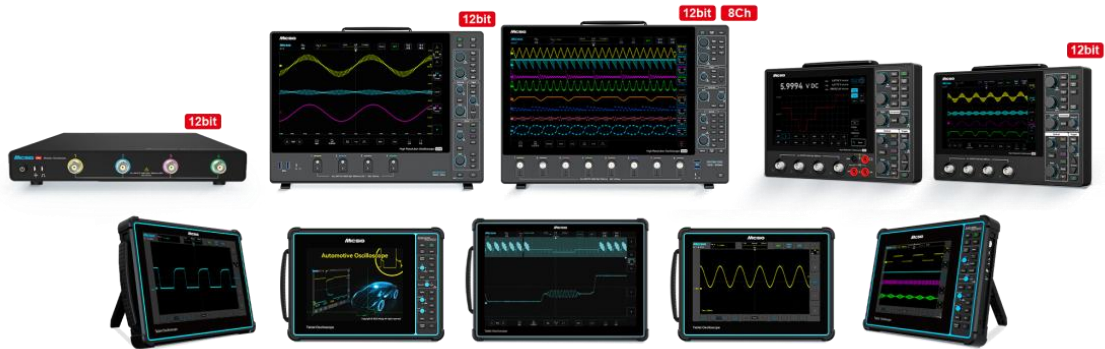


# Micsig Oscilloscope

## SCPI Commands Manual



\*Applicable to High Resolution Oscilloscope MHO6 series, MHO3 series, MHO1 series, MO3 series, MDO series, ETO series, STO series, SATO series, TO series, ATO series

Shenzhen Micsig Technology Co., Ltd.

---

## Contents

1. Document Introduction.....	1
1.1 Purpose of this document.....	1
1.2 Target Audience.....	1
1.3 Reference Documentation.....	1
1.4 Terminology.....	1
2. product description.....	2
2.1 Product Background.....	2
2.2 Product target user groups and demand description.....	2
2.3 Roles in the product.....	2
2.4 Business workflow.....	2
2.5 Target operating software and hardware environment.....	3
2.6 Constraints and Restrictions.....	3
2.7 Applicable interface.....	3
3. SCPI Requirements.....	4
3.1 Introduction to SCPI.....	4
3.1.1 Command Format.....	4

---

3.1.2 Explanation of symbols.....	4
3.1.3 Parameter Types.....	5
3.1.4 Command abbreviation.....	5
3.2 Command system.....	6
3.2.1 Common commands.....	6
3.2.2 :MENU menu function commands.....	7
3.2.3 Sampling Command Subsystem.....	15
3.2.4 Channel command subsystem.....	24
3.2.5 Math command subsystem.....	33
3.2.6 Cursor Command Subsystem.....	50
3.2.7 Display command subsystem.....	61
3.2.8 Measurement command subsystem.....	66
3.2.9 Trigger command subsystem.....	79
3.2.10 Time base command subsystem.....	114
3.2.11 Storage command subsystem.....	117
3.2.12 Bus configuration command subsystem.....	122
3.2.13 Reference waveform command subsystem.....	137

3.2.14 AUTO Setting Subsystem ..... 142

3. 2. 15 Waveform command subsystem ..... 147

# 1. Document Introduction

## 1.1 Purpose of this document

This document aims to define the SCPI requirements for oscilloscopes and to provide preparation for Micsig oscilloscopes to support the SCPI protocol and comply with the IEEE488.2 standard.

## 1.2 Target Audience

Developers and testers

## 1.3 Reference Documentation

## 1.4 Terminology

Abbreviations and terms	explain
SCPI	Standard Commands for Programmable Instruments Manual
...	

## 2. product description

### 2.1 Product Background

The SCPI command processing module is embedded in our products to comply with the IEEE488.2 standard. As an SCPI instrument, we must develop it strictly in accordance with the provisions of the IEEE488.2 standard for instruments.

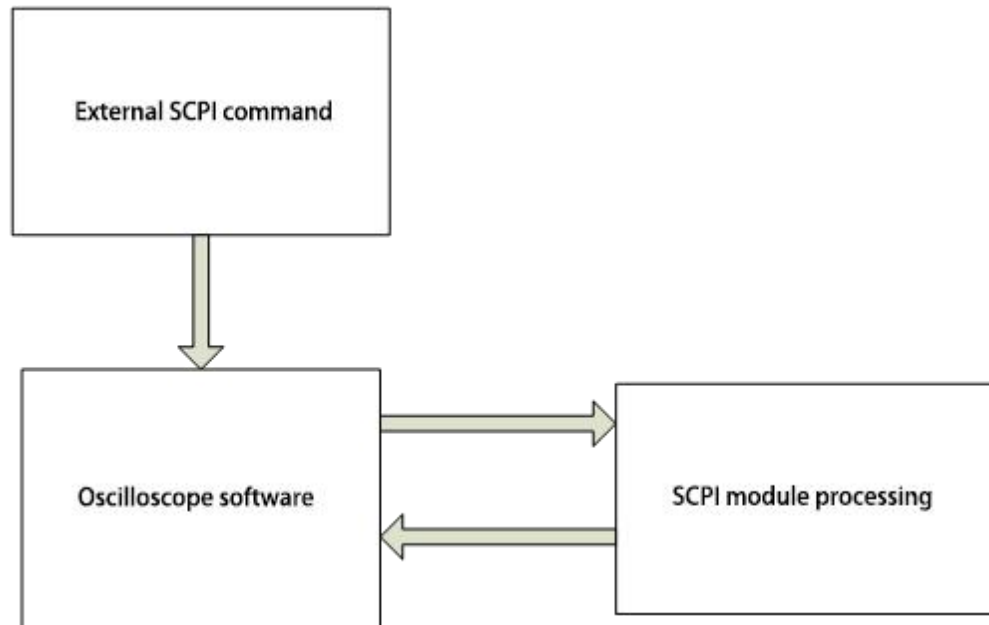
### 2.2 Product target user groups and demand description

The SCPI command processing module is only for the software itself and is used to process all SCPI commands sent to the software from outside the device.

### 2.3 Roles in the product

Role Name	Responsibilities
Common Command System	Handles common commands for all instruments
Essential command system	Essential commands for handling instruments and equipment
other	Optional commands for handling instrumentation

### 2.4 Business workflow



## 2.5 Target operating software and hardware environment

Operating system:Android

Hardware environment:Tablet oscilloscope

## 2.6 Constraints and Restrictions

Since this module is a module of the software, the constraints and limitations are the same as those of the product.

## 2.7 Applicable interface

USB, LAN, WIFI.

## 3. SCPI Requirements

### 3.1 Introduction to SCPI

#### 3.1.1 Command Format

SCPI commands are tree-like hierarchical structures, including multiple subsystems, each of which consists of a root keyword and one or more hierarchical keywords.

The command line usually starts with a colon ":"; **keywords are separated by colons ":"**, followed by optional parameter settings; a question mark "?" is added after the command line to indicate a query for this function; commands and parameters are separated by "spaces".

#### 3.1.2 Explanation of symbols

##### 1、 big parantheses { }

The content in curly brackets is the parameter options. Parameter items are usually separated by a vertical line "|". When using a command, you must select one of the parameters.

##### 2、 Vertical Line |

The vertical bar is used to separate multiple parameter options. When using a command, you must select one of the parameters.

##### 3、 Square brackets [ ]

The text in square brackets is optional.

---

#### 4、 Triangle brackets < >

The parameters in the triangle brackets must be replaced by a valid value.

### **3.1.3 Parameter Types**

#### 1、 Bool

The parameter values are "OFF", "ON", "0", or "1".

#### 2、 Discrete

The parameter values are the listed options.

#### 3、 Integer

Unless otherwise specified, the parameter can be any integer (NR1 format) within the valid value range. Note that please do not set the parameter to decimal format at this time, otherwise an exception will occur.

#### 4、 Real

The parameter can be any real number within the valid value range. The command accepts parameter input in decimal (NR2 format) and scientific notation (NR3 format).

#### 5、 ASCII string

The parameter value is a combination of ASCII characters.

### **3.1.4 Command abbreviation**

All commands are not case sensitive and can be all uppercase or lowercase.

However, if you want to abbreviate, you must enter all capital letters in the command format.

## 3.2 Command system

### 3.2.1 Common commands

#### 3.2.1.1 *\*IDN*

**Function:**Read oscilloscope related information. Include version number, manufacturer, product model, and product serial number.

**Format:**\*IDN?

**Return format:**

Micsig,< model>,<serial number >,XXXXXX

<model>:instrument model.

<serial number >:Instrument serial number.

XXXXXX:Instrument software version.

**Example:**

Micsig,MDO5004,390000029,1.388.132

#### 3.2.1.2 *:SYS:SCR?*

**Function:** Queries the data stream of the current oscilloscope screen image.

**Format:** :SYS:SCR?

**Return Format:** The query returns a binary data stream of the screen capture in PNG format. The returned data consists of three parts: an identifier, the data length, and the screenshot data stream.

**Example return data:**

```
#9000358370\FF\D8X\00\00\10JFIF\00\01\01\
```

In this data stream, # is the identifier, 9 indicates that 9 bytes are used to describe the data length, 000358370 specifies the length of the returned data stream, and the subsequent FF\D8X\00\00\10JFIF\00\01\01\ represents the screenshot data stream.

### **3.2.2 :MENU menu function commands**

#### ***3.2.2.1 :MENU:AUTO***

Automatic configuration can quickly configure the oscilloscope to display the best effect for the input signal. The automatic configuration includes: applicable to single channel and multiple channels ; automatic adjustment of signal horizontal scale , vertical scale and trigger level ; oscilloscope waveform reverse is turned off, bandwidth is set to full bandwidth, coupling mode is DC coupling, sampling mode is normal sampling ; trigger is set to edge trigger, trigger mode is automatic.

Function: Start or stop the automatic setting (auto range).

Automatic configuration can quickly configure the oscilloscope to display the best effect for the input signal. The automatic configuration includes: applicable to single channel and multiple channels ; automatic adjustment of signal horizontal scale , vertical scale and trigger level ; oscilloscope waveform reverse is turned off, bandwidth is set to full bandwidth, coupling mode is DC coupling, sampling mode is normal sampling ; trigger is set to edge trigger, trigger mode is automatic.

**Format:** :MENU:AUTO <bool>

:MENU:AUTO?

Among them, <bool>, Boolean type, {{0|OFF}}{1|ON}}.

**Return format:** The query returns "0" or "1".

**Example:**

The following command turns on the display of channel 1.

:MENU:AUTO ON or:MENU:AUTO 1

The query below returns "1".

:MENU:AUTO?

### **3.2.2.2 :MENU:RUN**

---

**Function:** Make the oscilloscope start running, meet the trigger conditions, and start collecting data.

**Format:** :MENU:RUN

### ***3.2.2.3 :MENU:STOP***

**Function:** Stop the oscilloscope and data acquisition.

**Format:** :MENU:STOP

### ***3.2.2.4 :MENU:SINGLE***

**Function:** Set the oscilloscope to single sequence. The oscilloscope captures and displays a single acquisition.

**Format:** :MENU:SINGLE

### ***3.2.2.5 :MENU:RESet***

**Function:** Restores the oscilloscope settings to factory defaults.

**Format:** :MENU:RESet

### ***3.2.2.6 :MENU:LOCK <bool>***

**Function:** Close/cancel closing the oscilloscope touch screen.

**Format:** :MENU:LOCK <bool>

:MENU:LOCK?

Among them, bool, Boolean type, {{0|OFF}}{1|ON}}.

**Return format:**The query returns "0" or "1".

**Example:**

The following command turns off the touch screen.

:MENU:LOCK ON or MENU:LOCK 1

The query below returns "1".

:MENU:LOCK?

### **3.2.2.7 :MENU:HALF**

#### **3.2.2.7.1 :MENU:HALF:CHANnel**

**Function:** Set the channel position to the vertical zero position (vertical center of the waveform display area).

**Format:** :MENU:HALF:CHANnel <channel>

Among them, <channel>, discrete type, {CH1|CH2|CH3|CH4}.

#### **3.2.2.7.2 :MENU:HALF:TRIGpos**

**Function:** Set the trigger position to the middle of the screen.

**Format:** :MENU:HALF:TRIGpos <source>

Where <source> is discrete, {CH1|CH2|CH3|CH4 }.

### 3.2.2.7.3 :MENU:HALF:XCURsor

**Function:** Set the vertical cursor of the channel to 50%.

**Format:** :MENU:HALF:XCURsor

### 3.2.2.7.4 :MENU:HALF:YCURsor

**Function:** Set the channel's horizontal cursor to 50%.

**Format:** :MENU:HALF:YCURSor

### 3.2.2.7.5 :MENU:HALF:LEVel

**Function:** Set the trigger level to the middle position of the trigger signal amplitude.

**Format:** :MENU:HALF:LEVel <channel>

Among them, <channel>, discrete type, {CH1|CH2|CH3|CH4}, defaults to the current channel.

### 3.2.2.8 :MENU:CHANnel <n>,<bool>

**Function:** Open or close the channel menu

**Format:** :MENU:CHANnel <n>,<bool>

:MENU:CHANnel? <n>

---

Among them, <n>, discrete type, {CH1|CH2|CH3|CH4 |MATH | REF|S1|S2 };

<bool>, Boolean type, {{0| OFF}| {1||ON}}.

**Return format:** The query returns "0" or "1".

**Example:**

The following command turns on the display of channel 1.

:MENU:CHANnel CH1, ON or:MENU:CHANnel CH1, 1

The query below returns "1".

:MENU:CHANnel? CH1

**3.2.2.9 :MENU:QUICK <bool>**

**Function:** Open or close the shortcut menu (bottom menu)

**Format:** :MENU:QUICK <bool>

:MENU:QUICK?

Where <bool> is a Boolean type, {{0|OFF}|{1||ON}}.

**Return Format:** The query returns "0" or "1".

**Example:**

The following command turns on the display of channel 1.

:MENU:QUICK ON or :MENU:QUICK 1

The following query returns "1".

```
:MENU:QUICK?
```

### **3.2.2.10 :MENU:MAIN <bool>**

**Function:** Open or close the main menu (top menu)

**Format:** :MENU:MAIN <bool>

```
:MENU:MAIN?
```

Among them <bool>, Boolean type, {{0|OFF}}{1|ON}}.

**Return format:** The query returns "0" or "1".

**Example:**

The following command turns on the display of channel 1.

```
:MENU:MAIN ON or :MENU:MAIN 1
```

The query below returns "1".

```
:MENU:MAIN?
```

### **3.2.2.11 :MENU:AUX:TRIGger**

#### **3.2.2.11.1 :MENU:AUX:TRIGger <type>**

**Function:** Sets the external trigger input/output.

**Format:** :MENU:AUX:TRIGger <type>

:MENU:AUX:TRIGger?

Among them <type> , discrete type, {IN | OUT}.

**Return Format:** The query returns "IN" or "OUT".

**Example:**

The following command switches to trigger input:

:MENU:AUX:TRIGger?

### 3.2.2.11.2 :MENU:AUX:INPutres

**Function:** Sets the trigger input impedance.

**Format:** :MENU:AUX:INPutres <type>

:MENU:AUX:INPutres?

Among them <type> — Discrete type, {FIFTy | MEGA}.

**Return Format:** The query returns "FIFTy" or "MEGA".

**Example:**

The following command sets the trigger input impedance to 50 ohms:

:MENU:AUX:INPutres FIFTy

### 3.2.2.12: :MENU:AUX:CLOCK

**Function:** Sets the clock input/output.

**Format:** :MENU:AUX:CLOCK <type>

:MENU:AUX:CLOCK?

Among them <type>, discrete type, {IN | OUT}.

**Return Format:** The query returns "IN" or "OUT".

**Example:**

The following command switches to clock output:

:MENU:AUX:CLOCK OUT

The following query returns "OUT".

:MENU:AUX:CLOCK?

### 3.2.3 Sampling Command Subsystem

#### 3.2.3.1 :ACquire:TYPE

**Function:** Set the sampling method.

**Format:** :ACquire:TYPE <type>

:ACquire:TYPE?

Where <type> is discrete, { NORMal | MEAN | ENVelop | PEAK }

**Return format:** The query returns "NORMal", "MEAN", "PEAK", "ENVelop".

**Example:**

The following command selects the envelope sampling mode.

```
:ACQUIRE:TYPE ENVELOP
```

The query below returns " ENVELOP".

```
:ACQUIRE:TYPE?
```

### ***3.2.3.2 :ACQUIRE:MEAN***

**Function:** Set the average number of samples. The value set is an integer multiple of 2.

**Format:** :ACQUIRE:MEAN <count>

```
:ACQUIRE:MEAN?
```

Among them, <count>, discrete type, {2|4|8|16|32|64|128|256}

**Return format:** The query returns an integer.

#### **Example:**

The following command sets the average number of samples to "32".

```
:ACQUIRE:MEAN 32
```

The query below returns "32".

```
:ACQUIRE:MEAN?
```

### ***3.2.3.3 :ACQUIRE:ENVELOP***

**Function:** Set the envelope sampling times. The value to be set is an integer multiple of 2 or infinity.

**Format:** :ACQUIRE:ENVELOP <count>

:ACQUIRE:ENVELOP?

Where <count> is a discrete type, {2|4|8|16|32|64|128|256|inf}.

**Return Format:** The query returns an integer.

**Example:**

The following command sets the envelope sampling number to "32".

:ACQUIRE:ENVELOP 32

The following query returns "32".

:ACQUIRE:ENVELOP?

**3.2.3.4 :ACQUIRE:SEGMENTED**

**Function:** Set segment storage.

**3.2.3.4.1 :ACQUIRE:SEGMENTED <bool>**

:ACQUIRE:SEGMENTED?

