:BUS1:SPI:LEVel:SCL 1 Set the SPI decoding clock threshold of BUS1 to 1V

:BUS1:SPI:LEVel:SCL ? The query returns1.00000E+000

:BUS<n>:SPI:LEVel:BIT

- **Command format:**

:BUS<n>:SPI:LEVel:BIT <level>

:BUS<n>:SPI:LEVel:BIT ?

- **Function description:**

Set or query the word selection threshold for SPI decoding, default unit V。

- **Parameter:**

n:integer，range1~2

level:real，The unit is consistent with the vertical scale unit of the channel, and the default unit is V，range-8V~8V

- **Return format:**

The query returns the word selection threshold for SPI decoding。

- **For example:**

  :BUS1:SPI:LEVel:BIT 1 Set the SPI decoding word selection threshold for BUS1 to 1V

  :BUS1:SPI:LEVel:BIT ? The query returns1.00000E+000

:BUS<n>:SPI:LEVel:DATa

- **Command format:**

  :BUS<n>:SPI:LEVel:DATa <level>

  :BUS<n>:SPI:LEVel:DATa ?

- **Function description:**

  Set or query the data threshold for SPI decoding, default unit V。

- **Parameter:**

  n:integer，range1~2

  level:real，The unit is consistent with the vertical scale unit of the channel, and the default unit is V，range-8V~8V

- **Return format:**

  The query returns the data threshold for SPI decoding。

- **For example:**

  :BUS1:SPI:LEVel:DATa 1 Set the SPI decoding MOSI threshold of BUS1 to 1V

  :BUS1:SPI:LEVel:DATa ? The query returns1.00000E+000

:BUS<n>:SPI:EDGe:SCL

- **Command format:**

  :BUS<n>:SPI:EDGe:SCL <type>

  :BUS<n>:SPI:EDGe:SCL ?

- **Function description:**

  Set or query the clock edge for SPI decoding。

- **Parameter:**

  n:integer，range1~2

  type:Discrete type{FALL | RISe}

  FALL:Descending edge

  RISe:Rising edge

- **Return format:**

The query returns the clock edge of SPI decoding。

- **For example:**

:BUS1:SPI:EDGe:SCL RISe Set the SPI decoding clock edge of BUS1 to RISe

:BUS1:SPI:EDGe:SCL ? The query returnsRISe

:BUS<n>:SPI:EDGe:BIT

- **Command format:**

:BUS<n>:SPI:EDGe:BIT <type>

:BUS<n>:SPI:EDGe:BIT ?

- **Function description:**

Set or query the polarity of word selection for SPI decoding。

- **Parameter:**

n:integer，range1~2

type:Discrete type{LOW | HIGH}

LOW:Low Level

HIGH:high level

- **Return format:**

The query returns the polarity of word selection for SPI decoding。

- **For example:**

:BUS1:SPI:EDGe:BIT HIGH The query returns the polarity of word selection for SPI decoding

:BUS1:SPI:EDGe:BIT ? The query returnsHIGH

:BUS<n>:SPI:POLarity:DATa

- **Command format:**

:BUS<n>:SPI:POLarity:DATa <type>

:BUS<n>:SPI:POLarity:DATa ?

- **Function description:**

Set or query the polarity of SPI decoded data。

- **Parameter:**

n:integer，range1~2

type:Discrete type{NEGative | POSitive}

NEGative:negative polarity

POSitive:positive polarity

- **Return format:**

  The query returns data polarity decoded by SPI。

- **For example:**

  :BUS1:SPI:POLarity:DATa POSitive　　The query returns data polarity decoded by SPI

  :BUS1:SPI:POLarity:DATa ?　　The query returnsPOSitive

:BUS<n>:SPI:FRAMeCount

- **Command format:**

  :BUS<n>:SPI:FRAMeCount <value>

  :BUS<n>:SPI:FRAMeCount ?

- **Function description:**

  Set or query the data bit width of SPI。

- **Parameter:**

  n:integer，range1~2

  value:real4-32

- **Return format:**

  The data bit width of the query returnsSPI。

- **For example:**

  :BUS1:SPI:FRAMeCount 8　　Set the RS232 decoding bit width of BUS1 to 8

  :BUS1:SPI:FRAMeCount ?　　The query returns8

:BUS<n>:SPI:ORDer

- **Command format:**

  :BUS<n>:SPI:ORDer <type>

  :BUS<n>:SPI:ORDer ?

- **Function description:**

  Set or query the bit order of SPI decoding。

- **Parameter:**

  n:integer，range1~2

  type:Discrete type{MSB | LSB}

  MSB:Most Significant Bit

  LSB:Least Significant Bit

- **Return format:**

  The query returns the bit order of SPI decoding。

- **For example:**

  :BUS1:SPI:ORDer LSB Set the SPI decoding bit order of BUS1 to LSB

  :BUS1:SPI:ORDer ? The query returnsLSB

CAN

:BUS<n>:CAN:SOURce

- **Command format:**

  :BUS<n>:CAN:SOURce <source>

  :BUS<n>:CAN:SOURce ?

- **Function description:**

  Set or query the CAN decoding source.

- **Parameter:**

  n: Integer, ranging from 1 to 2

  source: Discrete type {C1 | C2 | C3 | C4}

- **Return format:**

  The query returns the CAN decoding source.

- **For example:**

  :BUS1:CAN:SOURce C1

  Set the CAN decoding source of BUS1 to C1.

  :BUS1:CAN:SOURce ?

  The query returns C1.

:BUS<n>:CAN:STYPe

- **Command format:**

  :BUS<n>:CAN:STYPe <type>

  :BUS<n>:CAN:STYPe ?

- **Function description:**

  Set or query the signal type for CAN decoding.

- **Parameter:**

  n: Integer, ranging from 1 to 2

type:Discrete type {L | H}

L: CAN_L

H: CAN_H

- **Return format:**

  The query returns the CAN decoding signal type.

- **For example:**

  :BUS1:CAN:STYPe H

  Set the CAN decoding signal type of BUS1 to CAN_H.

  :BUS1:CAN:STYPe ?

  The query returns H.

:BUS<n>:CAN:LEVel

- **Command format:**

  :BUS<n>:CAN:LEVel <level>

  :BUS<n>:CAN:LEVel ?

- **Function description:**

  Set or query the threshold for CAN decoding. The default unit is V.

- **Parameter:**

  n: Integer, ranging from 1 to 2

  level: Real type, the unit is consistent with the vertical scale unit of the channel. The default unit is V, and the range is -12V to 12V.

- **Return format:**

  The query returns the threshold for CAN decoding.

- **For example:**

  :BUS1:CAN:LEVel 1

  Set the CAN decoding threshold of BUS1 to 1V.

  :BUS1:CAN:LEVel ?

  The query returns 1.00000E+000.

:BUS<n>:CAN:BAUDrate:CURRent

- **Command format:**

  :BUS<n>:CAN:BAUDrate:CURRent <value>

  :BUS<n>:CAN:BAUDrate:CURRent ?

- **Function description:**

  Set or query the signal rate for CAN decoding.

- **Parameter:**

  n: Integer, ranging from 1 to 2

  value: Discrete type {CUSTom | 10000 | 19200 | 20000 | 33300 | 38400 | 50000 | 57600 | 62500 | 83300 | 100000 | 115200 | 125000 | 230400 | 250000 | 490800 | 500000 | 800000 | 921600 | 1000000 | 2000000 | 3000000 | 4000000 | 5000000}

- **Return format:**

  The query returns the signal rate for CAN decoding.

