UNI-T®

Instruments.uni-trend.com

# Programming Manual

UPO1000 Series Programmable Digital Oscilloscope

# Warranty and Statement

**File Number**
20230822

**Software Version**
V1.00.0018
Software upgrade may have some change and add more function, please subscribe UNI-T website to get the newest manual or contact UNI-T to update the version.

**Product Certification**
UNI-T has certified that the product conforms to China national product standard and industry product standard as well as ISO9001:2008 standard and ISO14001:2004 standard. UNI-T will go further to certificate product to meet the standard of other member of the international standards organization.

## Contact Us

If you have any question or problem, please contact UNI-T.
Mail: infosh@uni-trend.com.cn
Official Website：https://www.uni-trend.com

# 1. SCPI Introduction

SCPI (Standard Commands for Programmable Instruments) is a standardized instrument programming language that builds on existing standards IEEE 488.1 and IEEE 488.2 and follows the floating point rules of IEEE 754 standard, ISO 646 message exchange 7-bit encoding notation (equivalent to ASCII programming) and many other standards.

This section introduces the format, symbols, parameters, and abbreviations of the SCPI command.

## Instruction Format

The SCPI command is a tree-like hierarchy consisting of multiple subsystems, each consisting of a root keyword and one or more hierarchical key words. The command line usually begins with a colon ":"; Keywords are separated by the colon ":", followed by optional parameter settings. The command keyword is separated by spaces from the first parameter. The command string must end with a newline <NL> character. Add the question mark "? " after the command line. It is usually indicated that this feature is being queried.

## Symbol Description

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

- Brace { } usually contains multiple optional parameters, it should select one parameter when send command.

  Such as DISPlay:GRID:MODE { FULL | GRID | CROSS | NONE} command
- Vertical Bar | used to separated multiple parameters, it should select one parameter when send command.
- Square Brackets [ ] the contents in square brackets (command keywords) can omissible. If the parameter is ignored, the instrument will set the parameter as the default value.

  Such as MEASure:NDUTy? [<source>] command, it represents current channel
- Triangular Brackets < > The parameter in the brackets must be replaced with a valid value.

  Such as use DISPlay:GRID:BRIGhtness 30 form to send DISPlay:GRID:BRIGhtness <count> command

## Parameter Description

The parameter in this manual can divide into five types: Boolean, Integer, Real, Discrete, ASCII string

- Boolean

  Parameter value can set "ON" (1) or "OFF" (0)

Such as SYSTem:LOCK {{1 | ON} | {0 | OFF}}

- Integer

  Parameter can take any valid integer value unless there have some other descriptions.

  Such as command: DISPlay:GRID:BRIGhtness <count> , parameter of <count> can take integer from 0~100

  Note: Do not set decimal as parameter, otherwise it may occur error.

- Real

  Parameter can take any valid integer value unless there have some other descriptions.

  Such as for command CH1, CHANnelI: OFFSet <offset>, parameter of <offset> can take integer value.

- Discrete

  Parameter can only take some specified numbers or characters.

  Such as command DISPlay:GRID:MODE { FULL | GRID | CROSS | NONE} parameter can only take FULL, GRID, CROSS, NONE

- ASCII Character String

  String parameter contain all ASCII string sets. Strings must begin and end with paired quotes; it can use single or double quotation marks. The quotation and delimiter can also be part of a string by typing it twice and not adding any characters.
  Such as set IP SYST:COMM:LAN:IPAD "192.168.1.10"

## Shorthand Rule

All command can er capital and small letter, if command need enter shorthand, it should be all capital letter.

## Data Return

Data return is divided into single data and batch data. The single data return is the corresponding parameter type, in which the real return type is express in scientific notation. The part before e retains three figure behind the decimal point, and the e part retains three figure; the batch return must be obey IEEE
488.2# string data format, '#'+ the length of character bits [fixed to one character] + ASCII valid value+ valid data+ end string['\n']
Such as #3123xxxxxxxxxxxxxxxxxxxx\n represents 123 strings batch data return format, '3' represents "123" occupies three character bits.
Note: If return data is invalid data, use * to represent it.