Set and query whether segment storage is on or off;

Where bool, Boolean type, {0|OFF}{1|ON};

Example

The following command turns on segmented storage.

```
:ACQUIRE:SEGMentEd ON
```

The following query returns "1".

```
:ACQUIRE:SEGMentEd?
```

#### **3.2.3.4.2 :ACQUIRE:SEGMentEd:NO?**

```
:ACQUIRE:SEGMentEd:NO?
```

Query the number of segments that have been triggered currently;

Example

The following query returns "1003", indicating that there are currently 1003 segments that have been triggered and stored in the FPGA's memory.

```
:ACQUIRE:SEGMentEd:NO?
```

#### **3.2.3.4.3 :ACQUIRE:SEGMentEd:QTY <NO>**

```
:ACQUIRE:SEGMentEd:QTY?
```

Set and query the number of segments stored in segmented form;

Where no, integer, refer to the data manual;

Example

The following command sets the number of segmented storage segments to 4.

```
:ACQuire:SEGMented:QTY 4
```

The query below returns "4".

```
:ACQuire:SEGMented:QTY?
```

#### **3.2.3.4.4 :ACQuire:SEGMented:DISType < type >**

```
:ACQuire:SEGMented:DISType?
```

Set and query the display mode of segmented storage;

Where type , discrete type, { SINGL e| FIT }; SINGL e is a single frame display,  
FIT is a fitting display

Example

The following command sets up segmented storage for single frame display.

```
:ACQuire:SEGMented:DISType SINGLE
```

The query below returns " SINGLE ".

```
:ACQuire:SEGMented:DISType?
```

#### **3.2.3.4.5 :ACQuire:SEGMented:ORDER <type>**

```
:ACQuire:SEGMented:ORDER?
```

Set and query the segment storage playback order;

---

Where type, discrete, { ORD er | REOR der } ORD er is the order, REOR der is reverse order

Example

The following command sets segment storage sequential playback.

```
:ACQUIRE:SEGMmented:ORDER ORDER
```

The following query returns " ORDER ".

```
:ACQUIRE:SEGMmented:ORDER?
```

#### **3.2.3.4.6 :ACQUIRE:SEGMmented:PLAY**

Start autoplay

```
:ACQUIRE:SEGMmented:STOP
```

Pause autoplay

#### **3.2.3.4.7 :ACQUIRE:SEGMmented:FRA1 <value>**

```
:ACQUIRE:SEGMmented:FRA1?
```

Set and query the current frame when displaying a single frame;

Among them, value, integer type, 1~maximum value when stopping

Example

The following command sets the current frame to 546.

:ACQUIRE:SEGMmented:FRA1 546

The query below returns "546".

:ACQUIRE:SEGMmented:FRA1?

#### **3.2.3.4.8 :ACQUIRE:SEGMmented:FRA2 <value>**

:ACQUIRE:SEGMmented:FRA2<value>

:ACQUIRE:SEGMmented:FRA2?

Set the initial frame when displaying the query fit;

Where value is integer, 1 to the maximum value when stopping

Example

The following command sets the initial frame of the fitting display to 100.

:ACQUIRE:SEGMmented:FRA2 100

The following query returns "100".

:ACQUIRE:SEGMmented:FRA2?

#### **3.2.3.4.9 :ACQUIRE:SEGMmented:FRA3 <value>**

:ACQUIRE:SEGMmented:FRA3?

Set and query the end frame of the fitting display;

Where value, integer, FR2~the maximum value when stopped

Example

The following command sets the fitting display end frame to 150.

```
:ACQUIRE:SEGMmented:FRA3 150
```

The following query returns "150".

```
:ACQUIRE:SEGMmented:FRA3?
```

#### **3.2.3.4.10 :ACQUIRE:SEGMmented:PLAY:SPED <sped>**

```
:ACQUIRE:SEGMmented:SPED?
```

Set and query the speed of automatic playback;

Among them, sped, discrete type, {1|2|4|8}

Example

The following command sets the playback speed to 4 times.

```
:ACQUIRE:SEGMmented:PLAY:SPED 4
```

The following query returns "4".

```
:ACQUIRE:SEGMmented:PLAY:SPED?
```

#### **3.2.3.5 :ACQUIRE:SRATE**

**Function:** Query the sampling rate of the current analog channel.

**Format:** :ACQUIRE:SRATE?-

### 3.2.3.6 :ACquire:DEPSelect

**Function:** Set and query the current storage depth of the oscilloscope.

Format: :ACquire:DEPSelect <type>

:ACquire:DEPSelect?

Among them, <type>, discrete type, supports different values according to different models, and can be set to { AUTO|110000000|11000000|1100000|110000 | 11000}

Example

The following command sets the memory depth to AUTO.

```
:ACquire:DEPSelect 22000000
```

The following query returns "22000000".

```
:ACquire:DEPSelect?
```

### 3.2.3.7 :ACquire:DEPTh?

**Function:** Query the actual value of the current storage depth of the oscilloscope.

Format: :ACquire:DEPTh?

Among them, <type>, discrete type, returns the actual value of storage depth depending on the model.

Example

The query below returns "22000000".

```
:ACQUIRE:DEPTH?
```

### 3.2.4 Channel command subsystem

#### 3.2.4.1 :CHANNEL <n>:DISPLAY <bool>

**Function:** Open or close the channel

**Format:** :CHANNEL <n>:DISPLAY <bool>

```
:CHANNEL <n>:DISPLAY?
```

Among them, <n>, discrete type, { 1|2|3|4 }; <bool>, Boolean type, {{0| OFF} {1|ON}}.

**Return format:** The query returns "0" or "1".

**Example:**

The following command turns on the display of channel 1.

```
:CHANNEL1:DISPLAY ON or :CHANNEL1:DISPLAY 1
```

The query below returns "1".

```
:CHANNEL1:DISPLAY?
```

### 3.2.4.2 :CHANnel <n>:INVerse <bool>

**Function:** Turn on or off the inverted display of the analog channel.

**Format:** :CHANnel <n>:INVerse <bool>

:CHANnel <n>:INVerse?

Among them, <n>, discrete type, { 1|2|3|4 }; <bool>, Boolean type, {{0| OFF} {1|ON}}.

**Return Format:**The query returns "0" or "1".

#### Example:

The following command turns on the inverted display of channel 1.

:CHANnel1:INVerse ON or :CHANnel1:INVerse 1

The following query returns "1".

:CHANnel1:INVerse?

### 3.2.4.3 :CHANnel <n>:BAND <type> , <freq>

**Function:**Set the bandwidth limit of the analog channel to "20M" or "Full Bandwidth".

**Format:** :CHANnel <n>:BAND <type> , <freq>

:CHANnel <n>:BAND?

Parameters:<n>, discrete type, { 1|2|3| 4} ; <type>, discrete type, {20M|FULL|HIGH|LOW}; <freq> , real type, {refer to data sheet}, only Valid under "HIGH" and "LOW".

**Return format:**The query returns "20M", "FULL", "HIGH", and "LOW".

**Example:**

The following command sets the bandwidth limit of channel 1 to High ,10000000 .

```
:CHANnel1:BAND HIGH , 10000000
```

[Note] "10000000 " can be any value. This value is invalid under 20M and FULL.

```
:CHANnel1:BAND?
```

**3.2.4.4 :CHANnel <n>:PRTY <type>**

**Function:**Set the probe type of the analog channel to "voltage" or "current".

**Format:** :CHANnel <n>:PRTY <type>

```
:CHANnel <n>:PRTY?
```

Among them, <n>, discrete type, { 1|2|3|4} ; <type>, discrete type, {VOL|CUR | BAR|MPA| PSI }. (The three parameters marked are suitable for automotive oscilloscopes)

**Return format:** The query returns "VOL", "CUR", " BAR ", " MPA " or "PSI " .

**Example:**

The following command plots channel 1 with probe type voltage.

```
:CHANnel1:PRTY VOL
```

The query below returns "VOL".

```
:CHANnel1:PRTY?
```

**3.2.4.5 :CHANnel <n>:PROBe < atten >**

**Function:** Set the attenuation ratio of the analog channel probe.

**Format:** :CHANnel <n>:PROBe < atten >

```
:CHANnel <n>:PROBe?
```

Wherein, <n> is discrete, { 1|2|3|4} ; < atten > is discrete,

{ 0.001|0.002|0.005|0.01|0.02|0.05|0.1|0.2|0.5|1|2|5|10|20|50|100|200|500|1000}.

**Return format:** The query returns "0.001", "0.002", "0.005", "0.01", "0.02", "0.05", "0.1", "0.2", "0.2", "1", "2", "5", "10", "20", "50", "100", "200", "500", "1000".

**Example:**

The following command sets the attenuation ratio of the probe connected to channel 1 to 10.

```
:CHANnel1:PROBe 10
```

The following query returns "10".

:CHANnel1:PROBe?

### 3.2.4.6 :CHANnel <n>:COUPle <couple>

**Function:** Set the analog channel input coupling mode to "AC", "DC" or "GND".

**Format:** :CHANnel <n>:COUPle <couple>

:CHANnel <n>:COUPle?

Among them, <n>, discrete type, { 1|2|3|4 }; <couple>, discrete type, {AC|DC|GND}

**Return format:** The query returns "AC", "DC" or "GND".

#### Example:

The following command sets the input coupling mode of channel 1 to "AC".

```
:CHANnel1:COUPle AC
```

The query below returns "AC".

```
:CHANnel1:COUPle?
```

### 3.2.4.7 :CHANnel <n>:INPutres <input>

**Function:** Set the input impedance of the analog channel to "MEGA (1M  $\Omega$ )" or " FIFTy (50  $\Omega$ )".

**Format:** :CHANnel <n>:INPutres <input>

:CHANnel <n>:INPutres?

Where <n> is discrete, { 1|2|3|4 }; <input> is discrete, { MEGA| FIFTy }.

**Return format:** The query returns "MEGA" or " FIFTy ".

**Example:**

The following command sets the input impedance of channel 1 to 1M $\Omega$ .

```
:CHANnel1:INPutres MEGA
```

The following query returns "MEGA".

```
:CHANnel1:INPutres?
```

**3.2.4.8 :CHANnel <n>:SCALe <extent> (can also be used:CHANnel <n>:EXETent <extent>)**

**Function:** Set the vertical scale of the waveform display of the specified analog channel.

**Format:** :CHANnel <n>:SCALe <extent>

```
:CHANnel <n>:SCALe?
```

Where, <n> is a discrete type, {1|2|3|4}; <extent> is a real type, not exceeding the maximum and minimum values

Maximum value: oscilloscope maximum gear \* current probe multiple

Minimum value: oscilloscope minimum gear \* currently set probe multiple.

**Return Format:** The query returns the vertical scale value in scientific notation.

**Example:**

The following command sets the vertical scale of channel 1 to 1V/div.

```
:CHANnel1:SCALe 1
```

The following query returns "1.000000e+00".

```
:CHANnel1:SCALe?
```

**3.2.4.9 :CHANnel <n>:POSition <pos>**

**Function:** Set the vertical position of the specified channel waveform display.

**Format:** :CHANnel <n>:POSition <pos>

```
:CHANnel <n>:POSition?
```

Among them, <n>, discrete type, {1|2|3|4} ; <pos>, real type.

**Return format:** The query returns the offset value in scientific notation.

**Example:**

The following command sets the vertical offset of channel 1 to 0.01V.

```
:CHANnel1:POSition 0.01
```

The following query returns "1.000000e-02"

```
:CHANnel1:POSition?
```

**3.2.4.10 :CHANnel <n>:VREF <bool>**

**Function:** Set the vertical expansion base of the analog channel.

**Format:** :CHANnel <n>:VREF <bool>

:CHANnel <n>:VREF?

Among them: <n> , discrete type, { 1|2|3| 4} ; <bool> , discrete type, { CENTER|ZERO } .

**Return format:** The query returns " CENT " or " ZERO " .

**Example:**

The following command sets the vertical expansion datum of channel 1 to be the center.

:CHANnel1:VREF CENTER

The query below returns " CENT " .

:CHANnel1:VREF?

### **3.2.4.11 :CHANnel <n>:LABel < string >**

**Function:** Set the channel label of the analog channel.

**Format:** :CHANnel <n>:LABel < string >

:CHANnel <n>:LABel?

Where:<n> , discrete type, { 1|2|3| 4} ; <string> , string .

**Return format:** query return string.

**Example:**

The following command sets the label of channel 1 to DDR .

```
:CHANnel1:LABel DDR
```

The following query returns " DDR " .

```
:CHANnel1:LABel?
```

**3.2.4.12 :CHANnel <n>:LABel:CLEar**

**Function:** Clear channel label.

**Format:** :CHANnel <n>:LABel:CLEar

Among them:<n> , discrete type , { 1|2|3|4} .

**Example:**

The following command clears the label of channel 1 .

```
:CHANnel1:LABel:CLEar
```

**3.2.4.13 :CURRent:CHANnel < n >**

**Function:** Set the current channel.

**Format:** :Current:CHANnel <n>

```
:CURRent:CHANnel?
```

Among them: <n> , discrete type, {CH1|CH2|CH3|CH4|MATH|R1|R2}R3|R4|S1|S2} .

**Return format:** The query returns {CH1|CH2|CH3|CH4|MATH|R1|R2}R3|R4|S1|S2} .

**Example:**

The following command sets the vertical expansion datum of channel 1 to be the center.

:CURRent:CHANnel CH1

The query below returns " CH1 " .

:CURRent:CHANnel?

### 3.2.5 Math command subsystem

#### 3.2.5.1 :MATH:DISPlay

**Function:** Turn on or off the mathematical operation type.

**Format:** :MATH:DISPlay <bool>

:MATH:DISPlay?

Among them, <bool>, Boolean type, {{0|OFF}}{1|ON}}.

#### 3.2.5.2 :MATH:MODE

**Function:** Select the type of mathematical operation.

**Format:** :MATH:MODE <mode>

:MATH:MODE?

Among them, <mode>, discrete type, {BASE | FFT| AX+B|ADVAnced}.

**Return format:** The query returns "BASE", "FFT", "AX+B", and "ADVAnce d".

**Example:**

The following command selects the FFT operation.

:MATH:MODE FFT

The query below returns "FFT".

:MATH:MODE?

### **3.2.5.3 :MATH:VREF <bool>**

Function: Set the vertical expansion base of the math waveform.

Format: :MATH:VREF <bool>

:MATH:VREF?

Where:<bool> , discrete type, { CENTER| ZERO } .

Return format: The query returns "CENT " or " ZERO " .

Example:

The following command sets the vertical expansion base to the center.

:MATH:VREF CENTER

The following query returns "CENT".

:MATH:VREF?

### **3.2.5. 4 :MATH:BASE**

#### **3.2.5. 4 .1 :MATH:BASE:SOU1**

**Function:** Select source 1 for dual waveform operation

**Format:** :MATH:BASE:SOU1 <source>

:MATH:BASE:SOU1?

Among them, <source>, discrete type, {CH1|CH2|CH3|CH4}.

**Return format:** The query returns "CH1", "CH2", "CH3" or "CH4".

#### **Example:**

The following command selects channel 1 as source 1.

:MATH:BASE:SOU1 CH1

The query below returns "CH1".

:MATH:BASE:SOU1?

#### **3.2.5. 4.2 :MATH:BASE:SOU2**

**Function:** Select source 2 for dual waveform operation.

**Format:** :MATH:BASE:SOU2 <source>

:MATH:BASE:SOU2?

Among them, <source>, discrete type, {CH1|CH2|CH3|CH4}.

**Return format:** The query returns "CH1", "CH2", "CH3" or "CH4".

**Example:**

The following command selects channel 1 as source 2.

:MATH:BASE:SOU2 CH1

The query below returns "CH1".

:MATH:BASE:SOU2?

**3.2.5. 4.3 :MATH:BASE:VSCale**

**Function:** Set the vertical scale of the double waveform operation result.

**Format:** :MATH:BASE:VSCale < extent >

:MATH:BASE:VSCale?

Among them, <extent>, real type, <extent>, real type, {1e-15~5e14, can only be in steps of 1, 2, 5}.

**Return format:** The query returns the gear value in scientific notation.

**Example:**

The following command sets the vertical scale of the addition result to 1.

```
:MATH:BASE:VSCale 1
```

The query below returns "1.000000e+00".

```
:MATH:BASE:VSCale?
```

#### 3.2.5.4.4 :MATH:BASE:VPOSITION

Function: Set the vertical offset of the dual waveform operation result.

Format: :MATH:BASE:VPOSITION <position>

```
:MATH:BASE:VPOSITION?
```

Among them, <position> is real type and expressed in scientific notation.

```
:MATH:BASE:VPOSITION 8 /* Set the vertical offset to 8V*/
```

```
:MATH:BASE:VPOSITION? The query returns 8.000000E0*
```

#### 3.2.5.4.5 :MATH:BASE:OPERATOR

**Function:** Select the operator for addition operation

**Format:** :MATH:BASE:OPERATOR <operator>

```
:MATH:BASE:OPERATOR?
```

Among them, <operator> is discrete type, {ADD|SUB|MUL|DIV}.

**Return Format:** The query returns "ADD", "SUB", "MUL" or "DIV".

**Example:**

The following command sets the operator to plus.

```
:MATH:BASE:OPERator ADD
```

The query below returns "ADD".

```
:MATH:BASE:OPERator?
```

### **3.2.5. 5 :MATH:FFT**

#### **3.2.5.5.1 :MATH:FFT:SOURce**

**Function:** Select the source of FFT operation.

**Format:** :MATH:FFT:SOURce <source>

```
:MATH:FFT:SOURce?
```

Among them, <source>, discrete type, {CH1|CH2|CH3|CH4}.

**Return format :** The query returns "CH1", "CH2", "CH3" or "CH4".

**Example:**

The following command selects channel 1 as the source.

```
:MATH:FFT:SOURce CH1
```

The query below returns "CH1".

```
:MATH:FFT:SOURce?
```

### 3.2.5.5.2 :MATH:FFT:WINDow

**Function:** Select the window function for FFT operation.

**Format:** :MATH:FFT:WINDow <source>

:MATH:FFT:WINDow?

Among them, <source>, discrete type,  
{ RECTangle|HAMMing|BLACKman|HANNing }.

**Return format:** The query returns " RECTangle ", " HAMMing ", " BLACKman " or "HANNing".

**Example:**

The following command selects the HANNing window function.

:MATH:FFT:WINDow HANNing

The query below returns "HANNing".

:MATH:FFT:WINDow?

### 3.2.5.5.3 :MATH:FFT:TYPE

**Function:** Select the display mode of FFT waveform as “Linear” or “Logarithmic” .

**Format:** :MATH:FFT:TYPE <type>

:MATH:FFT:TYPE?

Among them, <type>, discrete type, {LINE|DB}.

**Return format:** The query returns "LINE" or "DB".

**Example:**

The following command selects logarithmic display mode.

```
:MATH:FFT:TYPE DB
```

The query below returns "DB".

```
:MATH:FFT:TYPE?
```

### 3.2.5.5.4 :MATH:FFT:VScale

**Function:** Set the vertical scale of the FFT operation result.

**Format:** :MATH:FFT:VScale <extent>

```
:MATH:FFT:VScale?
```

Among them, <extent>, real type, <extent>, real type, in line, {1e-15~5e14, can only be in steps of 1, 2, 5} or in db {1~500, 1, 2, 5 steps}.

**Return format:** The query returns the gear value in scientific notation.

**Example:**

The following command sets the vertical scale of the FFT operation result to 1.

```
:MATH:FFT:VScale 1
```

The following query returns "1.000000e+00".

:MATH:FFT:VScale?

### 3.2.5.5.5 :MATH:FFT:VPOsition

Function: Set the vertical offset of the FFT operation result.

Format: :MATH:FFT:VPOsition <position>

:MATH:FFT:VPOsition?

Among them, <positionoffset> is real type and expressed in scientific notation.