- **For example:**

  :BUS1:CAN:BAUDrate:CURRent 10000

  Set the CAN decoding signal rate of BUS1 to 10k.

  :BUS1:CAN:BAUDrate:CURRent ?

  The query returns 10000.

:BUS<n>:CAN:BAUDrate:CUSTom

- **Command format:**

  :BUS<n>:CAN:BAUDrate:CUSTom <value>

  :BUS<n>:CAN:BAUDrate:CUSTom ?

- **Function description:**

  Set or query the CAN decoding custom signal rate. The default unit is bps.

- **Parameter:**

  n: Integer, ranging from 1 to 2

  value: Integer type, range 10kbps - 10Mbps

- **Return format:**

  The query returns the CAN decoding custom signal rate.

- **For example:**

  :BUS1:CAN:BAUDrate:CUSTom 10000

  Set the CAN decoding custom signal rate of BUS1 to 10kbps.

  :BUS1:CAN:BAUDrate:CUSTom ?

  The query returns1.00000E+004.

:BUS<n>:CAN:SPOint

- **Command format:**

  :BUS<n>:CAN:SPOint <value>

  :BUS<n>:CAN:SPOint ?

- **Function description:**

  Set or query the CAN decoding sampling point. The default unit is %.

- **Parameter:**

  n: Integer, ranging from 1 to 2

  value: Integer type, range 30% - 90%

- **Return format:**

  The query returns the CAN sampling point.

- **For example:**

  :BUS1:CAN:SPOint 30

  Set the CAN decoding sampling point of BUS1 to 30%.

  :BUS1:CAN:SPOint ?

  The query returns 30.

CAN-FD

:BUS<n>:CAN_FD:SOURce

- **Command format:**

  :BUS<n>:CAN_FD:SOURce <source>

  :BUS<n>:CAN_FD:SOURce ?

- **Function description:**

  Set or query the source for CAN_FD decoding.

- **Parameter:**

  n: Integer, ranging from 1 to 2

  source: Discrete type {C1 | C2 | C3 | C4}

- **Return format:**

  The query returns the source for CAN_FD decoding.

- **For example:**

  :BUS1:CAN_FD:SOURce C1

Set the CAN_FD decoding source of BUS1 to C1.

:BUS1:CAN_FD:SOURce ?

The query returns C1.

:BUS<n>:CAN_FD:STYPe

- **Command format:**

  :BUS<n>:CAN_FD:STYPe <type>

  :BUS<n>:CAN_FD:STYPe ?

- **Function description:**

  Set or query the CAN_FD decoding signal type.

- **Parameter:**

  n: Integer, ranging from 1 to 2

  type: Discrete type {L | H}

  L: CAN-FD_L

  H: CAN-FD_H

- **Return format:**

  The query returns the CAN_FD decoding signal type.

- **For example:**

  :BUS1:CAN_FD:STYPe H

  Set the CAN_FD decoding signal type of BUS1 to CAN_H.

  :BUS1:CAN_FD:STYPe ?

  The query returns H.

:BUS<n>:CAN_FD:LEVel

- **Command format:**

  :BUS<n>:CAN_FD:LEVel <level>

  :BUS<n>:CAN_FD:LEVel ?

- **Function description:**

  Set or query the threshold for CAN_FD decoding. The default unit is V.

- **Parameter:**

  n: Integer, ranging from 1 to 2

  level: Real number type. The unit is consistent with the vertical scale unit of the channel. The default unit is V. The range is -12V to 12V.

- **Return format:**

  The query returns the threshold for CAN_FD decoding.

- **For example:**

  :BUS1:CAN_FD:LEVel 1

  Set the CAN_FD decoding threshold of BUS1 to 1V.

  :BUS1:CAN_FD:LEVel ?

  The query returns 1.00000E+000.

:BUS<n>:CAN_FD:BAUDrate:SDCURrent

- **Command format:**

  :BUS<n>:CAN_FD:BAUDrate:SDCURrent <value>

  :BUS<n>:CAN_FD:BAUDrate:SDCURrent ?

- **Function description:**

  Set or query the SD signal rate for CAN_FD decoding.

- **Parameter:**

  n: Integer, ranging from 1 to 2

  value: Discrete type {CUSTom | 10000 | 19200 | 20000 | 33300 | 38400 | 50000 | 57600 | 62500 | 83300 | 100000 | 115200 | 125000 | 230400 | 250000 | 490800 | 500000 | 800000 | 921600 | 1000000 | 2000000 | 3000000 | 4000000 | 5000000}

- **Return format:**

  The query returns the SD signal rate for CAN_FD decoding.

- **For example:**

  :BUS1:CAN_FD:BAUDrate:SDCURrent 10000

  Set the CAN_FD decoding SD signal rate of BUS1 to 10000.

  :BUS1:CAN_FD:BAUDrate:SDCURrent ?

  The query returns 10000.

:BUS<n>:CAN_FD:BAUDrate:SDCUStom

- **Command format:**

  :BUS<n>:CAN_FD:BAUDrate:SDCUStom <value>

  :BUS<n>:CAN_FD:BAUDrate:SDCUStom ?

- **Function description:**

  Set or query the CAN_FD decoding SD custom signal rate. The default unit is bps.

- **Parameter:**

  n: Integer, ranging from 1 to 2

  value: Integer type. The range is from 10kbps to 1Mbps.

- **Return format:**

  The query returns the CAN_FD decoding SD custom signal rate.

- **For example:**

  :BUS1:CAN_FD:BAUDrate:SDCUStom 10000

  Set the CAN_FD decoding SD custom signal rate to 10k.

  :BUS1:CAN_FD:BAUDrate:SDCUStom ?

  The query returns1.00000E+004.

:BUS<n>:CAN_FD:BAUDrate:FDCURrent

- **Command format:**

  :BUS<n>:CAN_FD:BAUDrate:FDCURrent <value>

  :BUS<n>:CAN_FD:BAUDrate:FDCURrent ?

- **Function description:**

  Set or query the FD signal rate for CAN_FD decoding.

- **Parameter:**

  n: Integer, ranging from 1 to 2

  value: Discrete type{CUSTom | 1000000 | 2000000 | 3000000 | 4000000 | 5000000 | 6000000 | 7000000 | 8000000}

- **Return format:**

  The query returns the FD signal rate for CAN_FD decoding.

- **For example:**

  :BUS1:CAN_FD:BAUDrate:FDCURrent 1000000

  Set the CAN_FD decoding FD signal rate to 1M.

  :BUS1:CAN_FD:BAUDrate:FDCURrent ?

  The query returns 1000000.

:BUS<n>:CAN_FD:BAUDrate:FDCUStom

- **Command format:**

  :BUS<n>:CAN_FD:BAUDrate:FDCustom <value>

  :BUS<n>:CAN_FD:BAUDrate:FDCustom ?

- **Function description:**

  Set or query the CAN_FD decoding FD custom signal rate. The default unit is bps.

- **Parameter:**

  n: Integer, ranging from 1 to 2

  value:Integer type. The range is from 500 kbps to 16 Mbps.

- **Return format:**

  The query returns the CAN_FD decoding FD custom signal rate.

- **For example:**

  :BUS1:CAN_FD:BAUDrate:FDCustom 1000000

  Set the CAN_FD decoding FD custom signal rate to 1M.

  :BUS1:CAN_FD:BAUDrate:FDCustom ?

  The query returns 1.00000E+006.

:BUS<n>:CAN_FD:SPOint

- **Command format:**

  :BUS<n>:CAN_FD:SPOint <value>

  :BUS<n>:CAN_FD:SPOint ?

- **Function description:**

  Set or query the sampling point for CAN_FD decoding. The default unit is %.