# 2. SCPI Command Explanation

## IEEE488.2 Common Command

### *IDN?

- **Command format:**
  *IDN?
- **Functional description:**
  For querying manufacture name, model, product serial number and software version.
- **Return format:**
  Manufacture name, model, product serial number, software version separated by dot mark.
- **For example:**
  UNI-T Technologies, UPO1000X, UPO1000, 00.00.01

### *RST

- **Command format:**
  *RST
- **Functional description:**
  Restore factory settings and clear the entire error message, send and receive queue buffers.

### *OPC

- **Command format:**
  *OPC
  *OPC?
- **Functional description:**
  This command is used to force the current instruction to complete and marked as the mark position 1.
- **Return format:**
  Query returns whether the current send instruction is executed, 1 represents it's completed, 0 represents it's not complete.
- **For example:**

| | |
|---|---|
| *OPC | Mark the executed instruction at mark position 1. |
| *OPC? | Query returns 1, it represents the command is executed. Otherwise, it's not. |

## SYSTem Command

The command is used for the basic operation of oscilloscope, including operating control, full keyboard lock, error queue and system data setting.

### :RUN

- **Command format:**
  :RUN
- **Functional description:**
  This command is used to start the sampling operating of the oscilloscope, if it need to stop, executing: STOP command.

## :STOP

- **Command format:**
  :STOP
- **Functional description:**
  This command is used to stop the sampling operating of the oscilloscope, if it need to restart, executing: RUN command.

## :AUTO

- **Command format:**
  :AUTO
- **Functional description:**
  This command is used to automatically set the control parameter of the instrument, the automatic setting can make the best display effect for the input waveform.

## :SYSTem:LOCK

- **Command format:**
  :SYSTem:LOCK {{1 | ON} | {0 | OFF}}
  :SYSTem:LOCK?
- **Functional description:**
  This command is used to lock/unlock full keyboard.
- **Return format:**
  Query returns full keyboard locked status, 0 represents locked, 1 represents unlock.
- **For example:**

| | |
|---|---|
| :SYSTem:LOCK ON/:SYST:LOCK 1 | Full keyboard locked. |
| :SYSTem:LOCK OFF/:SYST:LOCK 0 | Unlock full keyboard. |
| :SYSTem:LOCK? | Query returns 1, it represents locked. |

## :SYSTem:ERRor

- **Command format:**
  :SYSTem:ERRor
  :SYSTem:ERRor?
- **Functional description:**
  This command is used to empty error message queue.
- **Return format:**
  Query returns the last error message, Query returns error message in the format of "<Message number>, <Message content>. <Message number> is an integer; <Message content> is an ASCII character string with double quotation marks.
  Such as-113,"Undefined header; command cannot be found".
- **For example:**

| | |
|---|---|
| :SYSTem:ERR | Empty error message queue. |
| :SYSTem:ERR? | Query returns -113,"Undefined header; command cannot be found". |

## :SYSTem:SETup

- **Command format:**
  :SYSTem:SETup <setup_data>
  :SYSTem:SETup?
- **Functional description:**
  This command is used to configure the system setting data. <setup_data> is conform with Appendix 2:

IEEE 488.2 Binary Data Format.

- Return format:
  Query returns the system setting data.

## :SYSTem:LANGuage

- Command format:
  :SYSTem:LANGuage { ENGLish | SIMPlifiedchinese }
  :SYSTem:LANGuage?
- Functional description:
  This command is used to set the system lanauage.
- Return format:
  Query returns { ENGLish | SIMPlifiedchinese }.
- For example:

| | |
|---|---|
| :SYSTem:LANGuage ENGL | Set the system language to English. |
| :SYSTem:LANGuage? | Query returns ENGLish. |

## :SYSTem:RTC

- Command format:
  :SYSTem:RTC <year>,<month>,<day>,<hour>,<minute>,<second>
  :SYSTem:RTC?
- Functional description:
  This command is used to set the system time.
- Return format:
  Query returns year, month, date, hour, minute and second.
- For example:

| | |
|---|---|
| :SYSTem:RTC 2017,7,7,20,8,8 | Set the system time to 20:08:08, 7th July, 2017. |
| :SYSTem:RTC? | Query returns 2017,7,7,20,8,8. |

## :SYSTem:CAL

- Command format:
  :SYSTem:CAL
- Functional description:
  This command is used to set the self-calibration of the system, it cannot be normal communicated during self-calibration.