### 3.2.5.5.6 :MATH:FFT:HSCale

**Function:** Set the horizontal scale of FFT operation results.

**Format:** :MATH:FFT:HSCale <hscale>

:MATH:FFT:HSCale?

Among them, < hscale >, real type, {1Hz~100MHz, 1, 2, 5 steps}.

**Return Format:** The query returns the gear value in scientific notation.

#### **Example:**

The following command sets the horizontal scale of the FFT operation result to

1.

:MATH:FFT:HSCale 1

The following query returns "1.000000e+00".

```
:MATH:FFT:HSCale?
```

### 3.2.5.5.7 :MATH:FFT:HPOSITION

Function: Set the horizontal offset of the FFT operation result.

Format: :MATH:FFT:HPOSition <position >

```
:MATH:FFT:HPOSition?
```

Among them, <position>, real type,

**Return format:** The query returns the offset value in scientific notation.

#### Example:

The following command sets the horizontal offset to 2Hz.

```
:MATH:FFT:HPOSition 2
```

The query below returns "2.000000e0"

```
:MATH:FFT:HPOSition?
```

### 3.2.5.6 :MATH:AX+B

#### 3.2.5.6.1 :MATH:AX+B:SOURce

**Function:** Select the source of AX+B operation.

**Format:** :MATH:AX+B:SOURce <source>

:MATH:AX+B:SOURce?

Among them, <source>, discrete type, {CH1|CH2|CH3|CH4}.

**Return format:** The query returns " CH1 ", " CH2 ", " CH3 " or " CH4 ".

**Example:**

The following command selects channel 1 as the source.

:MATH:AX+B:SOURce CH1

The query below returns "CH1 ".

:MATH:AX+B:SOURce?

**3.2.5. 6.2 :MATH:AX+B:A**

**Function:** Select the value of A in AX+B.

**Format:** :MATH:AX+B:A <a>

:MATH:AX+B:A?

Among them, <a>, real type, please refer to the manual for the range.

**Return format:** The query returns a real value.

**Example:**

The following command sets the numerical value of A.

:MATH:AX+B:A 2

The query below returns "2".

:MATH:AX+B:A?

### 3.2.5. 6.3 :MATH:AX+B:B

**Function:** Select the value of B in AX+B.

**Format:** :MATH:AX+B:B <b>

:MATH:AX+B:B?

Among them, <b>, real type, please refer to the manual for the range.

**Return format:** The query returns a real value.

**Example:**

The following command sets the value of B.

:MATH:AX+B:B 100

The following query returns "100".

:MATH:AX+B:B?

### 3.2.5.6.4 :MATH:AX+B:UNIT <unit>

**Function:** Select the unit in AX+B.

**Format:** :MATH:AX+B:UNIT <unit>

:MATH:AX+B:UNIT?

Where <unit> is a string and its range is specified in the data sheet.

**Return Format:** The query returns a real value.

**Example:**

The following command math units.

:MATH:AX+B:UNIT W

The query below returns "W".

:MATH:AX+B:UNIT?

### 3.2.5.6.5 :MATH:AX+B:VScale

**Function:** Set the vertical scale of the operation result.

**Format:** :MATH:AX+B:VScale <extent>

:MATH:AX+B:EXTent?

Among them, <extent> is real type, {1e-15~5e14, can only be stepped by 1 , 2 , 5 }.

**Return Format:** The query returns the gear value in scientific notation.

**Example:** The following command sets the vertical scale of the logic operation result to 1.

```
:MATH:AX+B:VScale 1
```

The following query returns "1.000000e+00".

```
:MATH:AX+B:VScale?
```

### 3.2.5.6.6 :MATH:AX+B:VPOSton

**Function:** Set the vertical offset of the calculation result.

**Format:** :MATH:AX+B:VPOSton < position >

```
:MATH:AX+B:VPOSton?
```

Among them, < position > is real type and expressed in scientific notation.

### 3.2.5.7 :MATH:ADVanced

#### 3.2.5.7.1 :MATH:ADVanced:EXPRession

**Function:** Set the expression for advanced operations.

**Format:** :MATH:ADVanced:EXPRession <string>

```
:MATH:ADVanced:EXPRession?
```

Among them, <string> is an ASCII string.

**Return Format:** The query returns the current expression in string form.

**Example:**

The following command sets the expression to " CH1+CH2" .

```
:MATH:ADVanced:EXPRession CH1+CH2
```

The following query returns " CH1+CH2 " .

```
:MATH:ADVanced:EXPRession?
```

### 3.2.5.7.2 :MATH:ADVanced:VAR1

**Function:** Set the variable in the advanced operation expression 1.

**Format:** :MATH:ADVanced:VAR1 <value>

Where, <value> is real type, ranging from -9.9999E+9 to 9.9999E+9 . For the specific range, please refer to the data sheet.

**Return format:** The query returns the value of the current variable 1 in scientific notation .

**Example:** The following command sets the value of variable 1 to 100.

```
:MATH:ADVanced:VAR1 100
```

The query below returns " 1.000000e+02 " .

```
:MATH:ADVanced:VARiable1?
```

### 3.2.5. 7.3 :MATH:ADVanced:VAR2

**Function:** Set variable 2 in advanced operation expression

**Format:** :MATH:ADVanced:VAR2 <value>

Among them, <value> , real type,  $-9.9999E+9$  to  $9.9999E+9$  , please refer to the data sheet for the specific range.

**Return format:** The query returns the value of the current variable 2 in scientific notation .

**Example:** The following command sets the value of variable 2 to 100.

```
:MATH:ADVanced:VAR2 100
```

The query below returns " 1.000000e+02 " .

```
:MATH:ADVanced:VAR2?
```

#### 3.2.5.7.4 :MATH:ADVanced:VScale

**Function:** Set the vertical scale of advanced calculation results.

**Format:** :MATH:ADVanced:VScale <extent>

```
:MATH:ADVanced:VScale?
```

Among them, <extent> is real type,  $\{1e-15 \sim 5e14\}$ , can only be incremented by  $\{1, 2, 5\}$  .

**Return Format:** The query returns the gear value in scientific notation.

**Example:**

---

The following command sets the vertical scale of the advanced operation result to 1.

```
:MATH:ADVanced:VSCale 1
```

The query below returns " 1.000000e+00 ".

```
:MATH:ADVanced:VSCale?
```

#### **3.2.5.7.5 :MATH:ADVanced:VPOSiton**

Function: Set the vertical offset of advanced operation results.

Format: :MATH:ADVanced:VPOSiton <postion>

```
:MATH:ADVanced:VPOSiton?
```

Among them, <positon> , real type, expressed in scientific notation.

#### **3.2.5.7.6 :MATH:ADVanced:UNIT <unit>**

**Function:** Select units in ADVanced .

**Format:** :MATH:ADVanced:UNIT <unit>

```
:MATH:ADVanced:UNIT?
```

Among them, <unit>, string.

**Return format:** The query returns a real value .

**Example:**

---

The following command math units.

:MATH:ADVanced:UNIT W

The query below returns "W".

:MATH:ADVanced:UNIT?

### ***3.2.5.8 :MATH:SRATe?***

Query the sampling rate of mathematical waveforms, and the return value is expressed in scientific notation.

#### **Example:**

The query below returns " 2.500000e8 ".

:MATH:SRATe?

### ***3.2.5.9 :MATH:DEPth?***

Query the number of points of the math waveform. The returned value is expressed in scientific notation.

#### **Example:**

The following query returns " 7.000000e2 ".

:SAMPleACQuire:MATH:DEPth?

## **3.2.6 Cursor Command Subsystem**

### **3.2.6.1 :CURSor:HORizontal**

**Function:** Turn the horizontal cursor function on or off.

**Format:** :CURSor:HORizontal <bool>

:CURSor:HORizontal?

Among them , <bool>, Boolean type , {{0|OFF}}{1|ON}} .

### **3.2.6.2 :CURSor:VERTical**

**Function:** Turn the vertical cursor function on or off.

**Format:** :CURSor:VERTical <bool>

:CURSor:VERTical?

Among them , <bool>, Boolean type , {{0|OFF}}{1|ON}} .

### **3.2.6.3 :CURSor:CX1**

**Function:** Set the position of vertical cursor X1 .

**Format:** :CURSor:CX1 <px>

:CURSor:CX1?

Where <px> is an integer and is in pixels.

**Return format:** The query returns an integer.

**Example:**

---

The following command sets the horizontal position of the vertical cursor X1 to "100".

```
:CURSor:CX1 100
```

The query below translates "100".

```
:CURSor:CX1?
```

#### **3.2.6.4 :CURSor:CX2**

**Function:** Set the position of vertical cursor X2.

**Format:** :CURSor:CX2<px>

```
:CURSor:CX2?
```

Among them , <px>, integer , in pixels.

**Return format:** The query returns an integer.

**Example:**

The following command sets the horizontal position of the vertical cursor X2 to "100".

```
:CURSor:CX2 100
```

The query below translates "100".

```
:CURSor:CX2?
```

### **3.2.6.5 :CURSor:CY1**

**Function:** Set the position of horizontal cursor 1.

**Format:** :CURSor:CY1<px>

:CURSor:CY1?

Among them , <px>, integer , in pixels.

**Return format:** The query returns an integer.

**Example:**

The following command sets the vertical position of the horizontal cursor Y1 to "100".

:CURSor:CY1100

The query below translates "100".

:CURSor:CY1?

### **3.2.6.6 :CURSor:CY2**

**Function:** Set the position of horizontal cursor 2.

**Format:** :CURSor:CY2<px>

:CURSor:CY2?

Among them , <px>, integer , in pixels.

**Return format:** The query returns an integer.

**Example:**

The following command sets the vertical position of the horizontal cursor Y2 to "100".

```
:CURSor:CY2 100
```

The query below translates "100".

```
:CURSor:CY2?
```

**3.2.6.7 :CURSor:X1Value**

**Function:** Query the x value of vertical cursor X1.

**Format:** :CURSor:X1Value?

The units of the query value are determined by the current horizontal units.

**Return format:** The query returns the X value at the cursor X1 in scientific notation.

**Example:**

The query below returns "-0.000000e-02"

```
:CURSor:X1Value?
```

**3.2.6.8 :CURSor:X2Value**

**Function:** Query the x value of vertical cursor X2.

**Format:** :CURSor:X2Value?

The unit of the query value is determined by the current horizontal unit.

**Return Format:** The query returns the X value at cursor X2 in scientific notation.

**Example:**

The following query returns "-0.000000e-02"

:CURSor:X2Value?

### ***3.2.6.9 :CURSor:Y1Value***

**Function:** Query the y value of the horizontal cursor Y1.

**Format:** :CURSor:Y1Value?

The unit of the query value is determined by the current vertical unit.

**Return Format:** The query returns the Y value at cursor A in scientific notation.

**Example:**

The following query returns "-0.000000e-02"

:CURSor:YAValue?

### ***3.2.6.10 :CURSor:Y2Value***

**Function:** Query the y value of the horizontal cursor Y2.

**Format:** :CURSor:Y2Value?

The unit of the query value is determined by the vertical unit.

**Return Format:** The query returns the Y value at cursor B in scientific notation.

**Example:**

The following query returns "-0.000000e-02"

:CURSor:Y2Value?

### **3.2.6.11 :CURSor:XDELta**

**Function:** Query the difference between vertical cursors X1 and X2  $\Delta$ . The unit is the same as the horizontal unit.

**Format:** :CURSor:XDELta?

**Return format:** The query returns the current difference value X in scientific notation  $\Delta$ .

**Example:**

The query below returns "1.000000e-03".

:CURSor:XDELta?

### **3.2.6.12 :CURSor:YDELta**

**Function:** Query the difference between the horizontal cursor Y1 and Y2  $\Delta$ , the unit is the same as the vertical unit.

**Format:** :CURSor:YDELta?

**Return Format:** The query returns the current difference value X in scientific notation  $\Delta$

**Example:**

The following query returns "1.000000e-03".

:CURSor:YDELta?

### ***3.2.6.13 :CURSor:FREQ?***

**Function:** Query the 1/ between vertical cursors X1 and X2  $\Delta$ , in Hz.

**Format:** :CURSor:FREQ?

**Return Format:** The query returns the current value in scientific notation.

**Example:**

The following query returns "1.000000e 03".

:CURSor:FREQ?

### ***3.2.6.14 :CURSor:RATIo***

**Function: Query** the ratio between the difference between horizontal cursors A and B and the difference between vertical cursors A and B.

**Format:** :CURSor:RATIo?

**Return Format:** The query returns the value in scientific notation.

**Example:**

The following query returns "3.200000e-02".

```
:CURSor:RATio?
```

**3.2.6.15 :CURSor:SOURce**

**Function:** Set the channel source for cursor measurement.

**Format:** :CURSor:SOURce <source>

```
:CURSor:SOURce?
```

Among them, <source>, discrete type, {CH1|CH2|CH3|CH4| R1| R2| R3| R4|MATH |AUTO }.

**Return format:** The query returns "CH1", "CH2", "CH3", "CH4", "R1", "R2", "R3", "R4", or "MATH".

**Example:**

The following command sets channel 1 as the measurement source.

```
:CURSor:SOURce CH1
```

The query below returns "CH1".

```
:CURSor:SOURce?
```

### 3.2.6.16 phase cursor

3.2.6.16.1 :PHCursor [< bool >, < src1> , < src2>]

**Function:** Turn on/off/query the phase cursor and set related parameters.

**Format:** :PHCursor [< bool >, < src1> , < src2>]

:PHCursor?

Among them, <bool> , <bool>, Boolean type, {{0|OFF}}{1|ON}}; <src1> , integer, represents the number of cylinders to be set ; <src2> , integer, represents the set angle, which is generally a multiple of 1 80 ;

**Return format:** The query returns the corresponding parameters.

**Example:**

The following command turns on the phase cursor and sets the cursor to 4 cylinders, 720 degrees.

```
:PHCursor 1 , 4 , 720
```

The query below returns "1 , 4,720".

```
:PHCursor?
```

3.2.6.16.2 :PHCursor:X0 <px> ,

**Function:** Set or query the pixel position of the 0-degree cursor line.

**Format:** :PHCursor:X0 <px>

:PHCUrsor:X0?

Where, < px > is an integer, indicating the pixel position of the 0 degree cursor line on the screen, with the left side of the screen as the reference;

**Return Format:** The query returns the pixel position of the 0 degree cursor line on the screen.

**Example:**

The following command sets the position of the 0 degree cursor line to 100 pixels .

:PHCUrsor:X0 100

The following query returns " 100 " .

:PHCUrsor:X0?

3.2.6.16.3 :PHCUrsor:ZN <px> ,

**Function:** Set or query the pixel position of the last cursor line.

**Format:** :PHCUrsor:ZN <px>

:PHCUrsor:ZN?

Where, < px > is an integer, indicating the pixel position of the last cursor line, based on the left side of the screen ;

**Return format:** The query returns the pixel position of the last cursor line on the screen .

**Example:**

The following command sets the position of the last cursor line to 200 pixels.

```
:PHCursor: XN 200
```

The query below returns "200".

### 3.2.7 Display command subsystem

#### 3.2.7.1 :DISPlay:WAVeform

**Function:** Set the display mode of the waveform on the screen, "point display" or "line display".

**Format:** :DISPlay:WAVeform <type>

```
:DISPlay:WAVeform?
```

Among them, <type>, discrete type, { VECTors|DOTS }.

**Return format:** The query returns " VECTors " or "DOTS".

**Example:**

The following command sets the waveform display mode to "DOTS".

```
:DISPlay:WAVeform DOTS
```

The query below returns "DOTS".

```
:DISPlay:WAVeform?
```

### ***3.2.7.2 :DISPlay:BRIGhtness***

**Function:** Set the brightness of the waveform display on the screen.

**Format:** :DISPlay:BRIGhtness <time>

```
:DISPlay:BRIGhtness?
```

Among them, <time>, integer, 0 to 100.

**Return format:** The query returns an integer.

#### **Example:**

The following command sets the brightness of the waveform display to 80.

```
:DISPlay:BRIGhtness 80
```

The query below returns "80".

```
:DISPlay:BRIGhtness?
```

### ***3.2.7.3 :DISPlay:GRATicule***

**Function:** Set the grid type displayed on the screen.

**Format:** :DISPlay:GRATicule <type>

```
:DISPlay:GRATicule?
```

Among them, <type>, discrete type, { FULL|GRID|RETical|FRAME }.

**Return format:** query returns "FULL", "GRID", " RETical " or " FRAME ".

**Example:**

The following command yarns the screen grid type to FULL.

```
:DISPlay:GRATicule FULL
```

The query below returns "FULL".

```
:DISPlay:GRATicule?
```

### ***3.2.7.4 :DISPlay:INTensity***

**Function:** Set the brightness of the grid display on the screen.

**Format:** :DISPlay:INTensity <time>

```
:DISPlay:INTensity?
```

Among them, <time>, integer, 0 to 100.

**Return format:** The query returns an integer.

**Example:**

The following command sets the brightness of the screen grid to 80.

```
:DISPlay:INTensity 80
```

The query below returns "80".

:DISPlay:INTEnsity?

### **3.2.7.5 :DISPlay:PERsist**

#### **3.2.7.5.1 :DISPlay:PERsist:MODE**

**Function:** Set persistence display mode.

**Format:** :DISPlay:PERsist:MODE <mode>

:DISPlay:PERsist:MODE?

Where <mode> is discrete, { AUTO|NORMal|INFinite|none }.

#### **3.2.7.5.2 :DISPlay:PERsist:ADJust**

**Function:** Set the persistence time in normal display mode

**Format:** :DISPlay:PERsist:ADJust <time>

:DISPlay:PERsist:ADJust?

Where <time> is an integer in milliseconds , 100, 200, 300, 400, 500, 600, 700, 800, 900, 1000, 2000, 3000, 4000, 5000, 6000, 7000, 8000, 9000, 10000

#### **3.2.7.5.3 :DISPlay:PERsist:CLEar**

**Function:** Clear the afterglow display

**Format:** :DISPlay:PERsist:CLEar

### **3.2.7.6 :DISPlay:HIGH (valid in machines with independent high refresh mode)**

**Function:** Turn high refresh rate on or off

**Format:** :DISPlay:HIGH <bool>

:DISPlay:HIGH?

Among them, <bool>, Boolean type, {{0|OFF}}{1|ON}}.

### ***3.2.7.7 :DISPlay:HORRef***

**Function:** Set the screen horizontal expansion center mode, "trigger point" or "screen center".

**Format:** :DISPlay:HORRef <mode>

:DISPlay:HORRef?

Where <mode> is discrete and is { CENTER|TRIGpos }.

### ***3.2.7.8 :DISPlay:ZOOM***

Function: Open or close ZOOM

Format: :DISPlay:ZOOM <bool>

:DISPlay:ZOOM?