- **Parameter:**

  n: Integer, ranging from 1 to 2

  value: Integer type. The range is from 30% to 90%.

- **Return format:**

  The query returns the sampling point for CAN_FD decoding.

- **For example:**

  :BUS1:CAN_FD:SPOint 30

  Set the CAN decoding sampling point of BUS1 to 30%.

  :BUS1:CAN_FD:SPOint ?

  The query returns 30.

:BUS<n>:CAN_FD:DPOint

- **Command format:**

  :BUS<n>:CAN_FD:DPOint <value>

:BUS<n>:CAN_FD:DPOint ?

- **Function description:**

  Set or query sampling points in the CAN_SD data domain, default unit%。

- **Parameter:**

  n:integer，range 1~2

  value:integer，range 30%~90%

- **Return format:**

  Query and return sampling points in the CAN_SD data domain。

- **For example:**

  :BUS1:CAN_FD:DPOint 30　　Set the CAN decoding sampling point of BUS1 to 30%

  :BUS1:CAN_FD:DPOint ?　　Query returns 30

LIN

:BUS<n>:LIN:SOURce

- **Command format:**

  :BUS<n>:LIN:SOURce <source>

  :BUS<n>:LIN:SOURce ?

- **Function description:**

  Set or query the source for LIN decoding.

- **Parameter:**

  n: Integer, ranging from 1 to 2

  source:Discrete type {C1 | C2 | C3 | C4}

- **Return format:**

  The query returns the source for LIN decoding.

- **For example:**

  :BUS1:LIN:SOURce C1

  Set the LIN decoding source of BUS1 to C1.

  :BUS1:LIN:SOURce ?

  The query returns C1.

:BUS<n>:LIN:LEVel

- **Command format:**

  :BUS<n>:LIN:LEVel <level>

  :BUS<n>:LIN:LEVel ?

- **Function description:**

  Set or query the LIN decoding threshold. The default unit is V.

- **Parameter:**

  n: Integer, ranging from 1 to 2

  level:Real number type. The unit is consistent with the vertical scale unit of the channel. The default unit is V. The range is from -12V to 12V.

- **Return format:**

  The query returns the LIN decoding threshold.

- **For example:**

  :BUS1:LIN:LEVel 1

  Set the LIN decoding threshold of BUS1 to 1V.

  :BUS1:LIN:LEVel ?

  The query returns 1.00000E+000.

:BUS<n>:LIN:POLarity

- **Command format:**

  :BUS<n>:LIN:POLarity <type>

  :BUS<n>:LIN:POLarity ?

- **Function description:**

  Set or query the LIN decoding polarity.

- **Parameter:**

  n: Integer, ranging from 1 to 2

  type: Discrete type {POSitive | NEGative}

  NEGative: Negative polarity

  POSitive: Positive polarity

- **Return format:**

  The query returns the LIN decoding polarity.

- **For example:**

:BUS1:LIN:POLarity POSitive

Set the LIN decoding polarity of BUS1 to POSitive.

:BUS1:LIN:POLarity ?

The query returns POSitive.

:BUS<n>:LIN:BAUDrate:CURRent

- **Command format:**

  :BUS<n>:LIN:BAUDrate:CURRent <value>

  :BUS<n>:LIN:BAUDrate:CURRent ?

- **Function description:**

  Set or query the baud rate for LIN decoding.

- **Parameter:**

  n: Integer, ranging from 1 to 2

  Value: Discrete type {CUSTom | 2400 | 4800 | 9600 | 19200}

- **Return format:**

  The query returns the baud rate for LIN decoding.

- **For example:**

  :BUS1:LIN:BAUDrate:CURRent 9600

  Set the LIN decoding baud rate of BUS1 to 9600.

  :BUS1:LIN:BAUDrate:CURRent ?

  The query returns 9600.

:BUS<n>:LIN:BAUDrate:CUSTom

- **Command format:**

  :BUS<n>:LIN:BAUDrate:CUSTom <value>

  :BUS<n>:LIN:BAUDrate:CUSTom ?

- **Function description:**

  Set or query the custom LIN decoding baud rate. The default unit is bps.

- **Parameter:**

  n: Integer, ranging from 1 to 2

  value:Integer type. The range is from 1kbps to 200Mbps.

- **Return format:**

  The query returns the custom LIN decoding baud rate.

- **For example:**

  :BUS1:LIN:BAUDrate:CUSTom 10000

  Set the custom LIN decoding baud rate of BUS1 to 10kbps.

  :BUS1:LIN:BAUDrate:CUSTom ?

  The query returns 1.00000E+004.

:BUS<n>:LIN:STANdard

- **Command format:**

  :BUS<n>:LIN:STANdard <type>

  :BUS<n>:LIN:STANdard ?

- **Function description:**

  Set or query the signal standard for LIN decoding.

- **Parameter:**

  n: Integer, ranging from 1 to 2

  type: Discrete type {V1 | V2 | Both}

  V1: LIN1.0

  V2: LIN2.0

  Both: Both

- **Return format:**

  The query returns the signal standard for LIN decoding.

- **For example:**

  :BUS1:LIN:STANdard V1

  Set the LIN decoding signal standard of BUS1 to LIN1.0.

  :BUS1:LIN:STANdard ?

  The query returns V1.

:BUS<n>:LIN:CKPID

- **Command format:**

  :BUS<n>:LIN:CKPID <type>

  :BUS<n>:LIN:CKPID ?

- **Function description:**

  Check whether the LIN decoding returned includes parity bits in the ID.

- **Parameter:**

n: Integer, ranging from 1 to 2

type: Discrete type {Y | N}

Y:include

N:not included

- **Return format:**

  Check whether the LIN decoding returned includes parity bits in the ID.

- **For example:**

  :BUS1:LIN:CKPID Y

  Set the PID verification check for LIN decoding of BUS1.

  :BUS1:LIN:CKPID ?

  The query returns 1.

:BUS<n>:LIN:SPOint

- **Command format:**

  :BUS<n>:LIN:SPOint <value>

  :BUS<n>:LIN:SPOint ?

- **Function description:**

  Set or query decoding sampling points。

- **Parameter:**

  n:integer，range 1~2

  value:integer range：50%-90%

- **Return format:**

  Query and return decoding sampling points。

- **For example:**

  :BUS1:LIN:SPOint 60　　Set the decoding sampling point to 60

  :BUS1:LIN:SPOint ?　　Query return60

FLEXRAY

:BUS<n>:FR:SOURce

- **Command format:**

  :BUS<n>:FR:SOURce <source>

:BUS<n>:FR:SOURce ?

- **Function description:**

  Set or query the source for FLEXRAY decoding.

- **Parameter:**

  n: Integer, ranging from 1 to 2

  source: Discrete type {C1 | C2 | C3 | C4}

- **Return format:**

  The query returns the source for FLEXRAY decoding.

- **For example:**

  :BUS1:FR:SOURce C1

  Set the source of FLEXRAY decoding of BUS1 as C1.

  :BUS1:FR:SOURce ?

  The query returns C1.

:BUS<n>:FR:STYPe

- **Command format:**

  :BUS<n>:FR:STYPe <type>

  :BUS<n>:FR:STYPe ?

- **Function description:**

  Set or query the type of the FLEXRAY decoding source.

- **Parameter:**

  n: Integer, ranging from 1 to 2

  type: Discrete type {BP | BM}

- **Return format:**

  The query returns the type of the FLEXRAY decoding source.

- **For example:**

  :BUS1:FR:STYPe BP

  Set the type of the FLEXRAY decoding source of BUS1 as BP.