## :SYSTem:CLEAr

- Command format:
  :SYSTem:CLEAr
- Functional description:
  Empty all the saved waveforms and data settings.

## :SYSTem:CYMOmeter

- Command format:
  :SYSTem:CYMOmeter {1 | ON} | {0 | OFF}
  :SYSTem:CYMOmeter?
- Functional description:
  This command is used to turn on/off the frequency meter.
- Return format:
  Query returns the status of frequency meter, 1 represets ON, 0 represents OFF.
- For example:

| | |
|---|---|
| :SYSTem:CYMOmeter ON | Turn on frequency meter. |
| :SYSTem:CYMOmeter? | Query returns 1. |

### :SYSTem:CYMOmeter:FREQuency?

- **Command format:**
  :SYSTem:CYMOmeter:FREQuency?
- **Functional description:**
  This command is used to acquire the frequency value of the frequency meter measurement.
- **Return format:**
  Query returns the frequency value of the frequency meter measurement, and invalid value returns *.
- **For example:**
  :SYSTem:CYMOmeter:FREQuency?    Query returns 1.20000E+3.

### :SYSTem:SQUare:SELect

- **Command format:**
  :SYSTem:SQUare:SELect  { 10 Hz | 100 Hz | 1 kHz | 10 kHz }
  :SYSTem:SQUare:SELect?
- **Functional description:**
  This command is used to select square wave output.
- **Return format:**
  Query returns { 10 Hz | 100 Hz | 1 kHz | 10 kHz }.
- **For example:**

| | |
|---|---|
| :SYSTem:SQUare:SELect 10 Hz | Select 10 Hz square wave output. |
| :SYSTem:SQUare:SELect? | Query returns 10 Hz. |

### :SYSTem:OUTPut:SELect

- **Command format:**
  :SYSTem:OUTPut:SELect  { TRIGger | PASS_FAIL | DVM }
  :SYSTem:OUTPut:SELect?
- **Functional description:**
  This command is used to set output selection TRIGger , PASS_FAIL (pass&fail) or DVM.
- **Return format:**
  Query returns {TRIGger | PASS_FAIL| DVM}.
- **For example:**

| | |
|---|---|
| :SYSTem:OUTPut:SELect TRIG | Output selection sets to trigger. |
| :SYSTem:OUTPut:SELect? | Query returns TRIG. |

### :SYSTem:BOOT:LOAD

- **Command format:**
  :SYSTem:BOOT:LOAD  {DEFault | LAST }
  :SYSTem:BOOT:LOAD?
- **Functional description:**
  This command is used to set boot loading mode. DEFault represents default, LAST represents the last setting.
- **Return format:**
  Query returns {DEFault | LAST }.
- **For example:**

| | |
|---|---|
| :SYSTem:BOOT:LOAD LAST | Boot loading the last setting. |
| :SYSTem:BOOT:LOAD? | Query returns LAST. |

## :SYSTem:POWer:MODe

- Command format:
  :SYSTem:POWer:MODe {OPEN | CLOSe}
  :SYSTem:POWer:MODe?
- Functional description:
  This command is used to set power-on mode to always open or always shuntdown. OPEN represents always open, CLOSe represents always shuntdown.
- Return format:
  Query returns {OPEN | CLOSe}.
- For example:

| | |
|---|---|
| :SYSTem:POWer:MODe OPEN | Set the power-on mode to always open. |
| :SYSTem:POWer:MODe? | Query returns OPEN. |

## :SYSTem:MNUDisplay

- Command format:
  :SYSTem:MNUDisplay { 5S | 10S | 20S | INFinite}
  :SYSTem:MNUDisplay?
- Functional description:
  This command is used to set the menu display time, INFinite represents the menu is always displayed.
- Return format:
  Query returns {5S | 10S | 20S | INFinite}.
- For example:
  :SYSTem:MNUDisplay 5S
  Set the menu display time to 5s, the menu will automatically withdraw after 5s.
  :SYSTem:MNUDisplay? Query returns 5s.