Among them, <bool>, Boolean type, {{0|OFF}}{1|ON}}.

### ***3.2.7.9 :DISPlay:CCT***

Function: Turn on or off color temperature display

Format: :DISPlay:CCT <bool>

:DISPlay:CCT?

Among them, <bool>, Boolean type, {{0|OFF}}{1|ON}}.

### 3.2.8 Measurement command subsystem

#### 3.2.8.1 :MEASure:OPEN [<item>,< n1> , < n2> ,< src1> ,< src2>]

**Function:** Add measurement items of specified channels on the interface

**Format:** :MEASure:OPEN [<item>,< n1> , < n2> ,< src1> ,< src2>]

Note:<item> is discrete, referring to the measurement item, { PERiod | FREQ | RISE time| FALL time |DELAY| PDUTy| NDUTy| PWIDth| NWIDth |BURStw|ROV|FOV|PHASe|PKPK|AMP|HIGH|LOW|MAX|MIN|RMS|CRMS|MEAN|CM EAn |ACRMS|+RATE|-RATE}.

<n1> refers to the source, discrete type, {CH1|CH2|CH3|CH4|R1|R2|R3|R4| MATH }.

<n2> refers to the source, valid when DELay and PHAS e , discrete type, {CH1|CH2|CH3|CH4|R1|R2|R3|R4| MATH}.

<src1> refers to the parameters of some measurement items, valid when DELay , discrete type, [ FRISe|FFALL|LRISe| LFALL ] .

<src2> refers to the parameters of some measurement items, valid when DELay , discrete type, [ FRISe|FFALL|LRISe| LFALL ] .

**Example:**

The following settings turn on the period measurement for channel 1 on the screen

```
:MEASure:OPEN PERiod , CH1
```

the first rising edge of channel 2 to the first rising edge of channel 3 on the screen.

```
:MEASure:OPEN DELay CH2,CH3,FRISe,FRISe
```

### 3.2.8.2 *:MEASure:<item>? [<n 1> , <n 2> ,<src1> ,<src2>]*

**Function:** Query the value of the open measurement item

**Format:** :MEASure:<item>? [<n 1> , <n 2> ,<src1> ,<src2>]

Note:<item> discrete type refers to the measurement item, { PEROID | FREQ | RISE time | FALL time | DELAY | PDUTY | NDUTY | PWIDth | NWIDth | BURStw | ROV | FOV | PHASe | PKPK | AMP | HIGH | LOW | MAX | MIN | RMS | CRMS | MEAN | CMEAN | ACRMS | +RATE | -RATE }.

<n1> refers to the source, discrete type, { CH1 | CH2 | CH3 | CH4 | R1 | R2 | R3 | R4 | MATH }.

<n2> refers to the source, which is valid when DELay and PHAS e , and invalid at other times . It does not need to be written, discrete type, { CH1 | CH2 | CH3 | CH4 | R1 | R2 | R3 | R4 | MATH }.

<src1> refers to the parameters of some measurement items, valid when DELAY , discrete type, [ RISE|FFALL|LRISe| LFALL ] .

<src2> refers to the parameters of some measurement items, which is valid when DELAY , discrete type, [ RISE|FFALL|LRISe| LFALL ] .

**Example:** When the period measurement of channel 1 is turned on, the following settings query the period measurement value of channel 1

```
:MEASure:PERiod? CH1
```

### 3.2.8.3 :MEASure:CLOSe [<item>,< n1> , < n2> ,< src1> ,< src2>]

**Function:** Delete the measurement items of the specified channel in the open state on the interface

**Format:** :MEASure:CLOS e [<item>,< n1> , < n2> ,< src1> ,< src2>]

Note:<item> is discrete, referring to the measurement item, { PEROid| FREQ | RISE time| FALL time |DELAy| PDUTy| NDUTy| PWIDth| NWIDth |BURStw|ROV|FOV|PHASe|PKPK|AMP|HIGH|LOW|MAX|MIN|RMS|CRMS|MEAN|CM EAn |ACRMS |+RATE|-RATE }.

<n1> refers to the source, discrete type, {CH1|CH2|CH3|CH4|R1|R2|R3|R4| MATH }.

<n2> refers to the source, which is valid when DELAY and PHASe , discrete type, {CH1|CH2|CH3|CH4|R1|R2|R3|R4|MATH}.

<src1> refers to the parameters of some measurement items, which is valid when DELAY , discrete type, [ FRISe|FFALL|LRISe| LFALL ] .

< src2> refers to the parameters of some measurement items, which is valid when DELAY , discrete type, [ FRISe|FFALL|LRISe| LFALL ] .

**Example:**

The following settings turn off the period measurement of channel 1 on the screen.

:MEASure:CLOSe PERiod , CH1

**3.2.8. 4 :MEASure:CLEAr <item0|item1.....|item10| all>**

**Function:** Clear all items in the open measurement items.

**Format:** :MEASure:CLEAr <item>

Among them, <item> is discrete, { item1| item 2| item 3| item 4| item 5| item 6| item 7| item 8|item9| item 10| all }.

1~10 correspond to the 10 measurement options on the screen .

**3.2.8.5 :MESAure:STATistic**

**3.2.8.5.1 :MEASure:STATistic:DISPlay**

**Function:** Turn the statistics function on or off.

**Format:** :MEASure:STATistic:DISPlay <bool>

:MEASure:STATistic:DISPlay?

Among them, <bool> is a Boolean type, {{0|OFF}}{1|ON}} .

#### **3.2.8.5.2 :MEASure:STATistic:RESet**

**Function:** Clear historical statistics and re-count.

**Format:** :MEASure:STATistic:RESet

#### **3.2.8.5.3 :MEASure:STATistic:MEAN <bool>**

**Function:** Turn on or off the average value display in statistics

**Format:** :MEASure:STATistic:MEAN <bool>

:MEASure:STATistic:MEAN?

Among them, <bool>, Boolean type, {{0|OFF}}{1|ON}}.

#### **3.2.8.5.4 :MEASure:STATistic:MAX <bool>**

**Function:** Turn on or off the maximum value display in statistics

**Format:** :MEASure:STATistic:MAX <bool>

:MEASure:STATistic:MAX?

Among them, <bool>, Boolean type, {{0|OFF}}{1|ON}}.

#### **3.2.8.5.5 :MEASure:STATistic:MIN <bool>**

**Function:** Turn on or off the minimum value display in statistics

**Format:** :MEASure:STATistic:MIN <bool>

:MEASure:STATistic:MIN?

Among them, <bool>, Boolean type, {{0|OFF}}{1|ON}}.

### 3.2.8.5.6 :MEASure:STATistic:DEV <bool>

**Function:** Turn on or off the mean square error display in statistics

**Format:** :MEASure:STATistic:DEV <bool>

:MEASure:STATistic:DEV?

Among them, <bool>, Boolean type, {{0|OFF}}{1|ON}}.

### 3.2.8.5.7 :MEASure:STATistic:COUNt <bool>

**Function:** Turn on or off the count display in statistics

**Format:** :MEASure:STATistic:COUNt <bool>

:MEASure:STATistic:COUNt?

Among them, <bool>, Boolean type, {{0|OFF}}{1|ON}}.

### 3.2.8.5.8 :MEASure:STATistic:VIEW?

**Function:** Query all values of statistical items (valid when the statistical function is turned on)

**Format:** :MEASure:STATistic:VIEW? <item> , <source>

Among them, <item> , the measurement item that has been opened, <source>, discrete type, measurement source {CH1|CH2|CH3|CH4|R1|R2|R3|R4|MATH}.

**Note:** <source> can be omitted, and the default is the channel source currently specified by the oscilloscope.

**Return format:** The query returned values are displayed in scientific notation, followed by current value, average value, maximum value, minimum value, root mean square, and count.

**For example:** The following query returns the statistical data of the peak-to-peak measurement value of channel 1, such as: 1.00000 7 e-02, 1.00000 5 e-02, 1.00000 9 e-02, 1.00000 1 e-02, 1.000000e-02, 1 .75e02 ,

```
:MEASure:STATistic:VIEW? PKPK, CH1
```

If the current measurement source set by the oscilloscope is also channel 1, use the following command directly:

```
:MEASure:STATistic:VIEW? PKPK
```

### 3.2.8.5.9 :MEASure:STATistic:MEAN:VIEW?

**Function:** Query the average value of statistical items (valid when the statistics function is turned on)

**Format:** :MEASure:STATistic:MEAN:VIEW? <item> , <source>

Among them, <item> , the measurement item that has been opened, <source>, discrete type, measurement source {CH1|CH2|CH3|CH4|R1|R2|R3|R4|MATH}.

**Note:** <source> can be omitted, and the default is the channel source currently specified by the oscilloscope.

**Return format:** The value returned by the query is displayed in scientific notation.

**Example:** The following query returns the statistical calculation of the peak-to-peak measurement value of channel 1. Average value, such as: 1.000007 e-02

```
:MEASure:STATistic:MEAN:VIEW? PKPK, CH1
```

If the current measurement source set by the oscilloscope is also channel 1, use the following command directly:

```
:MEASure:STATistic:MEAN:VIEW? PKPK,
```

### 3.2.8.5.10 :MEASure:STATistic:MAX:VIEW?

**Function:** Query the maximum value of statistical items (valid when the statistics function is turned on)

**Format:** :MEASure:STATistic:MAX:VIEW? <item> , <source>

Among them, <item> , the measurement item that has been opened, <source>, discrete type, measurement source {CH1|CH2|CH3|CH4|R1|R2|R3|R4|MATH}.

**Note:** <source> can be omitted, and the default is the channel source currently specified by the oscilloscope.

**Return format:** The value returned by the query is displayed in scientific notation.

**Example:** The following query returns the statistical calculation of the peak-to-peak measurement value of channel 1. Maximum value, such as:1.000007 e-02

```
:MEASure:STATistic:MAX:VIEW? PKPK, CH1
```

If the current measurement source set by the oscilloscope is also channel 1, use the following command directly:

```
:MEASure:STATistic:MAX:VIEW? PKPK,
```

#### 3.2.8.5.11 :MEASure:STATistic:MIN:VIEW?

**Function:** Query the minimum value of statistical items (valid when the statistical function is turned on)

**Format:** :MEASure:STATistic:MIN:VIEW? <item> , <source>

Among them, <item> , the measurement item that has been opened, <source>, discrete type, measurement source {CH1|CH2|CH3|CH4|R1|R2|R3|R4|MATH}.

**Note:** <source> can be omitted, and the default is the channel source currently specified by the oscilloscope.

**Return format:** The value returned by the query is displayed in scientific notation.

**Example:** The following query returns the statistical calculation of the peak-to-peak measurement value of channel 1. Minimum value, such as: 1.000007e-02

```
:MEASure:STATistic:MIN:VIEW? PKPK, CH1
```

If the current measurement source set by the oscilloscope is also channel 1, use the following command directly:

```
:MEASure:STATistic:MIN:VIEW? PKPK,
```

### 3.2.8.5.12 :MEASure:STATistic:DAV:VIEW?

**Function:** Query the mean square error of statistical items (valid when the statistical function is turned on)

**Format:** :MEASure:STATistic:DAV:VIEW? <item >, < source >

Among them, <item> is the measurement item that has been turned on, <source> is the discrete type, and the measurement source is {CH1|CH2|CH3|CH4|R1|R2|R3|R4|MATH}.

**Note:** <source> can be omitted and the default is the channel source currently specified by the oscilloscope.

**Return Format:** The query return value is displayed in scientific notation.

**For example:** The following query returns the statistical calculation of the peak-to-peak measurement value of channel 1 Mean square error, e.g. 1.000007 e-02

:MEASure:STATistic:DAV:VIEW? PKPK, CH1

If the measurement source of the current oscilloscope setting is also channel 1, use the following command directly:

:MEASure:STATistic:DAV:VIEW? PKPK,

### 3.2.8.5.13 :MEASure:STATistic:COUNt:VIEW?

**Function:** Query the statistical quantity of statistical items (valid when the statistics function is turned on)

**Format:** :MEASure:STATistic:COUNt:VIEW? <item> , <source>

Among them, <item> , the measurement item that has been opened, <source>, discrete type, measurement source {CH1|CH2|CH3|CH4|R1|R2|R3|R4|MATH}.

**Note:** <source> can be omitted and the default is the channel source currently specified by the oscilloscope.

**Return Format:** The query return value is displayed in scientific notation.

**For example:** The following query returns the statistical calculation of the peak-to-peak measurement value of channel 1 Statistical quantity, such as:1.000007e-02

:MEASure:STATistic:COUNt:VIEW? PKPK, CH1

If the measurement source of the current oscilloscope setting is also channel 1, use the following command directly:

:MEASure:STATistic:COUNt:VIEW? PKPK,

### 3.2.8.5.14 :MEASure:STATistic:CURRent:VIEW?

**Function:** Query the current value of statistical items (valid when the statistical function is turned on)

**Format:** :MEASure:STATistic:CURRent:VIEW? <item> , <source>

Among them, <item> , the measurement item that has been opened, <source>, discrete type, measurement source {CH1|CH2|CH3|CH4|R1|R2|R3|R4|MATH}.

**Note:** <source> can be omitted, and the default is the channel source currently specified by the oscilloscope.

**Return format:** The value returned by the query is displayed in scientific notation.

**Example:** The following query returns the statistical calculation of the peak-to-peak measurement value of channel 1. Average value, such as:1.000007e-02

:MEASure:STATistic:CURRent:VIEW? PKPK, CH1

If the current oscilloscope measurement source is also channel 1, use the following command directly:

:MEASure:STATistic:CURRent:VIEW? PKPK,

### **3.2.8.6 :MEASure:ADISplay**

**Function:** Turn all measurements on or off.

**Format:** :MEASure:ADISplay <bool>

:MEASure:ADISplay?

Among them, <bool>, Boolean type, {{0|OFF}}{1|ON}}.

### **3.2.8.7 :MEASure:COUNter**

#### **3.2.8.7.1 :MEASure:COUNter:SOURce**

**Function:** Set or query the measurement source of the counter.

:MEASure:COUNter:SOURce <sour>

:MEASure:COUNter:SOURce?

Where, <sour> is discrete, {CLOS e| CH1|CH2|CH3|CH4} .

#### **3.2.8.7.2 :MEASure:COUNter:MODE <mode>**

#### **3.2.8.7.3 :MEASure:COUNter:VALue?**

**Function:** Query the measurement result of the counter.

:MEASure:COUNter:VALue?

The query returns the current measurement in scientific notation. If the frequency counter function is not currently turned on, 0.0000000e+00 will be returned.

### 3.2.9 Trigger command subsystem

#### 3.2.9.1 :TRIGger:TYPE

**Function:** Select trigger type.

**Format:** :TRIGger:TYPE <type>

:TRIGger:TYPE?

Among them, <type>, discrete type,  
{EDGE|PULSe|LOGic|NEDGE|DWARt|SLOPe|TIMEout|VIDeo|S1|S2}

**Return format:**The query returns the currently used trigger type.

#### **Example:**

The following command selects edge triggering.

```
:TRIGger:TYPE EDGE
```

The query below returns "EDGE".

```
:TRIGger:TYPE?
```

#### 3.2.9.2 :TRIGger:HOLDoff

**Function:** Set trigger holdoff time .

**Format:** :TRIGger:HOLDoff <value>

:TRIGger:HOLDoff?

Where <value> is a real number ranging from 200ns to 10s.

**Return Format:** The query returns the trigger holdoff time in scientific notation.

**Example:**

The following command sets the trigger holdoff time to 200ns.

```
:TRIGger:HOLDoff 0.0000002
```

The following query returns "2.000000e-07".

```
:TRIGger:HOLDoff?
```

### 3.2.9.3 TRIGger:MODE

**Function:** Set the trigger mode:automatic or normal.

**Format:** :TRIGger:MODE <mode>

```
:TRIGger:MODE?
```

Where <mode> is discrete and is { AUTO|NORMal }.

**Return Format:** The query returns "AUTO" or "NORMal".

**Example:**

The following command selects the automatic trigger mode.

:TRIGGER:MODE AUTO

The following query returns "AUTO".

:TRIGger:MODE?

#### **3.2.9.4 :TRIGger:STATus**

**Function:** Query the current trigger status.

**Format:** :TRIGger:STATus?

**Return format:** The query returns "RUN", "WAIT", "AUTO", and "STOP".

#### **3.2.9.5 :TRIGger:EDGE**

##### **3.2.9.5.1 :TRIGger:EDGE:SOURce**

**Function:** Select the trigger source of edge trigger.

**Format:** :TRIGger:EDGE:SOURce <source>

:TRIGger:EDGE:SOURce?

Among them, <source>, discrete type, {CH1|CH2|CH3|CH4}.

**Return format:** The query returns "CH1", "CH2", "CH3" or "CH4".

#### **Example:**

The following command sets channel 1 as the trigger source.

:TRIGger:EDGE:SOURce CH1

The query below returns "CH1".

```
:TRIGger:EDGE:SOURce?
```

### 3.2.9.5.2 :TRIGger:EDGE:SLOPe

**Function:** Select the edge type of edge trigger.

**Format:** :TRIGger:EDGE:SLOPe <edge>

```
:TRIGger:EDGE:SLOPe?
```

Among them, <edge>, discrete type, {RISE|FALL|DUAL}.

**Return format:** The query returns "RISE", "FALL" or "DUAL".

#### Example:

The following command selects rising edge triggering.

```
:TRIGger:EDGE:SLOPe RISE
```

The following query returns "RISE".

```
:TRIGger:EDGE:SLOPe?
```

### 3.2.9.5.3 :TRIGger:EDGE:LEVel

**Function:** Set the trigger level when edge triggering

**Format:** :TRIGger:EDGE:LEVel <level>

```
:TRIGger:EDGE:LEVel?
```

Among them, <level> is a real type.

**Return Format:** The query returns the trigger level value in scientific notation.

**Example:**

The following command sets the trigger level to 150mV.

```
:TRIGger:EDGE:LEVel 0.15
```

The following query returns "1.500000e-01".

```
:TRIGger:EDGE:LEVel?
```

### 3.2.9.5.4 :TRIGger:EDGE:COUPle

**Function:** Set the edge trigger coupling mode.

**Format:** :TRIGger:EDGE:COUPle <couple>

```
:TRIGger:EDGE:COUPle?
```

Where, <couple> is discrete, { DC|AC|HFRej|LFRej|Noiserej }.

**Return format:** The query returns "DC", "AC", " HFRej ", " LFRej " or " Noiserej ".

**Example:**

The following command selects DC coupling mode.

```
:TRIGger:EDGE:COUPle DC
```

The following query returns "DC".

:TRIGger:EDGE:COUPlE?