  :BUS1:FR:STYPe ?

  The query returns BP.

:BUS<n>:FR:BAUDrate:CURRent

- **Command format:**

:BUS<n>:FR:BAUDrate:CURRent <value>

:BUS<n>:FR:BAUDrate:CURRent ?

- **Function description:**

  Set or query the baud rate of FLEXRAY decoding, with the default unit bps.

- **Parameter:**

  n: Integer, ranging from 1 to 2

  value: Discrete type {CUSTOM | 1000000 | 5000000 | 10000000}

- **Return format:**

  The query returns the baud rate of FLEXRAY decoding.

- **For example:**

  :BUS1:FR:BAUDrate:CURRent 1000000

  The query returns the baud rate of FLEXRAY decoding.

  :BUS1:FR:BAUDrate:CURRent ?

  The query returns 1000000.

:BUS<n>:FR:BAUDrate:CUSTom

- **Command format:**

  :BUS<n>:FR:BAUDrate:CUSTom <value>

  :BUS<n>:FR:BAUDrate:CUSTom ?

- **Function description:**

  Set or query the custom baud rate of FLEXRAY decoding, with the default unit bps.

- **Parameter:**

  n: Integer, ranging from 1 to 2

  Value: Integer type, range from 1Mbps to 10Mbps

- **Return format:**

  The query returns the custom baud rate of FLEXRAY decoding.

- **For example:**

  :BUS1:FR:BAUDrate:CUSTom 1000000

  Set the custom baud rate of FLEXRAY decoding of BUS1 as 1Mbps.

  :BUS1:FR:BAUDrate:CUSTom ?

  The query returns 1.00000E+006.

:BUS<n>:FR:LEVel

- **Command format:**

  :BUS<n>:FR:LEVel <level>

  :BUS<n>:FR:LEVel ?

- **Function description:**

  Set or query the threshold of FLEXRAY decoding, with the default unit V.

- **Parameter:**

  n: Integer, ranging from 1 to 2

  Level: Real number type. The unit is consistent with the vertical scale unit of the channel. The default unit is V, and the range is from -12V to 12V.

- **Return format:**

  The query returns the threshold of FLEXRAY decoding.

- **For example:**

  :BUS1:FR:LEVel 1

  Set the threshold of FLEXRAY decoding of BUS1 as 1V.

  :BUS1:FR:LEVel ?

  The query returns 1.00000E+000.

:BUS<n>:FR:CHANnel

- **Command format:**

  :BUS<n>:FR:CHANnel <type>

  :BUS<n>:FR:CHANnel ?

- **Function description:**

  Set or query the channel type of FLEXRAY decoding.

- **Parameter:**

  n: Integer, ranging from 1 to 2

  type: Discrete type {A | B}

- **Return format:**

  The query returns the channel type of FLEXRAY decoding.

- **For example:**

  :BUS1:FR:CHANnel A

  Set the channel type of FLEXRAY decoding of BUS1 as A.

:BUS1:FR:CHANnel ?

The query returns A.

AUDIOBUS

:BUS<n>:AB:TYPe

- **Command format:**

  :BUS<n>:AB:TYPe <type>

  :BUS<n>:AB:TYPe ?

- **Function description:**

  Set or query the decoding protocol type of AUDIOBUS.

- **Parameter:**

  n: Integer, ranging from 1 to 2

  type:Discrete type {I2S | LJ | RJ | TDM}

- **Return format:**

  The query returns the decoding protocol type of AUDIOBUS.

- **For example:**

  :BUS1:AB:TYPe I2S

  Set the decoding protocol type of AUDIOBUS of BUS1 as I2S.

  :BUS1:AB:TYPe ?

  The query returns I2S.

:BUS<n>:AB:DPOLarity

- **Command format:**

  :BUS<n>:AB:DPOLarity <type>

  :BUS<n>:AB:DPOLarity ?

- **Function description:**

  Set or query the data polarity of AUDIOBUS decoding.

- **Parameter:**

  n: Integer, ranging from 1 to 2

  type: Discrete type {POSitive | NEGative}

  POSitive: Positive polarity

  NEGative: Negative polarity

- **Return format:**

  The query returns the data polarity of AUDIOBUS decoding.

- **For example:**

  :BUS1:AB:DPOLarity NEGative

  Set the data polarity of AUDIOBUS decoding of BUS1 as NEGative.

  :BUS1:AB:DPOLarity ?

  The query returns NEGative.

:BUS<n>:AB:SPOLarity

- **Command format:**

  :BUS<n>:AB:SPOLarity <type>

  :BUS<n>:AB:SPOLarity ?

- **Function description:**

  Set or query the synchronization polarity of AUDIOBUS decoding.

- **Parameter:**

  n: Integer, ranging from 1 to 2

  type: Discrete type {POSitive | NEGative}

  POSitive: Positive polarity

  NEGative: Negative polarity

- **Return format:**

  The query returns the synchronization polarity of AUDIOBUS decoding.

- **For example:**

  :BUS1:AB:SPOLarity NEGative

  Set the synchronization polarity of AUDIOBUS decoding of BUS1 as NEGative.

  :BUS1:AB:SPOLarity ?

  The query returns NEGative.

:BUS<n>:AB:CEDGe

- **Command format:**

  :BUS<n>:AB:CEDGe <type>

  :BUS<n>:AB:CEDGe?

- **Function description:**

  Set or query the valid clock edge of AUDIOBUS decoding.

- **Parameter:**

  n: Integer, ranging from 1 to 2

  type: Discrete type {RISe | FALL}

  RISe: Rising edge

  FALL: Falling edge

- **Return format:**

  The query returns the valid clock edge of AUDIOBUS decoding.

- **For example:**

  :BUS1:AB:CEDGe FALL

  Set the valid clock edge of AUDIOBUS decoding of BUS1 as FALL.

  :BUS1:AB:CEDGe ?

  The query returns FALL.

:BUS<n>:AB:ORDer

- **Command format:**

  :BUS<n>:AB:ORDer <type>

  :BUS<n>:AB:ORDer ?

- **Function description:**

  Set or query the bit sequence of AUDIOBUS decoding.

- **Parameter:**

  n: Integer, ranging from 1 to 2

  type: Discrete type {MSB | LSB}

- **Return format:**

  The query returns the bit sequence of AUDIOBUS decoding.

- **For example:**

  :BUS1:AB:ORDer LSB

  Set the bit sequence of AUDIOBUS decoding of BUS1 as LSB.

  :BUS1:AB:ORDer ?

  The query returns LSB.

:BUS<n>:AB:SCTYpe

- **Command format:**

  :BUS<n>:AB:SCTYpe <type>

:BUS<n>:AB:SCTYpe ?

- **Function description:**

  Set or query the audio channel type of AUDIOBUS decoding.

- **Parameter:**

  n: Integer, ranging from 1 to 2

  type: Discrete type {LR | L | R}

  LR: Left/Right channel

  L: Left channel

  R: Right channel

- **Return format:**

  The query returns the audio channel type of AUDIOBUS decoding.

- **For example:**

  :BUS1:AB:SCTYpe R

  Set the audio channel type of AUDIOBUS decoding of BUS1 as Right channel.

  :BUS1:AB:SCTYpe ?

  The query returns R.

:BUS<n>:AB:SCCount

- **Command format:**

  :BUS<n>:AB:SCCount <value>

  :BUS<n>:AB:SCCount ?

- **Function description:**

  Set or query the number of audio channels of AUDIOBUS decoding.