## :SYSTem:BRIGhtness

- Command format:
  :SYSTem:BRIGhtness <count>
  :SYSTem:BRIGhtness?
- Functional description:
  This command is used to set the screen brightness, <count> takes value from 1~100, the bigger the number, the brighter the screen.
- Return format:
  Query returns the current screen brightness.
- For example:

| | |
|---|---|
| :SYSTem:BRIGhtness 50 | Set screen brightness to 50. |
| :SYSTem:BRIGhtness? | Query returns 50. |

## :SYSTem:VERSion?

- Command format:
  :SYSTem:VERSion?
- Return format:
  Query returns version information, which are 128 bytes character string.
  HW is hardware version number, SW is software version number, PD is created date, ICV is protocol version number.
- For example:
  :SYST:VERS? Query returns HW:1.0;SW:1.0;PD:2014-11-20;ICV:1.4.0.

## :SYSTem:COMMunicate:LAN:APPLy

- Command format:

:SYSTem:COMMunicate:LAN:APPLy

- **Functional description:**
  This command is used to take effect the current network parameter immediately.

## :SYSTem:COMMunicate:LAN:GATEway

- **Command format:**
  :SYSTem:COMMunicate:LAN:GATEway <gateway>
  :SYSTem:COMMunicate:LAN:GATEway?
- **Functional description:**
  This command is used to set the default gateway. <gateway> is belong to the parameter of ASCII character string, the format is xxx.xxx.xxx.xxx.
- **Return format:**
  Query returns the default gateway.
- **For example:**

| | |
|---|---|
| :SYST:COMM:LAN:GATE "192.168.1.1" | Set the default gateway to 192.168.1.1. |
| :SYST:COMM:LAN:GATE? | Query returns192.168.1.1. |

## :SYSTem:COMMunicate:LAN:SMASK

- **Command format:**
  :SYSTem:COMMunicate:LAN:SMASK <submask>
  :SYSTem:COMMunicate:LAN:SMASK?
- **Functional description:**
  This command is used to set subnet mask. <submask> is belong to the parameter of ASCII character string, the format is xxx.xxx.xxx.xxx.
- **Return format:**
  Query returns subnet mask.
- **For example:**

| | |
|---|---|
| :SYST:COMM:LAN:SMASK "255.255.255.0" | Set subnet mask to 255.255.255.0. |
| :SYST:COMM:LAN:SMASK? | Query returns 255.255.255.0. |

## :SYSTem:COMMunicate:LAN:IPADdress

- **Command format:**
  :SYSTem:COMMunicate:LAN:IPADdress <ip>
  :SYSTem:COMMunicate:LAN:IPADdress?
- **Functional description:**
  This command is used to set IP address. <ip> is belong to the parameter of ASCII character string, the format is xxx.xxx.xxx.xxx.
- **Return format:**
  Query returns IP address.
- **For example:**

| | |
|---|---|
| :SYST:COMM:LAN:IPAD "192.168.1.10" | Set IP address to 192.168.1.10 |
| :SYST:COMM:LAN:IPAD? | Query returns 192.168.1.10. |

## :SYSTem:COMMunicate:LAN:DHCP

- **Command format:**
  :SYSTem:COMMunicate:LAN:DHCP {{1 | ON} | {0 | OFF}}
  :SYSTem:COMMunicate:LAN:DHCP?
- **Functional description:**
  This command is used to switch the configuration mode to automatic IP or manual IP.
- **Return format:**
  Query returns dynamic allocation mode, 0 represents manual IP, 1 represents automatic IP.
- **For example:**

| | |
|---|---|
| :SYST:COMM:LAN:DHCP ON | Turn on IP dynamic allocation. |
| :SYST:COMM:LAN:DHCP? | Query returns 1. |