### 3.2.9.6 :TRIGger:PULSe

#### 3.2.9.6.1 :TRIGger:PULSe:SOURce

**Function:** Set the trigger source of pulse width trigger.

**Format:** :TRIGger:PULSe:SOURce <source>

:TRIGger:PULSe:SOURce

Among them, <source>, discrete type, {CH1|CH2|CH3|CH4}.

**Return format:** The query returns "CH1", "CH2", "CH3" or "CH4".

#### **Example:**

The following command sets channel 1 as the trigger source.

```
:TRIGger:PULSe:SOURce CH1
```

The query below returns "CH1".

```
:TRIGger:PULSe:SOURce?
```

#### 3.2.9.6.2 :TRIGger:PULSe:POLarity

**Function:** Set the polarity of pulse width triggering.

**Format:** :TRIGger:PULSe:POLarity <polarity>

:TRIGger:PULSe:POLarity?

Among them, <polarity>, discrete type, { POSitive|NEGative }.

**Return format:** The query returns "POSitive " or "NEGative ".

**Example:**

The following command selects rising edge triggering.

```
:TRIGger:PULSe:POLarity POSitive
```

The following query returns " POSitive ".

```
:TRIGger:PULSe:POLarity?
```

### 3.2.9.6.3 :TRIGger:PULSe:WIDTh

**Function:** Set the pulse width value during pulse width triggering.

**Format:** :TRIGger:PULSe:WIDTh <width>

```
:TRIGger:PULSe:WIDTh?
```

Among them, <width>, real type, 40ns to 10s.

**Return format:** The query returns real numbers.

**Example:**

The following command sets the pulse width value to 4ns.

```
:TRIGger:PULSe:WIDTh 4.000000e-08
```

The query below returns "4.000000e-08".

:TRIGger:PULSe:WIDTh?

#### 3.2.9.6.4 :TRIGger:PULSe:CONDition

**Function:** Set pulse width trigger conditions.

**Format:** :TRIGger:PULSe:CONDition <condition>

:TRIGger:PULSe:CONDition?

Among them, <condition>, discrete type, { GREat|LESS|EQUal|UNEQual }.

GREat: The oscilloscope input signal pulse width is greater than the specified pulse width ;

LESS: The oscilloscope input signal pulse width is less than the specified pulse width;

EQUal: The oscilloscope input signal pulse width is equal to the specified pulse width;

UNEQual: The oscilloscope input signal pulse width is not equal to the specified pulse width;

#### 3.2.9.6.5 :TRIGger:PULSe:LEVel

**Function:** Set the trigger level for pulse width triggering

**Format:** :TRIGger:PULSe:LEVel <level>

:TRIGger:PULSe:LEVel?

Among them, <level> is a real type.

**Return Format:** The query returns the trigger level value in scientific notation.

**Example:**

The following command sets the trigger level to 150mV.

```
:TRIGger:PULSe:LEVel 0.15
```

The following query returns "1.500000e-01".

```
:TRIGger:PULSe:LEVel?
```

### ***3.2.9.7 :TRIGger:LOGic***

#### **3.2.9.7.1 :TRIGger:LOGic:STATus**

**Function:** Set the logic state of each channel in the logic trigger

**Format:** :TRIGger:LOGic:STATus <channel>,<status>

```
:TRIGger:LOGic:STATus? <channel>
```

Among them, <channel>, discrete type, {CH1|CH2|CH3|CH4}. <status>, discrete type, {HIGH|LOW| NONE}.

#### **3.2.9.7.2 :TRIGger:LOGic:FUNction**

**Function:** Set the comparison function of the logic trigger.

**Format:** :TRIGger:LOGic:FUNction <function>

:TRIGger:LOGic:FUNcTion?

Where <function> is a discrete type, and can be “AND” , “OR” , “NAND” or “NOR” .

### 3.2.9.7.3 :TRIGger:LOGic:CONDition

**Function:** Set the logic trigger condition.

**Format:** :TRIGger:LOGic:CONDition <condition>

:TRIGger:LOGic:CONDition?

Where <condition> is discrete, { GREat|LESS|EQUal|UNEQual|TRUE|FALSe }.

GREat: Triggered when the logic state is true for a longer time than the trigger logic time ;

LESS: Triggered when the logic state is true for a shorter time than the trigger logic time;

EQUal: Trigger when the logic state is true and the hold time is equal to the trigger logic time;

UNEQual: Triggered when the holding time of the logic state is true is not equal to the trigger logic time;

TRUL: Triggered when the logical state is true;

FALSe: Triggered when the logical state is false.

#### 3.2.9.7.4 :TRIGger:LOGic:TIME

**Function:** Set the trigger logic time.

**Format:** :TRIGger:LOGic:TIME <time>

:TRIGger:LOGic:TIME?

Where, <time> is a real type ranging from 200ns to 10s.

#### 3.2.9.7.5 :TRIGger:LOGic:LEVel

**Function:** Set the trigger level of each channel during logic triggering

**Format:** :TRIGger:LOGic:LEVel <channel>,<level>

:TRIGger:LOGic:LEVel? <channel>

Among them, <channel> is discrete type, {CH1|CH2|CH3|CH4}; <level> is real type.

#### 3.2.9.8 :TRIGger:Runt

##### 3.2.9.8.1 :TRIGger:Runt:SOURce

Function: Set the trigger source of the runt trigger .

Format: :TRIGger:Runt:SOURce <source>

:TRIGger:Runt:SOURce?

Where <source> is discrete, {CH1|CH2|CH3|CH4}.

##### 3.2.9.8.2 :TRIGger:Runt:POLARity

Function: Set the pulse polarity of runt trigger .

Format: :TRIGger:Runt:POLArity <polarity>

:TRIGger:Runt:POLArity?

Among them, <polarity>, discrete type, { POSItive|NEGAtive|EITHer }.

### **3.2.9.8.3 :TRIGger:Runt:CONDition**

Function: Set the pulse width limit condition.

Format: :TRIGger:Runt:CONDition <condition>

:TRIGger:Runt:CONDition?

Among them, <condition> is discrete type, { GREAt|LESS|BETWEEen|NONE }.

GREAt: The pulse width of the oscilloscope input signal is greater than the specified pulse width;

LESS: The pulse width of the oscilloscope input signal is less than the specified pulse width;

BETWEEen: The pulse width of the oscilloscope input signal is between the specified pulse widths;

NONE: irrelevant;

### **3.2.9.8.4 :TRIGger:Runt:HTIME**

Function: Set the upper limit time of runt trigger.

---

Format: :TRIGger:Runt:HTIME <time>

:TRIGger:Runt:HTIME?

Where <time> is a real type ranging from 8ns to 10s.

#### **3.2.9.8.5 :TRIGger:Runt:LTIME**

Function: Set the lower limit of the runt trigger time.

Format: :TRIGger:Runt:LTIME <time>

:TRIGger:Runt:LTIME?

Where <time> is a real type ranging from 8ns to 10s.

#### **3.2.9.8.6 :TRIGger:Runt:BTIME**

Function: Set the time interval for runt trigger.

Format: :TRIGger:Runt:BTIME < htime >, < ltime >

:TRIGger:Runt:BTIME? <type>

Among them, < htime >, < ltime >, real type, 8ns to 10s. (high>low)

< type >, discrete type , { HIGH|LOW }

#### **3.2.9.8.7 :TRIGger:Runt:HLEVEL**

Function: Set the high level during runt trigger .

Format: :TRIGger:Runt:HLEVEL <level>

:TRIGger:Runt:HLEVEL?

Among them, <level> is a real type.

### **3.2.9.8.8 :TRIGger:Runt:LLEVEL**

Function: Set the low level when the runt triggers .

Format: :TRIGger:Runt:LLEVEL <level>

:TRIGger:Runt:LLEVEL?

Among them, <level> is real type. ( H LEV el > LLEV el )

### **3.2.9.9 :TRIGger:SLOPe**

#### **3.2.9.9.1 :TRIGger:SLOPe:SOURce**

Function: Set the trigger source of slope trigger.

Format: :TRIGger:SLOPe:SOURce <source>

:TRIGger:SLOPe:SOURce?

Among them, <source>, discrete type, {CH1|CH2|CH3|CH4}.

#### **3.2.9.9.2 :TRIGger:SLOPe:EDGE**

Function :Set the slope trigger edge.

Format: :TRIGger:SLOPe:EDGE <edge>

:TRIGger:SLOPe:EDGE?

Where <edge> is discrete, { RISE|FALL|EITHER }.

### **3.2.9.9.3 :TRIGger:SLOPe:CONDition**

Function: Set the limit conditions for slope triggering.

Format: :TRIGger:SLOPe:CONDition <condition>

:TRIGger:SLOPe:CONDition?

Among them, <condition> is discrete, { GREat|LESS|BETWeen }.

GREat: The oscilloscope input signal slope is greater than the specified time setting ;

LESS: The oscilloscope input signal slope is less than the specified time setting;

BETWeen: The slope of the oscilloscope input signal is greater than the specified upper time limit and less than the specified lower time limit.

### **3.2.9.9.4 :TRIGger:SLOPe:HTIME**

Function: Set the upper limit of the slope trigger time.

Format: :TRIGger:SLOPe:HTIME <time>

:TRIGger:SLOPe:HTIME?

Where <time> is a real type ranging from 8ns to 10s.

### **3.2.9.9.5 :TRIGger:SLOPe:LTIME**

Function: Set the lower limit of the slope trigger time.

---

Format: :TRIGger:SLOPe:LTIMe <time>

:TRIGger:SLOPe:LTIMe?

Where <time> is a real type ranging from 8ns to 10s.

#### **3.2.9.9.6 :TRIGger:SLOPe:HLEVel**

Function: Set the high level when the slope is triggered.

Format: :TRIGger:SLOPe:HLEVel <level>

:TRIGger:SLOPe:HLEVel?

Among them, <level> is real type.

#### **3.2.9.9.7 :TRIGger:SLOPe:LLEVel**

Function: Set the low level when the slope triggers.

Format: :TRIGger:SLOPe:LLEVel <level>

:TRIGger:SLOPe:LLEVel?

Among them, <level> is real type. ( HLEV el > LLEV el )

#### **3.2.9.10 :TRIGger:TIMeout**

##### **3.2.9.10.1 :TRIGger:TIMeout:SOURce**

Function: Set the trigger source of timeout trigger.

Format: :TRIGger:TIMeout:SOURce <source>

:TRIGger:TIMEout:SOURce?

Among them, <source>, discrete type, {CH1|CH2|CH3|CH4}

#### **3.2.9.10.2 :TRIGger:TIMEout:POLarity**

Function: Set timeout trigger polarity.

Format: :TRIGger:TIMEout:POLarity <polarity>

:TRIGger:TIMEout:POLarity?

Among them, < polarity >, discrete type, { POSitive|NEGative|EITHer }.

#### **3.2.9.10.3 :TRIGger:TIMEout:TIME**

Function: Set the timeout period for timeout trigger.

Format: :TRIGger:TIMEout:TIME <time>

:TRIGger:TIMEout:TIME?

Among them, <time>, real type, 8ns to 10s.

#### **3.2.9.10.4 :TRIGger:TIMEout:LEVel**

Function: Set the over-trigger level for time-out trigger.

Format: :TRIGger:TIMEout:LEVel <level>

:TRIGger:TIMEout:LEVel?

Among them, <level>, real type, and range refer to the datasheet.

### **3.2.9.11 :TRIGger:NEDGe**

#### **3.2.9.11.1 :TRIGger:NEDGe:SOURce**

Function: Set the trigger source of Nth edge trigger.

Format: :TRIGger:NEDGe:SOURce <source>

:TRIGger:NEDGe:SOURce?

Among them, <source>, discrete type, {CH1|CH2|CH3|CH4}.

#### **3.2.9.11.2 :TRIGger:NEDGe:SLOPe**

Function: Set the edge type of Nth edge trigger.

Format: :TRIGger:NEDGe:SLOPe <slope>

:TRIGger:NEDGe:SLOPe?

Among them, <slope>, discrete type, {RISE|FALL}.

#### **3.2.9.11.3 :TRIGger:NEDGe:IDLE**

Function: Set the idle time before starting edge counting in the Nth edge trigger.

Format: :TRIGger:NEDGe:IDLE <time>

:TRIGger:NEDGe:IDLE?

Where <time> is a real type ranging from 8ns to 10s.

#### **3.2.9.11.4 :TRIGger:NEDGe:EDGE**

Function: Set the value of N triggered by the Nth edge.

Format: :TRIGger:NEDGe:EDGE <number>

:TRIGger:NEDGe:EDGE?

Among them, <number>, real type, 1 to 65535.

#### **3.2.9.11.5 :TRIGger:NEDGe:LEVel**

Function: Set the trigger level when triggering on the Nth edge

Format: :TRIGger:NEDGe:LEVel <level>

:TRIGger:NEDGe:LEVel?

Among them, <level> is real type.

#### **3.2.9.12 :TRIGger:VIDeo**

##### **3.2.9.12.1 :TRIGger:VIDeo:SOURce**

Function: Set the trigger source of video trigger.

Format: :TRIGger:VIDeo:SOURce <source>

:TRIGger:VIDeo:SOURce?

Among them, <source>, discrete type, {CH1|CH2|CH3|CH4}.

##### **3.2.9.12.2 :TRIGger:VIDeo:POLarity**

Function: Set the polarity of video triggering

Format: :TRIGger:VIDeo:POLarity <polarity>

:TRIGger:VIDeo:POLarity?

Among them, <polarity>, discrete type, { POSItive|NEGAtive }.

### **3.2.9.12.3 :TRIGger:VIDeo:STANdard**

Function: Select the video standard when video triggering.

Format: :TRIGger:VIDeo:STANdard <standard>

:TRIGger:VIDeo:STANdard?

Among them, <standard>, discrete type, {PAL|SECAm|NTSC|720P|1080I|1080P}.

### **3.2.9.12.4:TRIGger:VIDeo:MODE**

Function: Select the synchronization type of video trigger when the trigger standard is PAL, SECAm , NTSC, or 1080I.

Format: :TRIGger:VIDeo:MODE <mode>

:TRIGger:VIDeo:MODE?

Among them, <mode>, discrete type, { ODDField|EVENfield|A LL  
Field|ALLLINE|LINE }.

( The executable parameter marked yellow is EVENfied|ALLField|ALLLine )

### **3.2.9.12.5 :TRIGger:VIDEO:FREQuence**

Function: Select the signal frequency of the video trigger when the trigger standard is 1080P.

Format: :TRIGger:VIDeo:FREQuence <frequency>

:TRIGger:VIDeo:FR EQuence?

Among them, < frequency > is discrete, {60Hz|50Hz|30Hz|25Hz|24Hz}.

### **3.2.9.12.6 :TRIGger:VIDeo:LINE**

Function: Select the specified trigger row for triggering.

Format: :TRIGger:VIDeo:LINE <line>

:TRIGger:VIDeo:LINE?

Among them, <line>, real type, 1~n, depending on the video type, the maximum value of n is different.

### **3.2.9.13 bus trigger command**

Need to pay attention before executing this command

- A、 Open the channel of S1 or S2
- B、 Make channel settings for S1 or S2
- C、 Trigger type is S1 or S2
- D、 And the following commands should be compared with the set S1 or S2 type ;

For example:

Open S1 and S2 ,

Set S1 to CAN and S2 to LIN

Set the trigger type to S1

Then the following trigger should be set to S1 CAN type

### **3.2.9. 13.1 :TRIGger:UART**

#### **3.2.9.13. 1.1 :TRIGger:UART:TYPE**

Function: Set the trigger conditions for UART trigger.

Format: :TRIGger:UART:TYPE < s >, <type>

:TRIGger:UART:TYPE? < s >

Where, <s> is a discrete type, { S1|S2 }; <type> is a discrete type, {START|STOP|DATA|0:DATA|1:DATA|X:DATA| PARItY}.

When the bus word length is set to 9 bits in the bus setting, the trigger type DATA cannot be set;

When the bus word length is set to 5 bit, 6 bit, 7 bit, or 8 bit in the bus setting, 0:DATA, 1:DATA, or X:DATA in the trigger type cannot be set.

**Return format:** The query returns "START", "STOP", "DATA", "0:DATA", "1:DATA", "X:DATA", " PARItY }".

#### **Example:**

The following command sets the START trigger of the S1 channel UART .

:TRIGger:UART:TYPE S1, START

The query below returns "START " .

:TRIGger:UART:TYPE? S1

### **3.2.9.13.2 :TRIGger:UART:RELAtion**

Function: When the UART bus trigger condition is selected as DATA, 0:DATA, 1:DATA, X:DATA, set the UART bus trigger relationship.

Format: :TRIGger:UART:RELAtion < s >, <RELATION>

:TRIGger:UART:RELAtion? < s >

Among them, <s> is discrete type, { S1|S2 }; <RELATION> is discrete type, { GREAT|LESS|EQUAL| UNEQual }.

GREAT: The oscilloscope input data is greater than the specified trigger data ;

LESS: The oscilloscope input data is less than the specified trigger data;

EQUAL: The oscilloscope input data is equal to the specified trigger data;

UNEQual: The oscilloscope input data is not equal to the specified trigger data;

**Return format:** The query returns " GREAT ", "LESS", " EQUAL ", and " UNEQual ".

#### **Example:**

The following command sets the trigger relationship of S1 channel UART to GREAT.

:TRIGger:UART:RELAtion S1, GREAt

The query below returns "GREAt ".

:TRIGger:UART:RELAtion? S1

### **3.2.9.13.3 :TRIGger:UART:DATA**

Function: When the UART bus trigger condition is selected as DATA, 0:DATA, 1:DATA, X:DATA, set the UART bus trigger data.

Format: :TRIGger:UART:DATA < s >, <data>

:TRIGger:UART:DATA?

Where, <s> is a discrete type, { S1|S2 }; <data> is an integer type, hexadecimal, 0 to FF.

**Return format:** The query returns hexadecimal values from 0 to FF.

#### **Example:**

The following command sets the DATA value of S1 channel UART to AA.

:TRIGger:UART:DATA S1, AA

The query below returns " AA ".

:TRIGger:UART:DATA? S1

### **3.2.9.13.4 :TRIGger:LIN**

#### **3.2.9.13.4.1 :TRIGger:LIN:TYPE**

Function: Set the trigger condition for LIN trigger.

Format: :TRIGger:LIN:TYPE < s >, <type>

:TRIGger:LIN:TYPE?

Among them, <s>, discrete type, {S1|S2}; <type>, discrete type, { SRISe|FID| IDATa }.

SRISe, synchronous rising edge; FID, frame ID; IDATa , frame ID and data.

**Return format:** The query returns "SRISe" "FID" " IDATa ".

**Example:**

The following command sets the S1 channel .

:TRIGger:LIN:TYPE S1, SRISe

The following query returns "SRISe".

:TRIGger:LIN:TYPE? S1

**3.2.9.13.4.2 :TRIGger:LIN:ID**

Function: When the LIN bus trigger condition is FID or IDATa, set the trigger ID value of the LIN trigger.