- **Parameter:**

  n: Integer, ranging from 1 to 2

  value: Integer type, range 2 - 32

- **Return format:**

  The query returns the number of audio channels of AUDIOBUS decoding.

- **For example:**

  :BUS1:AB:SCCount 2

  Set the number of audio channels of AUDIOBUS decoding of BUS1 as 2.

  :BUS1:AB:SCCount ?

  The query returns 2.

:BUS<n>:AB:DBCount

- **Command format:**

  :BUS<n>:AB:DBCount <value>

  :BUS<n>:AB:DBCount ?

- **Function description:**

  Set or query the number of data bits per audio channel of AUDIOBUS decoding.

- **Parameter:**

  n: Integer, ranging from 1 to 2

  value: Integer type, range 0 - 64

- **Return format:**

  The query returns the number of data bits per audio channel of AUDIOBUS decoding.

- **For example:**

  :BUS1:AB:DBCount 8

  Set the number of data bits per audio channel of AUDIOBUS decoding of BUS1 as 8.

  :BUS1:AB:DBCount ?

  The query returns 8.

:BUS<n>:AB:CBCount

- **Command format:**

  :BUS<n>:AB:CBCount <value>

  :BUS<n>:AB:CBCount ?

- **Function description:**

  Set or query the number of clock bits per audio channel for AUDIOBUS decoding.

- **Parameter:**

  n: Integer, ranging from 1 to 2

  value: Integer type, range 0 - 64

- **Return format:**

  The query returns the number of clock bits per audio channel for AUDIOBUS decoding.

- **For example:**

  :BUS1:AB:CBCount 8

  Set the number of clock bits per audio channel of AUDIOBUS decoding of BUS1 as 8.

  :BUS1:AB:CBCount ?

The query returns 8.

:BUS<n>:AB:SOURce:SCL

- **Command format:**

  :BUS<n>:AB:SOURce:SCL <source>

  :BUS<n>:AB:SOURce:SCL ?

- **Function description:**

  Set or query the clock source of AUDIOBUS decoding.

- **Parameter:**

  n: Integer, ranging from 1 to 2

  source: Discrete type {C1 | C2 | C3 | C4}

- **Return format:**

  The query returns the clock source of AUDIOBUS decoding.

- **For example:**

  :BUS1:AB:SOURce:SCL C4

  Set the clock source of AUDIOBUS decoding of BUS1 as C4.

  :BUS1:AB:SOURce:SCL ?

  The query returns C4.

:BUS<n>:AB:SOURce:CS

- **Command format:**

  :BUS<n>:AB:SOURce:CS <source>

  :BUS<n>:AB:SOURce:CS ?

- **Function description:**

  Set or query the chip select source of AUDIOBUS decoding.

- **Parameter:**

  n: Integer, ranging from 1 to 2

  source: Discrete type {C1 | C2 | C3 | C4}

- **Return format:**

  The query returns the chip select source of AUDIOBUS decoding.

- **For example:**

  :BUS1:AB:SOURce:CS C3

  Set the chip select source of AUDIOBUS decoding of BUS1 as C3.

:BUS1:AB:SOURce:CS ?

The query returns C3.

:BUS<n>:AB:SOURce:SDA

- **Command format:**

  :BUS<n>:AB:SOURce:SDA <source>

  :BUS<n>:AB:SOURce:SDA ?

- **Function description:**

  Set or query the data source of AUDIOBUS decoding.

- **Parameter:**

  n: Integer, ranging from 1 to 2

  source: Discrete type {C1 | C2 | C3 | C4}

- **Return format:**

  The query returns the data source of AUDIOBUS decoding.

- **For example:**

  :BUS1:AB:SOURce:SDA C2

  Set the data source of AUDIOBUS decoding of BUS1 as C2.

  :BUS1:AB:SOURce:SDA ?

  The query returns C2.

:BUS<n>:AB:LEVel:SCL

- **Command format:**

  :BUS<n>:AB:LEVel:SCL <level>

  :BUS<n>:AB:LEVel:SCL ?

- **Function description:**

  Set or query the clock threshold of AUDIOBUS decoding. The default unit is V.

- **Parameter:**

  n: Integer, ranging from 1 to 2

  level: Real number type. The unit is consistent with the unit of the vertical scale of the channel. The default unit is V. The range is -20V to 20V.

- **Return format:**

  The query returns the clock threshold of AUDIOBUS decoding.

- **For example:**

:BUS1:AB:LEVel:SCL 1

Set the clock threshold of AUDIOBUS decoding of BUS1 as 1V.

:BUS1:AB:LEVel:SCL ?

The query returns 1.00000E+000.

:BUS<n>:AB:LEVel:CS

- **Command format:**

  :BUS<n>:AB:LEVel:CS <level>

  :BUS<n>:AB:LEVel:CS ?

- **Function description:**

  Set or query the chip select threshold of AUDIOBUS decoding.

- **Parameter:**

  n: Integer, ranging from 1 to 2

  level: Real number type. The unit is consistent with the unit of the vertical scale of the channel. The default unit is V. The range is -20V to 20V.

- **Return format:**

  The query returns the chip select threshold of AUDIOBUS decoding.

- **For example:**

  :BUS1:AB:LEVel:CS 1

  Set the chip select threshold of AUDIOBUS decoding of BUS1 as 1V.

  :BUS1:AB:LEVel:CS ?

  The query returns 1.00000E+000.

:BUS<n>:AB:LEVel:SDA

- **Command format:**

  :BUS<n>:AB:LEVel:SDA <level>

  :BUS<n>:AB:LEVel:SDA ?

- **Function description:**

  Set or query the data threshold of AUDIOBUS decoding.

- **Parameter:**

  n: Integer, ranging from 1 to 2

  level: Real number type. The unit is consistent with the unit of the vertical scale of the channel. The default unit is V. The range is -20V to 20V.

- **Return format:**

  The query returns the data threshold of AUDIOBUS decoding.

- **For example:**

  :BUS1:AB:LEVel:SDA 1

  Set the data threshold of AUDIOBUS decoding of BUS1 as 1V.

  :BUS1:AB:LEVel:SDA ?

  The query returns 1.00000E+000.

:BUS<n>:AB:BITDelay

- **Command format:**

  :BUS<n>:AB:BITDelay <value>

  :BUS<n>:AB:BITDelay ?

- **Function description:**

  Set or query the bit delay of AUDIOBUS decoding.

- **Parameter:**

  n: Integer, ranging from 1 to 2

  value: Integer type. The range is 0 to 64.

- **Return format:**

  The query returns the bit delay of AUDIOBUS decoding.

- **For example:**

  :BUS1:AB:BITDelay 10

  Set the bit delay for AUDIOBUS decoding on BUS1 to 10.

  :BUS1:AB:BITDelay ?

  The query returns 10.

MIL_STD_1553

:BUS<n>:MS:SOURce

- **Command format:**

  :BUS<n>:MS:SOURce <source>

  :BUS<n>:MS:SOURce ?

- **Function description:**

  Set or query the source of MIL_STD_1553 decoding.

- **Parameter:**

  n: Integer, ranging from 1 to 2

  source: Discrete type {C1 | C2 | C3 | C4}

- **Return format:**

  The query returns the source of MIL_STD_1553 decoding.

- **For example:**

  :BUS1:MS:SOURce C4

  Set the decoding source for MIL_STD_1553 on BUS1 to C4.

  :BUS1:MS:SOURce ?

  The query returns C4.

:BUS<n>:MS:BAUDrate:CURRent

- **Command format:**

  :BUS<n>:MS:BAUDrate:CURRent <value>

  :BUS<n>:MS:BAUDrate:CURRent ?