Format: :TRIGger:LIN:ID <s> , <data>

:TRIGger:LIN:ID?

Among them, <s>, discrete type, {S1|S2}; <data>, integer type, hexadecimal, 0 to 3F.

**Return format:** The query returns hexadecimal, values from 0 to 3F.

**Example:**

The following command sets the DATA value of S1 channel LIN to 0A .

:TRIGger:LIN:ID S1, 0A

The following query returns "0A".

:TRIGger:LIN:ID? S1

**3.2.9.13.4.3 :TRIGger:LIN:DATA**

Function: When the LIN bus trigger condition is IDATa , set the trigger data of LIN trigger.

Format: :TRIGger:LIN:DATA < s >, <data>

:TRIGger:LIN:DATA?

Among them, <s> is discrete type, {S1|S2}; <data> is integer type, hexadecimal, 0 to FFFF,FFFF ,FFFF,FFFF.

**Return format:** The query returns a hexadecimal value from 0 to FFFF,FFFF,FFFF,FFFF.

**Example:**

The following command sets the DATA value of S1 channel LIN to 0A .

```
:TRIGger:LIN:DATA S1, 0A
```

The following query returns "0A".

```
:TRIGger:LIN:DATA? S1
```

### 3.2.9.13.5 :TRIGGER:CAN

#### 3.2.9.13.5.1 :TRIGger:CAN:TYPE

Function: Set the trigger condition of CAN trigger

Format: :TRIGger:CAN:TYPE < s >, <type>

```
:TRIGger:CAN:TYPE?
```

Where, <s> is a discrete type, {S1|S2}; <type> is a discrete type, { FSTArt|RFID|DFID|RDID|IDATa|WRFR|AERRor|ACKError| OVERload }.

FSTArt , frame start; RFID, remote frame ID; DFID data frame ID; RDID, remote frame/data frame ID; IDATa , data frame ID and data; WRFR, error frame; AERRor , all errors; ACKError , acknowledgement error; OVERload , overload frame.

**3.2.9.13.5.2 :TRIGger:CAN:ID**

Function: When the trigger condition of CAN trigger is RFID, DFID, IDATa or RDID, set the trigger ID value of CAN trigger.

Format: :TRIGger:CAN:ID < s >, <data>

:TRIGger:CAN:ID?

Among them, <s>, discrete type, {S1|S2}; <data>, integer type, hexadecimal, 0 to 7 FFF,FFFF .

**3.2.9.13.5.3 :TRIGger:CAN:DLC**

Function: When the trigger condition of CAN trigger is IDATa , set the DLC value of CAN trigger.

Format: :TRIGger:CAN:DLC <s> , <data>

:TRIGger:CAN:DLC?

Among them, <s>, discrete type, {S1|S2}; <data>, integer type, 0 to 8 , 1 2 , 1 6 , 2 0 , 2 6 , 32 , 4 8 , 6 4 .

**3.2.9.13.5.4 :TRIGger:CAN:DATA**

Function: When the trigger condition of CAN trigger is IDATa , set the trigger data value of CAN trigger.

Format: :TRIGger:CAN:DATA <s> , <data>

:TRIGger:CAN:DATA?

Among them, <s>, discrete type, {S1|S2}; <data>, integer type, hexadecimal, the number of data digits is determined by DLC.

### **3.2.9.13.6 :TRIGger:SPI**

#### **3.2.9.13.6.1 :TRIGger:SPI:DATA**

Function: Set the data value under SPI trigger.

Format: :TRIGger:SPI:DATA <s>,<data>

:TRIGger:SPI:DATA?

Among them, <s> is discrete type, {S1|S2} ; <data> is integer type, binary.

#### **3.2.9.13.6.2 :TRIGger:SPI:TYPE**

Function: Set the data value under SPI trigger.

Format: :TRIGger:SPI:TYPE <s>,< dtype >

:TRIGger:SPI:TYPE?

Among them, <s> , discrete type, {S1|S2} ; <type> discrete type ,  
{ CS|DATA|X:DATA }

### **3.2.9.13.7 :TRIGger:IIC**

#### **3.2.9.13.7.1 :TRIGger:IIC:TYPE**

Function: Set the trigger type of IIC trigger.

Format: :TRIGger:IIC:TYPE <s> , <type>

:TRIGger:IIC:TYPE?

Among them, <s>, discrete type, {S1|S2}; <type>, discrete type, {START|STOP|ACKLost|NACKaddress|REStart|RDATa|FRAM1|FRAM2 |WRITe10 }.

START, start condition; STOP, stop condition; ACKLost , confirmation loss; NACKaddress , no confirmation in address field; REStart , restart; RDATa , EEPROM data reading; FRAM1, frame type 1; FRAM2, frame type 2; WRITe10 , 10 -bit write frame.

### **3.2.9.13.7.2 :TRIGger:IIC:ADDRess**

Function: When the IIC trigger condition is NACKaddress , FRAM1, FRAM2 or WRITe10 , set the trigger address of the IIC bus trigger.

Format: :TRIGger:IIC:ADDRess < s >, <data>

:TRIGger:IIC:ADDRess?

Where, <s> is a discrete type, {S1|S2}; <data> is an integer type, hexadecimal, 0 to 7F (7 bits) or 0 to 3FF ( 10 bits ).

### **3.2.9.13.7.3 :TRIGger:IIC:RELation**

Function: When the IIC trigger condition is RDATa , set the trigger relationship of the IIC bus trigger.

Format: :TRIGger:IIC:RELation <s> , <relation>

:TRIGger:IIC:RELation

Among them, <s>, discrete type, {S1|S2}; <RELATION>, discrete type, { GREAT|LESS|EQUAL| UNEQual }.

GREAT: The oscilloscope input data is greater than the specified trigger data ;

LESS: The oscilloscope input data is less than the specified trigger data;

EQUAL: The oscilloscope input data is equal to the specified trigger data;

UNEQual: The oscilloscope input data is not equal to the specified trigger data;

#### **3.2.9.13.7.4 :TRIGger:IIC:DATA**

Function: when the IIC trigger condition is RDATA , FRAM1, FRAM2 or WRITe 10 , set the trigger data for the IIC bus trigger.

Format: :TRIGger:IIC:DATA <s> , <data>

:TRIGger:IIC:DATA?

Among them, <s> is discrete type, {S1|S2}; <data> is integer type, hexadecimal, 0 -FF .

#### **3.2.9.13.7.5 :TRIGger:IIC:DATA 2**

Function: When the IIC trigger condition is FRAM2, set the trigger data triggered by the IIC bus.

Format: :TRIGger:IIC:DATA2 < s >, <data>

:TRIGger:IIC:DATA2?

Among them, <s> is discrete type, {S1|S2}; <data> is integer type, hexadecimal, 0 -FF .

### **3.2.9.13.8 :TRIGger:1553B**

#### **3.2.9.13.8.1 :TRIGger:1553B:TYPE**

Function: Set the trigger condition for 1553B bus trigger.

Format: :TRIGger:1553B:TYPE < s >, <type>

:TRIGger:1553B:TYPE?

Among them, <s> is a discrete type, {S1|S2}; <type> is a discrete type, { CSSYnc|DWSYnc|CSWOrd|DWORD|RTADdress|OPERror|MERRor| AERRor }.

CSSYnc , command/status word synchronization header; DWSYnc , data word synchronization header; CSWOrd , command/status word; DWORDd , data word; RTADdress , remote terminal address; OPERror , odd parity error; MERRor , Manchester code error; AERRor , all errors.

#### **3.2.9.13.8.2 :TRIGger:1553B:CSWOrd**

Function: When the 1553B trigger condition is CSWOrd , set the instruction/status word value triggered by the 1553B bus.

Format: :TRIGger:1553B:CSWOrd < s >, <data>

:TRIGger:1553B:CSWOrd?

Among them, <s>, discrete type, {S1|S2}; <data>, integer type, 0 to FFFF.

### **3.2.9.13.8.3 :TRIGger:1553B:DWORd**

Function: When the 1553B trigger condition is DWORd , set the trigger data value of the 1553B bus trigger.

Format: :TRIGger:1553B:DWORd < s >, <data>

:TRIGger:1553B:DWORd?

Among them, <s>, discrete type, {S1|S2}; <data>, integer type, 0 to FFFF.

### **3.2.9.13.8.4 :TRIGger:1553B:RTAdress**

Function: When the 1553B bus trigger condition is RTAdress , set the remote terminal address triggered by the 1553B bus.

Format: :TRIGger:1553B:RTAdress < s >, <address>

:TRIGger:1553B:RTAdress?

Among them, <s>, discrete type, {S1|S2}; <address>, integer type, 0 to FF.

### **3.2.9.13.9 :TRIGger:429**

#### **3.2.9.13.9.1 :TRIGger:429:TYPE**

Function: Set the trigger condition for 429 bus trigger.

Format: :TRIGger:429:TYPE < s >, <type>

:TRIGger:429:TYPE?

Among them, <s> is discrete type, {S1|S2}; <type> is discrete type ,  
 { WBEGin|WEND |  
 LABEL|SDI|DATA|SSM|LSDI|LDATa|LSSM|WERROr|WINTerval|VERRor|AERRor|ALL0|AL  
 L1}.

WBEGin, word start; WEND, word end; LSDI, LABEL+SDI; LDATa ,  
 LABEL+DATA; LSSM, LABEL+SSM; WERROr , word error; WINTerval , word gap  
 error; VERRor , checksum error; AERRor , all errors ; ALL0, all 0 bits; ALL1, all 1 bits.

Among them, after selecting LSDI , LDATa , and LSSM , you need to set the  
 accompanying parameters, so use LABEL , SDI , DATA , and SSM to set the  
 parameters;

For example:

:TRIGger:429:TYPE S1, LSDI

:TRIGger:429:LABEL S1,377

:TRIGger:429:SDI S1,11

### **3.2.9.13.9.2 :TRIGger:429:WBEG in**

Function: When the 429 bus trigger condition is WORD start, set the trigger  
 word value of the 429 bus trigger.

Format: :TRIGger:429:WBEG in < s > >

:TRIGger:429:WBEG in?

Among them, <s> is discrete, {S1|S2}.

### **3.2.9.13.9.3 :TRIGger:429:WEND**

Function: When the 429 bus trigger condition is WORD ended, set the trigger word value of the 429 bus trigger.

Format: :TRIGger:429:WEND < s >

:TRIGger:429:WEND?

Among them, <s> is discrete, {S1|S2}.

### **3.2.9.13.9.4 :TRIGger:429:LABEL**

Function: When the 429 bus trigger condition is LABEL , LSDI, LDATa or LSSM, set the trigger LABEL value of the 429 bus trigger.

Format: :TRIGger:429:LABEL <s> , <data>

:TRIGger:429:LABEL?

Where, <s> is a discrete type, {S1|S2}; <data> is an integer type, octal, 0 to 3 77 .

### **3.2.9.13.9.5 :TRIGger:429:SDI**

Function: When the 429 bus trigger condition is SDI or LSDI, set the trigger SDI value of the 429 bus trigger.

Format: :TRIGger:429:SDI < s >, <data>

:TRIGger:429:SDI?

Among them, <s> is discrete type, {S1|S2}; <data> is integer type, binary, 0 0 to 11 .

### **3.2.9.13.9.6 :TRIGger:429:DATA**

Function: When the 429 bus trigger condition is DATA or LDATa , set the trigger data value of the 429 bus trigger.

Format: :TRIGger:429:DATA <s> , <data>

:TRIGger:429:DATA?

Where, <s> is a discrete type, {S1|S2}; <data> is an integer type, hexadecimal, ranging from 0 to FFFFFFFF.

### **3.2.9.13.9.7 :TRIGger:429:SSM**

Function: When the 429 bus trigger condition is SSM or LSSM, set the trigger data value of the 429 bus trigger.

Format: :TRIGger:429:SSM < s > , <data>

:TRIGger:429:SSM?

Among them, <s>, discrete type, {S1|S2}; <data>, integer type, binary, 0 to 11 .

## **3.2.10 Time base command subsystem**

### **3.2.10.1 :TIMebase:EXTent**

Function: Set the horizontal time base gear.

Format: :TIMebase:EXTent <extent>

:TIMebase:EXTent?

Among them, <extent>, real type. Unit:S

**Return format:** The query returns the offset value in scientific notation.

#### **Example:**

The following command sets the horizontal time base to 2us.

Format: :TIMebase:EXTent 2.000000e-6

The following query returns "2.000000e-06"

:TIMebase:EXTent?

### **3.2.10.2 :TIMebase:MODE**

Function: Set the screen time base display mode. "YT" or "XY".

Format: :TIMebase:MODE <mode>

:TIMebase:MODE?

Where <mode> is discrete, "YT" or "XY".

### **3.2.10.3 :TIMebase:ROLL:DISPlay**

Function: Turn on or off ROLL mode ( time base above 100 ms).

Format: :TIMebase:ROLL:DISPlay <bool>

:TIMebase:ROLL:DISPlay?

Among them, <bool> , Boolean type, {{0|OFF}}{1|ON}}.

Note:If you need to work in roll mode, after turning on this switch, you also need to set the time base above 100ms / div.

### **3.2.10.4 :TIMebase:POSition**

**Function:** Set the horizontal offset of the waveform display.

**Format:** :TIMebase:POSition <position>

:TIMebase:POSition?

Among them, < POSition > is real type.

**Return Format:** The query returns the offset value in scientific notation.

#### **Example:**

The following command sets the horizontal offset to 2us.

```
:TIMebase:POSition 0.000002
```

The following query returns "2.000000e-06"

```
:TIMebase:POSition?
```

### **3.2.10.5 :TIMebase:ZOOM:SCAle**

**Function:** Set and query the time base of the large window after zoom is opened.

:TIMebase:ZOOM:SCAle < value >

:TIMebase:ZOOM:SCAle?

Among them, <value> is a real type, {1e-9~1e3}

### **3.2.11 Storage command subsystem**

#### **3.2.11.1 :STORAge:SAVE**

Function: Store the waveform of the specified channel to the specified location.

Format: :STORAge:SAVE <channel>,<save>

:STORAge:SAVE <channel>

Among them, <channel>, discrete type, {CH1|CH2|CH3|CH4|MATH}; <save> , discrete type , { LOCAL|UDISK }, default LOCAL

Note:In segmented storage, the current frame is stored.

#### **3.2.11.1.1 :STORAge:SAVE:SOURce**

Format: :STORAge:SAVE:SOURce <channel>

:STORAge:SAVE:SOURce?

Among them, <channel>, discrete type, {CH1|CH2|CH3|CH4|MATH};

### 3.2.11.1.2 :STORage:SAVE:LOCAtion

Format: :STORage:SAVE:LOCAtion <location>

:STORage:SAVE:LOCAtion?

Among them, < LOCAtion >, discrete type, { LOCa||UDISk };

### 3.2.11.1.3 :STORage:SAVE:TYPE

Format: :STORage:SAVE:TYPE <type>

:STORage:SAVE:TYPE?

Among them, <TYPE>, discrete type, {WAV|BIN|CSV};

### 3.2.11.1.4 :STORage:SAVE:FILEname

Format: :STORage:SAVE:FILEname <filename>

:STORage:SAVE:FILEname?

Where, < file name>:= quoted ASCII string

### 3.2.11.1.5 :STORage:SAVE:ALLSegments <bool>

Function: In the case of segmented storage, set to save all segments

Format: :STORage:SAVE:ALLSegments < bool >

:STORage:SAVE:ALLSegments?

Among them, <bool>, Boolean type, {{0|OFF}}{1|ON}}.

**Return Format:** The query returns "0" or "1".

**Example:**

The following command enables storage of all segments in the case of segmented storage.

```
:STORage:SAVE:ALLSegments ON or :STORage:SAVE:ALLSegments 1
```

The query below returns "1".

```
:STORage:SAVE:ALLS elements?
```

Note: This command is only valid when segmented storage is turned on. After this command is turned on, the storage type: `:STORage:SAVE:TYPE <type>` can only select BIN .

### **3.2.11.1.6 :STORage:SAVE:START**

Format: `:STORage:SAVE:START`

Start storing.

### **3.2.11.2 :STORage:LOAD**

Function: Load ref.

Format: `:STORage:LOAD <ref><bool>, <filename>`

Among them, `<source>`, discrete type, {R1| R2| R3| R4 }; `< filename>`, discrete type, the loaded name, `<bool>`, Boolean type, {{0|OFF}}{1|ON}}.

### 3.2.11.3 :STORage:CAPTure

Function: Screenshot related settings.

#### 3.2.11.3.1 :STORage:CAPTure:TIME <bool>

Functions: Settings and queries, timestamp of screenshots

Format: :STORage:CAPTure:TIMEstamp <bool>

:STORage:CAPTure:TIMEstamp?

Among them, <bool>, Boolean type, {{0|OFF}}{1|ON}}.

**Return format:** The query returns "0" or "1".

#### **Example:**

The following command turns on the timestamp of the screenshot.

:STORage:CAPTure:TIMEstamp ON or :STORage:CAPTure:TIMEstamp 1

The following query returns "1".

:STORage:CAPTure:TIMEstamp?

#### 3.2.11.3.2 :STORage:CAPTure:INCOlor <bool>

Function: Set and query whether the screenshot is inverted

Format: :STORage:CAPTure:INCOlor < bool >

:STORage:CAPTure:INCOlor?

Among them, <bool>, Boolean type, {{0|OFF}}{1|ON}}.

**Return Format:** The query returns "0" or "1".

**Example:**

The following command turns on timestamps for screenshots.

:STORage:CAPTure:INCOlor ON or :STORage:CAPTure:INCOlor 1

The following query returns "1".

:STORage:CAPTure:INCOlor?

### **3.2.11.3.3 :STORage:CAPTure:STARt**

Function: Start screenshot

Format: :STORage:CAPTure:STARt

**Example:**

The following command starts taking screenshots.

:STORage:CAPTure:STARt

### **3.2.11.4 :STORage:CONSave**

Function: Store oscilloscope settings.

Defines the name of the oscilloscope setting

Format: :STORage:CONSave:FILEname <filename>

where, < file name>:= quoted ASCII string

Format: :STORage:CONSave:START

Start storing.

### **3.2.11.5 :STORage:CONLoad:FILEname<filename>**

Function:Call the oscilloscope settings with the corresponding name.

## **3.2.12 Bus configuration command subsystem**

### **3.2.12.1 :BUS<S>**

#### **3.2.12.1.1 :BUS <S>:DISPlay**

Function: switch decoding channel.

Format: :BUS <S>:DISPlay <bool>

:BUS <S>:DISPlay?

Among them, <n>, discrete type, { 1|2|3|4} ; <bool>, Boolean type, {{0| OFF} {1|ON}}.

**Return format:**The query returns "0" or "1".

#### **Example:**

The following command turns on the display of decoding channel 1.

:BUS1:DIAPlay ON or :BUS1:DIAPlay 1

The query below returns "1".

:BUS1:display?

### 3.2. 12 .1.2 :BUS <S>:TYPE

Function: Set the bus type of bus S1 or S2 .

Format: :BUS <S>:TYPE <type>

:BUS <S>:TYPE?

Among them, <s> is discrete, {S1|S2} ;

<type> , discrete type, {UART|LIN|SPI|CAN|IIC|1553B| 429} .

### 3.2.12.1.3 :BUS <S>:MODE <mode>

Function:Set the display mode of the bus, including graphic and text modes.

Format: :BUS <S>:MODE < mode >

:BUS <S>:MODE?

Among them, < mode > is discrete type, {GRAP|TXT} .

### 3.2. 12 .1.4 :BUS <S>:LEVEl < channel >,<level>

Function: Set the threshold level of the bus.

Format: :BUS <S>:LEVEl < channel >,<level>

:BUS <S>:LEVEl? <channel>

Among them, <channel>, discrete type, {CH1|CH2|CH3|CH4}; <level> , real type.

Note: This command needs to be set in graphics mode (:BUS<S>:MODE GRAP ) after completing other bus configurations.

#### **3.2.12.1.5 :BUS <S>:HLEVel < channel >,<level>**

Function: When the bus has 2 threshold levels, set the high threshold level of the bus.

Format: :BUS <S>:HLEVel < channel >,<level>

:BUS <S>:HLEVel? <channel>

Among them, <channel> is discrete type, {CH1|CH2|CH3|CH4}; <level> is real type.

Note: This command needs to be set in graphics mode (:BUS<S>:MODE GRAP ) after completing other bus configurations.

#### **3.2.12.1.6 :BUS <S>:LLEVel < channel > , <level>**

Function: When the bus has 2 threshold levels, set the lower threshold level of the bus.

Format: :BUS <S>:LLEVel < channel >,<level>

:BUS <S>:LLEVel? <channel>

Among them, <channel>, discrete type, {CH1|CH2|CH3|CH4}; <level> , real type.

---

Note: This command needs to be set in graphics mode (:BUS<S>:MODE GRAP )  
after completing other bus configurations.

### **3.2. 12.2 :BUS <S>:UART**

#### **3.2. 12 .2.1 :BUS <s>:UART:RX**

Function: Set the RX channel source of UART bus configuration .

Format: :BUS <s>:UART:RX <channel>

:BUS <s>:UART:RX?

Among them, <s> is discrete, {S1|S2}, and <channel> is discrete,  
{CH1|CH2|CH3|CH4} .

#### **3.2.12.2.2 :BUS <s>:UART:IDLElvl**

Function:Set the idle level state of the UART bus configuration.

Format: :BUS <s>:UART:IDLElvl <state>

:BUS <s>:UART:IDLElvl?

Among them, <state> , discrete type, { high|low } .

#### **3.2.12.2.3 :BUS <s>:UART:BAUDrate**

Function: Select the baud rate of UART bus configuration. Unit:b/s

Format: :BUS <s>:UART:BAUDrate < baudrate >

:BUS <s>:UART:BAUDrate?

Among them, < baudrate > , discrete type,  
{1200|2400|4800|9600|19200|38400|43000|56000|57600|115200} .

#### **3.2.12.2.4 :BUS <s>:UART:CHECK**

Function: Select the verification method of UART bus configuration.

Format: :BUS <s>:UART:CHECK <check>

:BUS <s>:UART:CHECK?

Among them, <check> , discrete type, {NONE|ODD|EVEN} .

#### **3.2.12.2.5 :BUS <s>:UART:USERbaud**

Function: Select the user-defined baud rate when configuring the UART bus.

Unit:b/s

Format: :BUS <s>:UART:USERbaud < baudrate >

:BUS <s>:UART:USERbaud?

Where, <baudrate> is an integer ranging from 1200 to 8000000 .

#### **3.2.12.2.6 :BUS <s>:UART:WIDTH**

Function: Select the data width when configuring the UART bus.

Format: :BUS <s>:UART:WIDTH <width>

:BUS <s>:UART:WIDTH?

Where <width> is discrete, {5|6|7|8|9} .

### **3.2.12.2.7 :BUS <s>:UART:DISPlay**

Function: Select the data display mode when UART bus is configured.

Format: :BUS <s>:UART:DISPlay < display >

:BUS <s>:UART:DISPlay?

Among them, < display > is discrete, {HEX|BIN|ASCII } .

### **3.2.12.3 :BUS <s>:LIN**

#### **3.2.12.3.1 :BUS <S>:LIN:CHANnel**

Function: Select the channel source for LIN bus configuration.

Format: :BUS <S>:LIN:CHANnel <channel>

:BUS <S>:LIN:CHANnel?

Where <channel> is a discrete type, {CH1|CH2|CH3|CH4} .

#### **3.2.12.3.2 :BUS <S>:LIN:IDLElvl**

Function: Set the idle level state of the LIN bus configuration.

Format: :BUS <S>:LIN:IDLElvl <state>

:BUS <S>:LIN:IDLElvl?

Among them, <state> , discrete type, { high|low } .

### **3.2.12.3.3 :BUS <S>:LIN:BAUDrate**

Function: Select the baud rate of LIN bus configuration. Unit, b/s .

Format: :BUS <S>:LIN:BAUDrate < baudrate >

:BUS <S>:LIN:BAUDrate?

Where < baudrate > is discrete, {2400|9600|19200} .

### **3.2.12.3.4 :BUS <S>:LIN:USERbaud**

Function: Select the user-defined baud rate when configuring the LIN bus.

Unit:b/s

Format: :BUS <S>:LIN:USERbaud <baudrate>

:BUS <S>:LIN:USERbaud?

Where, <baudrate> is an integer ranging from 2400 to 625000 .

### **3.2.12.4 :BUS <S>:SPI**

#### **3.2.12.4.1 :BUS <S>:SPI:CLK**

Function: Select the clock source for SPI bus configuration.

Format: :BUS <S>:SPI:CLK <channel>

:BUS <S>:SPI:CLK?

Where <channel> is a discrete type, {CH1|CH2|CH3|CH4} .

#### **3.2.12.4.2 :BUS <S>:SPI:DATA**

Function: Select the data source for SPI bus configuration.

Format: :BUS <S>:SPI:DATA <channel>

:BUS <S>:SPI:DATA?

Where <channel> is a discrete type, {CH1|CH2|CH3|CH4} .

#### **3.2.12.4.3 :BUS <S>:SPI:WIDTh**

Function: Select the data width when configuring the SPI bus.

Format: :BUS <S>:SPI:WIDTh <width>

:BUS <S>:SPI:WIDTh?

Where <width> is discrete and can be {4|8|16|24|32} .

#### **3.2.12.4.4 :BUS <S>:SPI:IDLElvI**

Function: Select the idle level state of the SPI bus configuration.

Format: :BUS <S>:SPI:IDLElvI <state>

:BUS <S>:SPI:IDLElvI?

Where <state> is discrete, { high|low } .

#### **3.2.12.4.5 :BUS <S>:SPI:SLOPe**

Function: Select the clock edge type for SPI bus configuration.

Format: :BUS <S>:SPI:SLOPe <slope>

:BUS <S>:SPI:SLOPe?

Where <slope> is discrete, {RISE|FALL}.

#### 3.2.12.4.6 :BUS <S>:SPI:CS

Function: Set and query the CS enable in SPI

Format: :BUS <S>:SPI:CS <bool>

:BUS <S>:SPI:CS?

Among them, <bool>, Boolean type, {{0|OFF}}{1|ON}}.

**Return Format:** The query returns "0" or "1".

#### Example:

The following command turns on CS enable .

:BUS <S>:SPI:CS <bool> ON or :BUS<S>:SPI:CS <bool> 1

The query below returns "1".

:BUS <S>:SPI:CS?

#### 3.2.12.4.7 :BUS <S>:SPI:CS:SOURce

Function: Set and query the source of CS in SPI

Format: :BUS <S>:SPI:CS:SOURce <channel>

:BUS <S>:SPI:CS:SOURce?

Among them, <channel> , discrete type, {CH1|CH2|CH3|CH4} .

**Return format:** The query returns " CH1 ", " CH2 ", " CH3 ", and " CH4 " .

**Example:**

The following command sets the source of CS .

```
:BUS <S>:SPI:CS:SOURce CH1
```

The query below returns " CH1 " .

```
:BUS <S>:SPI:CS:SOURce?
```

### 3.2.12.4.8 :BUS <S>:SPI:CS:IDLElvl

Function: Select the idle level state of CS in SPI bus configuration .

Format: :BUS <S>:SPI:CS:IDLElvl <state>

```
:BUS <S>:SPI:CS:IDLElvl?
```

Where <state> is discrete, {HIGH|LOW} .

**Return format:** The query returns "HIGH " and " LOW " .

**Example:**

The following command sets the source of CS .

```
:BUS <S>:SPI:CS:IDLElvl HIGH
```

The query below returns "HIGH " .

:BUS <S>:SPI:CS:IDLElvl?

### **3.2.12.5 :BUS <S>:CAN ( FD )**

#### **3.2.12.5.1 :BUS <S>:CAN:CHANnel**

Function: Select the channel source for CAN bus configuration.

Format: :BUS <S>:CAN:CHANnel <channel>

:BUS <S>:CAN:CHANnel?

Where <channel> is a discrete type, {CH1|CH2|CH3|CH4} .

#### **3.2.12.5.2 :BUS <S>:CAN:SIGNAL**

Function: Set the idle level state of the CAN bus configuration.

Format: :BUS <S>:CAN:SIGNAL < signal >

:BUS <S>:CAN:SIGNAL?

Among them, < signal > is discrete type, {CAN\_H|CAN\_L|H\_L|L\_H|RX|TX} .

#### **3.2.12.5.3 :BUS <S>:CAN:BAUDrate**

Function: Select the baud rate of the CAN bus configuration. Unit, b/s .

Format: :BUS <S>:CAN:BAUDrate < baudrate >

:BUS <S>:CAN:BAUDrate?

Where < baudrate > is discrete,  
{[50000|100000|250000|500000|800000|1000000]} .

#### **3.2.12.5.4 :BUS <S>:CAN:USERbaud**

Function:Select user-defined baud rate during CAN bus configuration. Unit:b/s

Format: :BUS <S>:CAN:USERbaud < baudrate >

:BUS <S>:CAN:USERbaud?

Among them, < baudrate > , integer, 10000 to 1000000 .

#### **3.2.12.5.5 :BUS <S>:CAN:SAMPlEpoint**

Function: Select the sampling point during CAN bus configuration and the sampling point of the arbitration field during CAN FD , unit %

Format: :BUS <S>:CAN:SAMPlEpoint < percent >

:BUS <S>:CAN:SAMPlEpoint?

Among them, <percent> , integer , 1 to 99 .

#### **3.2.12.5.6 :BUS <S>:CAN:FDBAud rate**

Function: Select the baud rate of the CAN FD bus configuration data bit.

Unit:b/s .

Format: :BUS <S>:CAN:FDBAud rate < baudrate >

:BUS <S>:CAN:FDBAud rate?

Where, < baudrate > , discrete, {NONE|2M|5M}

### **3.2.12.5.7 :BUS <S>:CAN:FDUSERbaud**

Function: Select the user-defined baud rate of the data bit when configuring the CANFD bus. Unit:b/s

Format: :BUS <S>:CAN:FDUSERbaud < baudrate >

:BUS <S>:CAN:FDUSERbaud?

Where, <baudrate> is an integer ranging from 1,000,000 to 12,000,000 .

### **3.2.12.5.8 :BUS <S>:CAN:FDSAmappoint**

Function: Select the sampling point of the CANF data field , unit:%

Format: :BUS <S>:CAN:FDSA mplepoint < percent >

:BUS <S>:CAN:FDSAmplepoint?

Where < percent > is an integer ranging from 1 to 99 .

### **3.2.12.5.9 :BUS <S>:CAN:ISO**

Function: Set the standard of CAN bus configuration, ISO or non- ISO .

Format: :BUS <S>:CAN:ISO < iso >

:BUS <S>:CAN:ISO?

Among them, <iso> , discrete type, {ISO|NON } .

### **3.2.12.6 :BUS <S>:IIC**

### **3.2.12.6.1 :BUS <S>:IIC:SDA**

Function: Set the serial data channel source of the IIC bus configuration.

Format: :BUS <S>:IIC:SDA <channel>

:BUS <S>:IIC:SDA?

Among them, <channel> , discrete type, {CH1|CH2|CH3|CH4} .

### **3.2.12.6.2 :BUS <S>:IIC:SCL**

Function: Set the channel source of the serial clock configured by the IIC bus.

Format: :BUS <S>:IIC:SCL <channel>

:BUS <S>:IIC:SCL?

Where <channel> is a discrete type, {CH1|CH2|CH3|CH4} .

### **3.2.12.7 :BUS<S>:1553B**

#### **3.2.12.7.1 :BUS <S>:1553B:SOURce**

Function: Set the channel source of 1553B bus configuration.

Format: :BUS <S>:1553B:SOURce <channel>

:BUS <S>:1553B:SOURce?

Among them, <channel> , discrete type, {CH1|CH2|CH3|CH4} .

#### **3.2.12.7.2 :BUS <S>:1553B:DISPlay**

Function: Set the display mode of 1553B bus configuration.

Format: :BUS <S>1553B:DISPlay <diaplay>

:BUS <S>:1553B:DISPlay?

Among them, <diaplay> , discrete type, { BINArY| HEX } .

### **3.2.12.8 :BUS<S>:429**

#### **3.2.12.8.1 :BUS <S>:429:SOURce**

Function: Set the channel source of 429 bus configuration.

Format: :BUS <S>:429:SOURce <channel>

:BUS <S>:429:SOURce?

Where <channel> is a discrete type, {CH1|CH2|CH3|CH4} .

#### **3.2.12.8.2 :BUS <S>:429:FORMat**

Function: Set the format of 429 bus configuration

Format: :BUS <S>:429:FORMat <format>

:BUS <S>:429:FORMat?

Where <format> is discrete, { LDAT |LDSS| LSDS }.

LSDS , LABEL+SDI+DATA+SSM ; LDSS , LABEL+DATA+SSM ; LDAT ,  
LABEL+DATA .

### 3.2.12.8.3 :BUS <S>:429:DISPlay

Function: Set the display mode of 429 bus configuration.

Format: :BUS <S>:429:DISPlay <diaplay>

:BUS <S>:429:DISPlay?

Among them, <diaplay> , discrete type, { BINARy| HEX } .

### 3.2.12.8.4 :BUS <S>:429:BANDrate

Function: Set the baud rate of 429 bus configuration.

Format: :BUS <S>:429:BANDrate < bandrate >

:BUS <S>:429:BANDrate?

Among them, < bandrate > , discrete type, {12500|100000} .

## 3.2.13 Reference waveform command subsystem

### 3.2.13.1 :REFerence:DISPlay

Function: Turn REF function on or off.

Format: :REFerence:DISPlay <bool>

:REFerence:DISPlay?

Among them, <bool> is a Boolean type, {{0|OFF}}{1|ON}} .

Return format: The query returns " 1 " or " 0 " .

Example:

The following command turns on the REF function.

```
:REFErence:DISPLay ON
```

The following command returns 1.

```
:REFErence:DISPLay?
```

### **3.2.13.2 :REFErence <n>:ENABle <bool>**

Function: Turn on or off the specified reference channel.

Format: :REFErence <n>:ENABle <bool>

```
:REFErence <n>:ENABle?
```

Among them, <n> , discrete type, { 1|2|3|4} ; <bool> , Boolean type, {{0| OFF} | {1|ON}} .

Return format: The query returns " 1 " or " 0 " .

Example:

The following command turns on R1 .

```
:REFErence1:ENABle ON
```

The following command returns 1.

```
:REFErence1:ENABle?
```

### **3.2.13.3 :REFerence <n>:HSCale <scale>**

Function: Set the horizontal scale of the reference channel.

Format: :REFerence <n>:HSCale <scale>

:REFerence <n>:HSCale?

Among them, <n> , discrete type, { 1|2|3|4} ; <scale> , real type, 1ns~1ks or 1mHz~1GHz .

Return format: The query returns the horizontal gear in scientific notation.

### **3.2.13.4 :REFerence <n>:VSCale <scale>**

Function: Set the vertical scale of the reference channel.

Format: :REFerence <n>:VSCale <scale>

:REFerence <n>:VSCale?

Among them, <n> , discrete type, { 1|2|3|4} ; <scale> , real type, 5mV~5GV .

Return format: The query returns the vertical gear in scientific notation. ( When the reference waveform called is fft , the channel horizontal scale will not change )

Example:

The following command sets the vertical scale of reference channel 1 to 2V .

```
:REFerence1:VSCale 2
```

The following command returns 2.000000e+00.

:REference1:VScale?

### **3.2.13.5 :CURRent:REFerence <n>**

Function: Select the current reference channel.

Format: :CURRent:REFerence <n>

Among them, <n> , discrete type, { R1|R2|R3|R4} .

### **3.2.13.6 :REFerence <n>:VPOSition <pos>**

**Function:** Set the vertical offset of the waveform display of the specified reference channel.

**Format:** :REFerence <n>:VPOSition <pos>

:REFerence <n>:VPOSition?

Among them, <n> is discrete type, { 1|2|3|4} ; <pos> is real type.

**Return Format:** The query returns the offset value in scientific notation.

#### **Example:**

The following command sets the vertical offset of channel 1 to 0.01V .

```
:REFerence 1:VPOSition 0.01
```

The following query returns " 1.000000e-02 "

```
:REFerence1:VPOSition?
```

### 3.2.13.7 :REference <n>:HPOSition <pos>

**Function:** Set the horizontal offset of the waveform display.

**Format:** :REference:HPOSition <n>,<pos>

:REference:HPOSition? <n>

Among them, <n> , discrete type, { 1|2|3|4} ; <pos> , real type.

**Return format:** The query returns the offset value in scientific notation.

#### Example:

The following command sets the R1 horizontal offset to 2us .

```
:REference1:HPOSition 0.000002
```

The query below returns " 2.000000e-06 "

```
:REference1:HPOSition?
```

### 3.2.13.8 :REF <n>:SRATe?

Query the sampling rate of the reference waveform

Where n is a real type {{ 1|2|3|4}}

### 3.2.13.9 :REF <n>:MDEPth?

Query the storage depth of the reference waveform

Where n is a real type { 1|2|3|4}

### 3.2.14 AUTO Setting Subsystem

#### 3.2.14.1 :AUTO:SET:CHANnel <bool>

**Function:** Enable the autoset channel to open and close automatically

**Format:** :AUTO:SET:CHANnel <bool>

:AUTO:SET:CHANnel?