- **Function description:**

  Set or query the baud rate for MIL_STD_1553 decoding. The default unit is bps.

- **Parameter:**

  n: Integer, ranging from 1 to 2

  value: Discrete type {CUSTom | 1000000 | 10000000}

- **Return format:**

  The query returns the baud rate of MIL_STD_1553 decoding.

- **For example:**

  :BUS1:MS:BAUDrate:CURRent 1000000

  Set the MIL_STD_1553 decoding baud rate on BUS1 to 1Mbps.

  :BUS1:MS:BAUDrate:CURRent ?

  The query returns 1000000.

:BUS<n>:MS:BAUDrate:CUSTom

- **Command format:**

  :BUS<n>:MS:BAUDrate:CUSTom <value>

  :BUS<n>:MS:BAUDrate:CUSTom ?

- **Function description:**

Set or query the baud rate for MIL_STD_1553 decoding. The default unit is bps.

- **Parameter:**

  n: Integer, ranging from 1 to 2

  value: Integer type. The range is 1kbps to 20Mbps.

- **Return format:**

  The query returns the baud rate for MIL_STD_1553 decoding.

- **For example:**

  :BUS1:MS:BAUDrate:CUSTom 1000000

  Set the MIL_STD_1553 decoding baud rate on BUS1 to 1Mbps.

  :BUS1:MS:BAUDrate:CUSTom ?

  The query returns 1.00000E+006.

:BUS<n>:MS:LEVel:HIGH

- **Command format:**

  :BUS<n>:MS:LEVel:HIGH <value>

  :BUS<n>:MS:LEVel:HIGH ?

- **Function description:**

  Set or query the high threshold for MIL_STD_1553 decoding. The default unit is V.

- **Parameter:**

  n: Integer, ranging from 1 to 2

  value: Real number type. The range is -30V to 30V.

- **Return format:**

  The query returns the high threshold for MIL_STD_1553 decoding.

- **For example:**

  :BUS1:MS:LEVel:HIGH 2

  Set the high threshold of MIL_STD_1553 decoding of BUS1 as 2V.

  :BUS1:MS:LEVel:HIGH ?

  The query returns 2.00000E+000.

:BUS<n>:MS:LEVel:LOW

- **Command format:**

  :BUS<n>:MS:LEVel:LOW <value>

  :BUS<n>:MS:LEVel:LOW ?

- **Function description:**

  Set or query the low threshold of MIL_STD_1553 decoding. The default unit is V.

- **Parameter:**

  n: Integer, ranging from 1 to 2

  value: Real number type. The range is -30V to 30V.

- **Return format:**

  The query returns the low threshold of MIL_STD_1553 decoding.

- **For example:**

  :BUS1:MS:LEVel:LOW 1

  Set the low threshold of MIL_STD_1553 decoding of BUS1 as 1V.

  :BUS1:MS:LEVel:LOW ?

  The query returns 1.00000E+000.

:BUS<n>:MS:POLarity

- **Command format:**

  :BUS<n>:MS:POLarity <polarity>

  :BUS<n>:MS:POLarity ?

- **Function description:**

  Set or query the decoding polarity for MIL_STD_1553.

- **Parameter:**

  n: Integer, ranging from 1 to 2

  polarity: Discrete type {POSitive | NEGative}

  POSitive: Positive polarity

  NEGative: Negative polarity

- **Return format:**

  The query returns the polarity of MIL_STD_1553 decoding.

- **For example:**

  :BUS1:MS:POLarity POSitive

  Set the decoding polarity of MIL_STD_1553 on BUS1 to POSitive.

  :BUS1:MS:POLarity ?

  The query returns POSitive.

ARINC429

:BUS<n>:A429:SOURce

- **Command format:**

  :BUS<n>:A429:SOURce <source>

  :BUS<n>:A429:SOURce ?

- **Function description:**

  Set or query the decoding source for ARINC429.

- **Parameter:**

  n: Integer, ranging from 1 to 2

  source: Discrete type {C1 | C2 | C3 | C4}

- **Return format:**

  The query returns the source of ARINC429 decoding.

- **For example:**

  :BUS1:A429:SOURce C4

  Set the decoding source of ARINC429 on BUS1 to C4.

  :BUS1:A429:SOURce ?

  The query returns C4.

:BUS<n>:A429:BAUDrate:CURRent

- **Command format:**

  :BUS<n>:A429:BAUDrate:CURRent <value>

  :BUS<n>:A429:BAUDrate:CURRent ?

- **Function description:**

  Set or query the baud rate for ARINC429 decoding. The default unit is bps.

- **Parameter:**

  n: Integer, ranging from 1 to 2

  value: Discrete type {CUSTOM | 12.5k | 100k}

- **Return format:**

  The query returns the baud rate of ARINC429 decoding.

- **For example:**

  :BUS1:A429:BAUDrate:CURRent 12.5k

Set the baud rate of ARINC429 decoding of BUS1 as 12.5kbps.

:BUS1:A429:BAUDrate:CURRent ?

The query returns 12.5k.

:BUS<n>:A429:BAUDrate:CUSTom

- **Command format:**

  :BUS<n>:A429:BAUDrate:CUSTom <value>

  :BUS<n>:A429:BAUDrate:CUSTom ?

- **Function description:**

  Set or query the baud rate for ARINC429 decoding. The default unit is bps.

- **Parameter:**

  n: Integer, ranging from 1 to 2

  value: Integer type, range 1kbps - 10Mbps

- **Return format:**

  The query returns the baud rate for ARINC429 decoding.

- **For example:**

  :BUS1:A429:BAUDrate:CUSTom 1000000

  Set the baud rate of ARINC429 decoding of BUS1 as 1Mbps.

  :BUS1:A429:BAUDrate:CUSTom ?

  The query returns 1.00000E+006.

:BUS<n>:A429:LEVel:HIGH

- **Command format:**

  :BUS<n>:A429:LEVel:HIGH <value>

  :BUS<n>:A429:LEVel:HIGH ?

- **Function description:**

  Set or query the high threshold for ARINC429 decoding. The default unit is V.

- **Parameter:**

  n: Integer, ranging from 1 to 2

  value: Real number type, range -20V to 20V

- **Return format:**

  The query returns the high threshold for ARINC429 decoding.

- **For example:**

:BUS1:A429:LEVel:HIGH 2

Set the high threshold of ARINC429 decoding of BUS1 as 2V.

:BUS1:A429:LEVel:HIGH ?

The query returns 2.00000E+000.

:BUS<n>:A429:LEVel:LOW

- **Command format:**

  :BUS<n>:A429:LEVel:LOW <value>

  :BUS<n>:A429:LEVel:LOW ?

- **Function description:**

  Set or query the low threshold for ARINC429 decoding. The default unit is V.

- **Parameter:**

  n: Integer, ranging from 1 to 2

  value: Real number type, range -20V to 20V

- **Return format:**

  The query returns the low threshold for ARINC429 decoding.

- **For example:**

  :BUS1:A429:LEVel:LOW 1

  Set the low threshold of ARINC429 decoding of BUS1 as 1V.

  :BUS1:A429:LEVel:LOW ?

  The query returns 1.00000E+000.

:BUS<n>:A429:POLarity

- **Command format:**

  :BUS<n>:A429:POLarity <polarity>

  :BUS<n>:A429:POLarity ?

- **Function description:**

  Set or query the decoding polarity for ARINC429.

- **Parameter:**

  n: Integer, ranging from 1 to 2

  polarity: Discrete type {POSitive | NEGative}

  POSitive: Positive polarity

  NEGative: Negative polarity

- **Return format:**

  The query returns the polarity of ARINC429 decoding.