Among them, <bool>, Boolean type, {{0|OFF}}{1|ON}}.

**Return Format:** The query returns "0" or "1".

**Example:**

The following command turns on the display of channel 1.

:AUTO:SET:CHANnel ON or :AUTO:SET:CHANnel1

The following query returns "1".

:AUTO:SET:CHANnel?

#### 3.2.14.2 :AUTO:SET:LEVel <level>

**Function:** Effective threshold for automatic opening and closing of channels in auto

**Format:** :AUTO:SET:LEVel <level>

:AUTO:SET:LEVel?

Among them, <level>, real type, 0.001V~99V.

**Return format:** The query returns the valid threshold value in scientific notation.

**Example:**

The following command sets the effective threshold to 150mV.

:AUTO:SET:LEVnel 0.15

The query below returns "1.500000e-01".

:AUTO:SET:LEVel?

**3.2.14.3 :AUTO:SET:SOURce <source>**

**Function:** When autoset is executed, the trigger source rule is selected first, current priority | maximum value priority

**Format:** :AUTO:SET:SOURce <source>

:AUTO:SET:SOURce?

Where <source> is discrete and is { CURrent|MAX }.

**Return Format:** The query returns “ CURrent ” or “MAX” .

**Example:**

When the following command is set to automatic, the trigger source maximum value takes priority

```
:AUTO:SET:SOURce MAX
```

The following query returns "MAX".

```
:AUTO:SET:SOURce?
```

### **3.2.14.4 :AUTO:RANge <bool>**

**Function:** The mode executed by pressing the Auto key, divided into autoset and autorange

**Format:** :AUTO:RANGE <bool>

```
:AUTO:RANge?
```

Where <bool> is a Boolean type, {{0|OFF}}{{1|ON}}, when it is 0, it is in AutoSet mode, and when it is 1, it is in AutoRange mode .

**Return Format:** The query returns "0" or "1".

**Example:**

The following command enables Autorange .

```
:AUTO:RANge ON or :AUTO:RANge 1
```

The following query returns "1".

```
:AUTO:RANge?
```

### 3.2.14.5 :AUTO:RANge:VERTical <bool>

**Function:** Whether the vertical scale factor is automatically adjusted during AutoRange

**Format:** :AUTO:RANge:VERTical <bool>

:AUTO:RANge:VERTical?

Where <bool> is a Boolean type, {{0|OFF}}{{1|ON}}

**Return Format:** The query returns "0" or "1".

**Example:**

The following command turns VERTical on automatically.

:AUTO:RANge:VERTical ON or :AUTO:RANge:VERTical 1

The following query returns "1".

:AUTO:RANge:VERTical?

### 3.2.14.6 :AUTO:RANge:HORizontal <bool>

**Function:** Whether the time base is automatically set during the AutoRange process

**Format:** :AUTO:RANge:HORizontal <bool>

:AUTO:RANge:HORizontal?

Where <bool> is a Boolean type, {{0|OFF}}{{1|ON}}

**Return Format:** The query returns "0" or "1".

**Example:**

The following command turns on HORizontal automatically.

:AUTO:RANge:HORizontal ON or :AUTO:RANge:HORizontal 1

The following query returns "1".

:AUTO:RANge:HORizontal?

**3.2.14.7 :AUTO:RANge:LEVel <bool>**

**Function:** Whether the trigger level is automatic during the AutoRange process

**Format:** :AUTO:RANge:LEVel <bool>

:AUTO:RANge:LEVel?

Among them, <bool>, Boolean type, {{0|OFF}}{1|ON}}

**Return format:**The query returns "0" or "1".

**Example:**

The following command turns on LEVel automatically.

:AUTO:RANge:LEVel ON or:AUTO:RANge:LEVel 1

The query below returns "1".

:AUTO:RANge:LEVel?

### **3. 2. 15 Waveform command subsystem**

:WAVeform:SOURce

:WAVeform:MODE

:WAVeform:FORMat

:WAVeform:DATA?

:WAVeform:STARt

:WAVeform:STOP

:WAVeform:PREamble?

:WAVeform:XINCrement?

:WAVeform:XORigin?

:WAVeform:XREFerence?

:WAVeform:YINCrement?

:WAVeform:YORigin?

:WAVeform:YREFerence?

#### **3.2.15.1 :WAVeform:SOURce**

**Function:** sets the channel source for waveform reading.

**Format:** :WAVeform:SOURce <source>

:WAVeform:SOURce?

Where, < source >, discrete type, {CH1|CH2|CH3|CH4}

### **Return Format**

The query returns "CH1", "CH2", "CH3", or "CH4".

### **Example**

The following command selects channel 2 as the channel source.

:WAVeform:SOURce CH2

The query below returns "CH2".

:WAVeform:SOURce?

### **3.2.15.2 :WAVeform:MODE**

**This function** sets or queries the mode of reading waveform.

**Format** :WAVeform:MODE < mode >

:WAVeform:MODE?

Where, < mode >, discrete, { NORMAl|MAXimum|RAW }

### **illustrate**

NORMAl: Returns the number of waveform points after sampling.

---

MAXimum: Returns the maximum number of valid data points in the current state. In the running state, it returns the number of data points displayed on the screen, and in the stopped state, it returns the number of data points in the memory.

RAW: Returns the number of data points in the current system memory. Only valid in the stopped state.

### **Return Format**

The query returns "NORMal ", "MAXimum ", or "RAW".

### **Example**

The following command selects RAW mode.

```
:WAVeform:MODE RAW
```

The following query returns "RAW".

```
:WAVeform:MODE?
```

### **3.2.15.3 :WAVeform:FORMat**

The format of the data returned when **the function** sets or queries the data.

**Format** :WAVeform:FORMat <format>

:WAVeform:FORMat?

Among them, <format> , discrete type, { WORD| ASCII }

### **illustrate**

WORD: The data of one point occupies 16 bits, two bytes, indicating the vertical value size.

ASCII: The data of the return point is displayed in scientific notation, and the data is separated by commas, for example, +3.590104E-02,-7.180208E-02,-7.180208E-02, +0.000000E+00,-3.590104E-02 ,-3.590104E-02,-7.180208E-02, .

### **Return format**

The query returns "WORD" or "ASCII".

### **Example**

The following command selects WORD mode.

```
:WAVeform:FORMat WORD
```

The following query returns "WORD".

```
:WAVeform:FORMat?
```

### 3.2.15.4 :WAVeform:START

**This function** sets or queries the starting position of the read data.

**Format** :WAVeform:START < no >

:WAVeform:START?

Among them , <no> , integer , the value is related to the set FORMat type

In NORMAL mode: 1 to the maximum horizontal pixel value of the screen waveform area (pixels per grid \* horizontal grid number)

MAX: 1 to the number of valid points on the current screen

RAW: 1 to the maximum value of the current memory depth

#### **illustrate**

When the amount of data on the screen is large, it usually cannot be read all at once and needs to be read several times. At this time, it is necessary to set the start and end points of each reading.

#### **Return Format**

The query returns an integer value .

#### **Example**

The following command sets the starting point to 500 .

:WAVeform:START 500

The following query returns "500".

```
:WAVeform:START?
```

### **3.2.15.5 :WAVeform:STOP**

**This function** sets or queries the end position of reading data.

**Format** :WAVeform:STOP < no >

```
:WAVeform:STOP?
```

Among them , <no> , integer , the value is related to the set FORMat type

In NORMAL mode: 1 to the maximum horizontal pixel value of the screen waveform area (pixels per grid \* horizontal grid number)

MAX: 1 to the number of valid points on the current screen

RAW: 1 to the maximum value of the current memory depth

#### **illustrate**

When the amount of data on the screen is relatively large, it usually cannot be read at once and needs to be read several times. At this time, it is necessary to set the starting point and end point of each reading. The value of the end point must be greater than or equal to the value of the starting point.

#### **Return format**

The query returns an integer value .

## Example

The following command sets the starting point to 1000 .

```
:WAVeform:STOP 1000
```

The query below returns "1000".

```
:WAVeform:STOP?
```

### 3.2.15.6 :WAVeform:DATA:<type>?

**Function:** Quickly reads all waveform data displayed on the screen.

#### Format:

:WAVeform:DATA:HEX? Returns hexadecimal characters

:WAVeform:DATA:BIN? Returns binary data

:WAVeform:DATA:AScii? Returns waveform data

#### Return Format:

The returned data consists of four parts: an identifier, a data-length descriptor, the data length, and the waveform data.

```
#MdddddddXXXX
```

In this format, # is the identifier. M indicates that the following M digits specify the total byte length of the returned waveform data, represented by dddddddd. The subsequent XXXX contains the waveform data.

**Note:** The amount of returned data is large; ensure that the receive buffer is configured with sufficient size.

**Example Procedure for Fast Full Waveform Data Acquisition:**

```
:WAVeform:SOURce CH1
```

```
:WAVeform:MODE NORMal
```

```
:WAVeform:DATA:BIN?
```

**Note:** WAVeform:DATA:<type>? is supported only on 12-bit oscilloscopes.

**3.2.15.7 :WAVeform:DATA?**

**Function** reads waveform data.

**Format** :WAVeform:DATA?

This command is affected by :WAVeform:SOURce, :WAVeform:FORMat, :WAVeform:MODE and other command settings.

**illustrate**

Screen waveform data reading process:

S1. :WAV:SOURce CH1	Sets the source for reading
---------------------	-----------------------------

S2. :WAV:MODE NORM	The waveform mode is NORM
S3. :WAV:FORMat BYTE	Set the data return format to BYTE
:WAV:DATA?	Get the data on the screen
Memory waveform data reading:	
S1. :MENU:STOP	Memory waveform can only be read in stop state
S2. :WAV:SOURce CH1	Set the source of reading
S3. :WAV:MODE	RAW waveform mode is RAW
S4. :WAV:FORMat BYTE	Sets the data return format to WORD
S5.:WAVeform:STARt 1	Set the starting position of the read point to 1
S6. :WAVeform:STOP 62500	Set the end position of the read point to 62500
S7. :WAV:DATA?	Get the data in the buffer

### Return

When :WAV:FORM is set to WORD,

The returned data consists of 4 parts: identifier, data length description, data length, and waveform data.

#M ddd ddddXXXX

one of them # Identifier

M represents the first M bits of the following data, which describes the total number of bytes of the returned waveform data, expressed as ddd dddd; The following X XXX is the waveform data

For example:

sending the :WAV:DATA? command, the data is returned.

# 90 00 00 10 24 80 81 82 83 89 .....

# is an identifier 9 The 9 bits following it represent the number of sampling points of the returned data. 0 00 00 10 24 is a total of 9 bits of data, indicating that the data volume is 1024 sampling points.

Notice:

If the amount of data in the memory is large, the user needs to read it in multiple times, read a block of data each time, and then connect the data read each time. The amount of data read each time is determined by the set data return format (:WAV:FORMat ), as shown in the following table:

Setting format	Maximum amount of data that can be read at one time	Remark
WORD	62,5000 sampling points	
ASCII	15625	

For example

The amount of memory data is 220K , and the data return format is set to WORD ; since in WORD mode, the amount of data read each time is 62,500 , and a total of 220,000 data needs to be read, it should be read 4 times;



S6.:WAVeform:STOP 187500      Set the end position of the read point to 187500

S7.:WAV:DATA?      Get the third segment of data

Read the fourth piece of data

S5.:WAVeform:STARt 187501      Sets the starting position of the reading point to 187501

S6.:WAVeform:STOP 220000      Sets the end position of the reading point to 220000

S7.:WAV:DATA?      Get the fourth segment of data

Connect all segments to get the data of the entire memory.

### ***3.2.15.8 :WAVeform:PREamble?***

**Command format** :WAVeform:PREamble?

**Function description** Query and return all waveform parameters.

Return format

The query returns 9 waveform parameters separated by " , ":

<format>,<type>,<count>,<xincrement>,<xorigin>,<xreference>,<yincrement>,<yorigin>,<yreference>

<format>: 0 (WORD) or 2 (ASCII) . Refer to :WAVeform:FORMat command .

<type>: 0 (NORMAL), 1 (MAXimum) or 2 (RAW). Refer to :WAVeform:MODE command .

<count>: The average number of times in average sampling mode (refer to :ACQuire:AVERages command ), and 1 in other modes .

<xincrement>: The time difference between two adjacent points in the X direction.

Reference :WAVeform:XINCrement? command .

---

<xorigin>: The time from the trigger point to the "reference time base" in the X direction.

Reference: :WAVeform:XORigin? command.

<xreference>: The reference time base of the data point in the X direction.

Reference : :WAVeform:XREFerence? command.

<yincrement >: Unit voltage value in the Y direction. Reference : :WAVeform:YINCrement?

command.

<yorigin >: The vertical offset in the Y direction relative to the " vertical reference position"

(reference : :WAVeform:YREFerence? command). Reference : :WAVeform:YORigin? command.

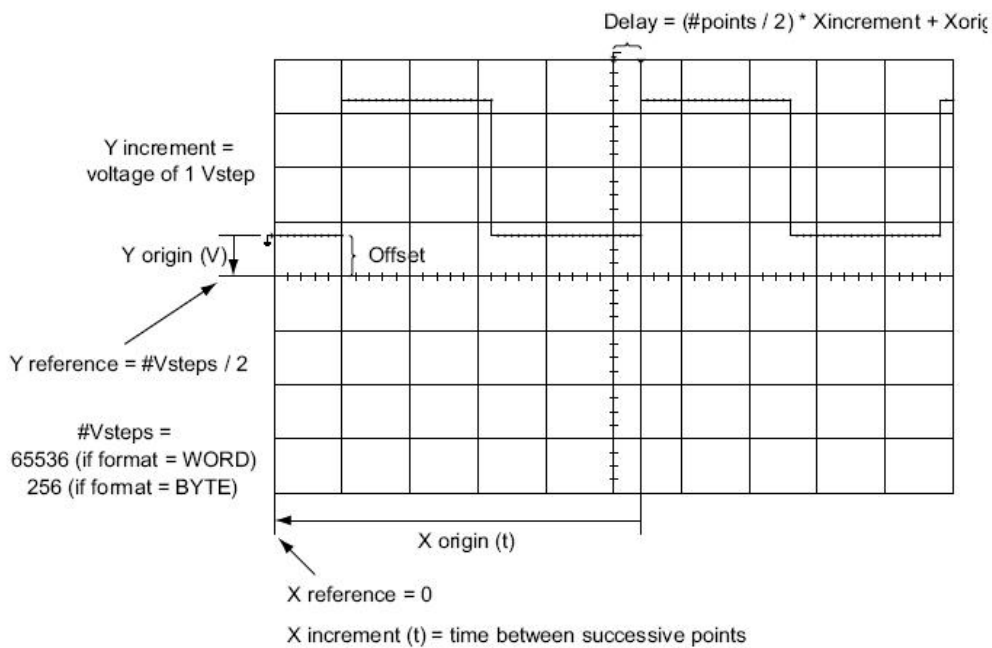
<yreference >: The vertical reference position in the Y direction.

Reference : :WAVeform:YREFerence? command.

#### Example

The following query returns " 1,2,1,0.000000,-0.001488,0,0.062500,3.968750,127 ".

:WAVeform:PREamble?



### 3.2.15.9:WAVeform:XINCrement?

Format :WAVeform:XINCrement?

**Functional Description** Query the time interval between two adjacent points in the X direction of the currently selected channel source.

The return value is related to the current data reading mode:

In NORMAL mode, XINCrement = 1/ mathematical waveform sampling rate.

In RAW mode, XINCrement = 1/ SampleRate .

MAX mode, when the instrument is running, XINCrement = 1/ mathematical waveform sampling rate; when the instrument is stopped, XINCrement = 1/ SampleRate .

Units are relative to the current channel source.

Return format

The query returns the time interval in scientific notation .

Example

The query below returns " 2.000000e-08 ".

:WAVeform:XINCrement?

### **3.2.15.10:WAVeform:XORigin?**

Command format :WAVeform:XORigin?

Function description

Query the specified source (Reference:WAVeform:SOURce command) The first waveform point in the X direction, the time to the trigger position (calculated with the trigger position as the base 0), the unit is s .

The return value is related to the current data reading mode:

NORMAl mode, returns the time from the first waveform point on the screen to the trigger position.

RAW mode, returns the time from the first waveform point in the memory to the trigger position.

MAX mode, when the instrument is in the running state, it returns the time from the first waveform point on the screen to the trigger position; when the instrument is in the stopped state, it returns the time from the first waveform point in the memory to the trigger position.

Return Format

The query returns the time value in scientific notation .

Example

The following query returns "-7.000000e-06".

:WAVeform:XORigin?

### **3.2.15.11 :WAVeform:XREFerence?**

Command format :WAVeform:XREFerence?

Function description

Query the specified source (reference:WAVeform:SOURce command) the reference time base of data points in the X direction. The unit is s , using scientific notation, the same as above .

Return format

The query returns the time base as an integer .

Example

The following query returns " 0 " .

:WAVeform:XREFerence?

### **3.2.15.12 :WAVeform:YINCrement?**

Format :WAVeform:YINCrement?

Functional Description

Query the unit voltage value in the Y direction of the specified source (reference:WAVeform:SOURce command) . The unit is consistent with the selected source unit.

Return Format

The query returns the voltage value in scientific notation .

Example

The query below returns "3.125000e-03 V".

:WAVeform:YINCrement?

### **3.2.15.13:WAVeform:YORigin?**

Format :WAVeform:YORigin?

Functional Description

Query the vertical offset of the specified source ( reference:WAVeform:SOURce command) in the Y direction relative to the "vertical reference position" (reference:WAVeform:YREFerence? command). The unit is consistent with the unit selected for the source.

Return format

The query returns the offset value in scientific notation .

Example

The query below returns " 3.968750e+00 V ".

:WAVeform:YORigin?

### **3.2.15.14:WAVeform:YREFerence?**

Command format :WAVeform:YREFerence?

Function description

Query the vertical reference position of the specified source (refer to :WAVeform:SOURce command) in the Y direction . The units are consistent with the units selected by the source.

Return format

The query returns the reference position as an integer .

Example

The query below returns "127".

:WAVeform:YREFerence?

**Contact us**

Email: sales@micsig.com

Company: Shenzhen Micsig Technology Co., Ltd.

Address: 6F, Jinhuan Building, No. 56, Tiezai Road, Bao'an District,  
Shenzhen, Guangdong, China.

**This manual is subject to change without prior notice .**

**The contents of this manual are believed to be correct. If the user finds any errors, omissions, etc., please contact Micsig .**

**The company is not responsible for accidents and damages caused by user's incorrect operation .**

**The copyright of this manual belongs to Micsig. No unit or individual may reproduce, copy or extract it without the authorization of Micsig. Micsig reserves the right to prosecute for the above acts.**