- **For example:**

  :BUS1:A429:POLarity POSitive

  Set the decoding polarity for ARINC429 on BUS1 to POSitive.

  :BUS1:A429:POLarity ?

  The query returns POSitive.

:BUS<n>:A429:LEVel:MODe

- **Command format:**

  :BUS<n>:A429:MODe <type>

  :BUS<n>:A429:MODe ?

- **Function description:**

  Set or query the decoding mode of ARINC429.

- **Parameter:**

  n: Integer, ranging from 1 to 2

  type: Discrete type {19 | 21 | 23}

- **Return format:**

  The query returns the decoding mode of ARINC429.

- **For example:**

  :BUS1:A429:MODe 21

  Set the decoding mode of ARINC429 of BUS1 as 21BIT.

  :BUS1:A429:MODe ?

  The query returns 21.

SENT

:BUS<n>:SENT:SOURce

- **Command format:**

  :BUS<n>:SENT:SOURce <source>

  :BUS<n>:SENT:SOURce ?

- **Function description:**

  Set or query SENT decoding source。

- **Parameter:**

  n:integer，range1~2

  source:Discrete type{C1 | C2 | C3 | C4}

- **Return format:**

  The query returns SENT decoding source。

- **For example:**

  :BUS1:SENT:SOURce C2　　Set the SENT decoding source of BUS1 to C2

  :BUS1:SENT:SOURce ?　　The query returnsC2

:BUS<n>:SENT:CLOCKtick:CURRent

- **Command format:**

  :BUS<n>:SENT:CLOCLKtick:CURRent <value>

  :BUS<n>:SENT:CLOCKtick:CURRent ?

- **Function description:**

  Set or query SENT decoding clock cycle, default unit is us。

- **Parameter:**

  n:integer，range1~2

  value:Discrete type{CUSTom | 1 | 3 | 10 | 30 | 100 | 300}

- **Return format:**

  The query returns SET decoding clock cycle。

- **For example:**

  :BUS1:SENT:CLOCKtick:CURRent 10　　Set the SENT decoding clock cycle of BUS1 to 10us

  :BUS1:SENT:CLOCKtick:CURRent ?　　The query returns10

:BUS<n>:SENT:BAUDtick:CUSTom

- **Command format:**

  :BUS<n>:SENT:BAUDtick:CUSTom <value>

  :BUS<n>:SENT:BAUDtick:CUSTom ?

- **Function description:**

  Set or query SENT decoding clock cycle, default unit is us。

- **Parameter:**

  n:integer，range1~2

  value:integer，range 50ns-300us

- **Return format:**

  The query returns SENT custom clock cycle。

- **For example:**

  :BUS1:SENT:BAUDtick:CUSTom 1.25000E-004 Set the SENT decoding clock cycle of BUS1 to 125us

  :BUS1:SENT:BAUDtick:CUSTom ? The query returns1.25000E-004

:BUS<n>:SENT:POLarity

- **Command format:**

  :BUS<n>:SENT:POLarity <polarity>

  :BUS<n>:SENT:POLarity ?

- **Function description:**

  Set or query SENT decoding polarity。

- **Parameter:**

  n:integer，range 1~2

  polarity:Discrete type{POSitive | NEGative}

  POSitive:positive polarity

  NEGative:negative polarity

- **Return format:**

  The query returnsSENT decoding polarity。

- **For example:**

  :BUS1:SENT:POLarity POSitive Set the SENT decoding polarity of BUS1 to POSitive

  :BUS1:SENT:POLarity ? The query returnsPOSitive

:BUS<n>:SENT:THReshold

- **Command format:**

  :BUS<n>:SENT:THReshold <value>

  :BUS<n>:SENT:THReshold ?

- **Function description:**

  Set or query SENT decoding threshold, default unit V。

- **Parameter:**

  n:integer，range1~2

value:real，range-12V~12V

- **Return format:**

  The query returnsSENT decoding threshold。

- **For example:**

  :BUS1:SENT:THReshold 1　　Set the SENT decoding threshold of BUS1 to 1V

  :BUS1:SENT:THReshold ?　　The query returns1.00000E+000

:BUS<n>:SENT:DLENgth

- **Command format:**

  :BUS<n>:SENT:DLENgth <value>

  :BUS<n>:SENT:DLENgth ?

- **Function description:**

  Set or query SENT decoding data length。

- **Parameter:**

  n:integer，range1~2

  value:Discrete type{1 | 2 | 3 | 4 | 5 | 6}

- **Return format:**

  The query returnsSENT data length。

- **For example:**

  :BUS1:SENT:DLENgth 2　　Set the SENT decoding data length of BUS1 to 2Nibbles

  :BUS1:SENT:DLENgth ?　　The query returns2

:BUS<n>:SENT:PAUSe

- **Command format:**

  :BUS<n>:SENT:PAUSe <type>

  :BUS<n>:SENT:PAUSe ?

- **Function description:**

  Set or query whether SENT decoding has pause bits。

- **Parameter:**

  n:integer，range1~2

  type:Discrete type{NO | YES}

  NO:not have

  YES:have

- **Return format:**

  The query returnsSENT decoding pause bit。

- **For example:**

  :BUS1:SENT:PAUSe YES Set SENT decoding with pause bit for BUS1

  :BUS1:SENT:PAUSe ? The query returns YES

:BUS<n>:SENT:MODe

- **Command format:**

  :BUS<n>:SENT:MODe <value>

  :BUS<n>:SENT:MODe ?

- **Function description:**

  Set or query SENT decoding mode。

- **Parameter:**

  n:integer，range1~2

  type:Discrete type{Fast | Slow}

  Fast:Fast track

  Slow:Slow track

- **Return format:**

  The query returnsSENT decoding mode。

- **For example:**

  :BUS1:SENT:MODe Fast Set the SENT decoding mode of BUS1 to Fast

  :BUS1:SENT:MODe ? The query returns Fast

:BUS<n>:SENT:SEGment

- **Command format:**

  :BUS<n>:SENT:SEGment <value>

  :BUS<n>:SENT:SEGment ?

- **Function description:**

  Set or query SENT data segment format。

- **Parameter:**

  n:integer，range1~2

  type:Discrete type{Nib | Fast}

  Nib:Nibble

Fast:Fast track

- **Return format:**

  The format of the query returnsSENT data segment。

- **For example:**

  :BUS1:SENT:SEGment Fast Set the SENT data segment format of BUS1 to Fast

  :BUS1:SENT:SEGment ? The query returns Fast

:BUS<n>:SENT:EBXTol

- **Command format:**

  :BUS<n>:SENT:EBXTol <value>

  :BUS<n>:SENT:EBXTol ?

- **Function description:**

  Set or query SENT clock tolerance。

- **Parameter:**

  n:integer，range1~2

  type:real range 1%-30%

- **Return format:**

  The query returnsSENT clock tolerance。

- **For example:**

  :BUS1:SENT:EBXTol 20 Set the SENT clock tolerance of BUS1 to 20%

  :BUS1:SENT:EBXTol ? The query returns 20

Aux In/Out

:AUX:INPut:TYPe

- **Command format:**

  :AUX:INPut:TYPe <type>

  :AUX:INPut:TYPe ?

- **Function description:**

  Set or query the type of auxiliary input.

- **Parameter:**

  type: Discrete type {CLOSe | TRIGger | AWGSync}

  CLOSe: Close

TRIGger: Trigger Synchronization

AWGSync: AWG external trigger

- **Return format:**

  The query returns the type of auxiliary input.

- **For example:**

  :AUX:INPut:TYPe TRIGger

  Set the auxiliary input as Trigger Synchronization.

  :AUX:INPut:TYPe ?

  The query returns TRIGger.

:AUX:INPut:POLarity

- **Command format:**

  :AUX:INPut:POLarity <type>

  :AUX:INPut:POLarity ?

- **Function description:**

  Set or query the polarity of the input.

- **Parameter:**

  type: Discrete type {POSitive | NEGative}

  RISe: Positive polarity

  NEGative: Negative polarity

- **Return format:**

  The query returns the input polarity.

- **For example:**

  :AUX:INPut:POLarity NEGative

  Set the input polarity to NEGative.

  :AUX:INPut:POLarity ?

  The query returns NEGative.

:AUX:OUTPut:TYPe

- **Command format:**

  :AUX:OUTPut:TYPe <type>

  :AUX:OUTPut:TYPe ?

- **Function description:**

Set or query the type of auxiliary output.

- **Parameter:**

  type: Discrete type{CLOSe | TRIGger | AWGSync | PFTest}

  CLOSe:close

  TRIGger:Trigger synchronization

  AWGSync:AWG trigger

  PFTest: Pass/Fail test

- **Return format:**

  The query returns the type of auxiliary output.

- **For example:**

  :AUX:OUTPut:TYPe TRIGger

  Set the auxiliary output as trigger synchronization.

  :AUX:OUTPut:TYPe ?

  The query returns TRIGger.

:AUX:OUTPut:POLarity

- **Command format:**

  :AUX:OUTPut:POLarity <type>

  :AUX:OUTPut:POLarity ?

- **Function description:**

  Set or query the polarity of the output.

- **Parameter:**

  type: Discrete type {RISe | FALL}

  RISe: Positive polarity

  FALL: Negative polarity

- **Return format:**

  The query returns the output polarity.

- **For example:**

  :AUX:OUTPut:POLarity FALL

  Set the output polarity to NEGative.

  :AUX:OUTPut:POLarity ?

  The query returns FALL.

NET

:NETWork:LAN:COUNt

- **Command format:**

  :NETWork:LAN:COUNt ?

- **Function description:**

  Query the number of Ethernet network adapters that are currently in the connected state.

:NETWork:LAN:SELect

- **Command format:**

  :NETWork:LAN:SELect <index>

  :NETWork:LAN:SELect ?

- **Function description:**

  Set or query the currently selected Ethernet network adapter.

- **Parameter:**

  index: Integer type, index is less than or equal to the number of connected Ethernet network adapters queried.

- **Return format:**

  The query returns the selected Ethernet network adapter.

- **For example:**

  :NETWork:LAN:SELect 1

  Select the first Ethernet network adapter.

  :NETWork:LAN:SELect ?

  The query returns 1.

:NETWork:LAN:RESet

- **Command format:**

  :NETWork:LAN:RESet

- **Function description:**

  Used to immediately reset the current set network parameters to the default parameters.

:NETWork:LAN:APPLy

- **Command format:**

  :NETWork:LAN:APPLy

- **Function description:**

  Set the selected adapter to immediately take effect the currently set network parameters.

:NETWork:LAN:IPADdress

- **Command format:**

  :NETWork:LAN:IPADdress <adress>

  :NETWork:LAN:IPADdress ?

- **Function description:**

  Used to set or query the IP address of the selected adapter, in the format of xxx.xxx.xxx.xxx.

- **Parameter:**

  adress: String type

- **Return format:**

  The query returns the IP address.

- **For example:**

  :NETWork:LAN:IPADdress “192.168.137.10”

  Set the IP address to 192.168.137.10.

  :NETWork:LAN:IPADdress ?

  The query returns 192.168.137.10.

:NETWork:LAN:SMASK

- **Command format:**

  :NETWork:LAN:SMASK <smask>

  :NETWork:LAN:SMASK ?

- **Function description:**

  Set or query the subnet mask for the selected adapter, in the format of xxx.xxx.xxx.xxx.

- **Parameter:**

  smask: String type

- **Return format:**

  The query returns the subnet mask.

- **For example:**

  :NETWork:LAN:SMASK “255.255.255.0”

  Set the subnet mask to 255.255.255.0.

  :NETWork:LAN:SMASK ?

  The query returns 255.255.255.0.

:NETWork:LAN:GATEway

- **Command format:**

  :NETWork:LAN:GATEway <gateway>

  :NETWork:LAN:GATEway ?

- **Function description:**

  Set or query the default gateway of the selected adapter, in the format of xxx.xxx.xxx.xxx.

- **Parameter:**

  gateway: String type

- **Return format:**

  The query returns the default gateway.

- **For example:**

  :NETWork:LAN:GATEway “192.168.137.1”

  Set the default gateway to 192.168.137.1.

  :NETWork:LAN:GATEway ?

  The query returns 192.168.137.1.

:NETWork:LAN:DHCP

- **Command format:**

  :NETWork:LAN:DHCP <type>

  :NETWork:LAN:DHCP ?

- **Function description:**

  Set or query the configuration mode (Automatic IP) and (Manual IP) of the selected adapter.

- **Parameter:**

  type: Discrete type {AUTO | MANual}

  AUTO:Auto

  MANual:Manual

- **Return format:**

The query returns the dynamic configuration mode.

- **For example:**

  :NETWork:LAN:DHCP AUTO

  Set the configuration mode as AUTO.

  :NETWork:LAN:DHCP ?

  The query returns AUTO.

:NETWork:LAN:SPEed

- **Command format:**

  :NETWork:LAN:SPEed ?

- **Function description:**

  Used to query the network transmission speed of the selected adapter.

:NETWork:LAN:MAC

- **Command format:**

  :NETWork:LAN:MAC ?

- **Function description:**

  Query the MAC address.

Limited Warranty and Liability

UNI-T guarantees the Instrument product is free from material and workmanship defects for three years from the purchase date. This warranty excludes damages resulting from accidents, negligence, misuse, modification, contamination, or improper handling. If you need warranty service within the warranty period, please contact your seller directly. UNI-T will not be responsible for any special, indirect, incidental or subsequent damage or loss caused by using this device. For the probes and accessories, the warranty period is one year. Visit instrument.uni-trend.com for full warranty information.

Learn more at: www.uni-trend.com

Register your product to confirm your ownership. You will also get product notifications, update alerts, exclusive offers and all the latest information you need to know

UNI-T is the licensed trademark of UNI-TREND TECHNONOLGY CO., Ltd. The product information in this document subject to update without notice. For more information on UNI-T Test & Measure Instrument products, applications or service, please contact UNI-T instrument for support, the support center is available on www.uni-trend.com ->instruments.uni-trend.com

https:\\instruments.uni-trend.com/ContactForm/

Headquarter

Addresses: No6, Gong Ye Bei 1st Road. Songshan Lake National Hiah-Tech Industrial Development Zone, Dongguan City, Guangdong Province, China

Tel: (86-769) 8572 3888

Europe

UNI-TREND TECHNOLOGY EU GmbH

Addresses: Affinger Str. 12 86167 Augsburg Germany

Tel: +49 (0)821 8879980

Europe

UNI-TREND TECHNOLOGY EU GmbH

Addresses: Affinger Str. 12 86167 Augsburg Germany

Tel: +49 (0)821 8879980

North America

Uni-Trend Technology US INC.

Addresses: 3171 Mercer Ave STE 104, Bellingham, WA 98225

Tel: +1-888-668-8648

North America

Uni-Trend Technology US INC.

Addresses: 3171 Mercer Ave STE 104, Bellingham, WA 98225

Tel: +1-888-668-8648