## :SYSTem:COMMunicate:LAN:MAC?

- **Command format:**
  :SYSTem:COMMunicate:LAN:MAC?
- **Return format:**
  Query returns MAC physical address.
- **For example:**

| | |
|---|---|
| :SYST:COMM:LAN:MAC? | Query returns 00-2A-A0-AA-E0-56. |

## :SYSTem:COMMunicate:RS232:BAUDrate

- **Command format:**
  :SYSTem:COMMunicate:RS232:BAUDrate <baudrate>
  :SYSTem:COMMunicate:RS232:BAUDrate?
- **Functional description:**
  This command is used to set the baud rate for RS232 communication port of the oscilloscope. The parameter is integer type, and the range is 600~115200.
- **Return format:**
  Query returns baud rate.
- **For example:**

| | |
|---|---|
| :SYST:COMM:RS232:BAUD 115200 | Set baud rate of RS232 to 115200b/s. |
| :SYST:COMM:RS232:BAUD? | Query returns 1.15200. |

## :SYSTem:COMMunicate:RS232:STOP:BIT

- **Command format:**
  :SYSTem:COMMunicate:RS232:STOP:BIT {1 | 2 }
  :SYSTem:COMMunicate:RS232:STOP:BIT?
- **Functional description:**
  This command is used to set the stop bit for RS232 communication port of the oscilloscope.
- **Return format:**
  Query returns {1 | 2 }.
- **For example:**

| | |
|---|---|
| :SYST:COMM:RS232:STOP:BIT 1 | Set stop bit of RS232 to 1. |
| :SYST:COMM:RS232:STOP:BIT? | Query returns 1. |

## :SYSTem:COMMunicate:RS232:PARity

- **Command format:**
  :SYSTem:COMMunicate:RS232:PARity {EVEN | ODD | NONE}
  :SYSTem:COMMunicate:RS232:PARity?
- **Functional description:**
  This command is used to set the parity for RS232 communication port of the oscilloscope.
- **Return format:**
  Query returns {EVEN | ODD | NONE}.
- **For example:**

| | |
|---|---|
| :SYST:COMM:RS232:PAR ODD | Set parity of RS232 bus to ODD. |
| :SYST:COMM:RS232:PAR? | Query returns ODD. |

### :SYSTem:COMMunicate:RS232:APPLy

- **Command format:**
  :SYSTem:COMMunicate:RS232:APPLy
- **Functional description:**
  The current setting of RS232 parameter will be take effect immediately.

## KEY Command

This command is used to control the key and rotary knob on the operating panel of the oscilloscope.

### :KEY:<key>

- **Command format:**
  :KEY:<key>
  :KEY:<key>:LOCK { {1 | ON} | {0 | OFF} }
  :KEY:<key>:LOCK?
  :KEY:<key>:LED?
- **Functional description:**
  This command is used to set key function and lock/unlock function. <key> definition and description refer to Appendix 1：Key List
- **Return format:**
  Query returns key status or LED indicator status;
  Lock status: 0 represents the key is unlock, 1 represents the key is locked;
  LED status： 0 represents LED is extinguished, 1 represents LED is illuminated.
- **For example:**
  :KEY:AUTO
  Automatically set the control parameter of the oscilloscope.
  :KEY:AUTO:LOCK ON/OFF Lock/unlock key.
  :KEY:AUTO:LOCK? Query returns key status, 1 represents the key is locked.
  :KEY:AUTO:LED?
  Query returns LED status, 0 represents LED is extinguished.

## CHANnel Command

This command is used to set each channel independently.

### :CHANnel<n>:BWLimit

- **Command format:**
  :CHANnel<n>:BWLimit {20M | FULL}
  :CHANnel<n>:BWLimit?
- **Functional description:**
  This command is used to set the bandwidth limit.
  20M: open the bandwidth limit 20MHz for reducing displayed noise.
  FULL: reach to full bandwidth display.
  <n>: {1|2|3|4}, it represents {CH1|CH2|CH3|CH4}.
- **Return format:**
  Query returns {20M | FULL}.
- **For example:**
  :CHAN1:BWL 20M Open the bandwidth limit 20MHz of CH1.
  :CHAN1:BWL? Query returns 20M.

### :CHANnel<n>:COUPling

- Command format：
  :CHANnel<n>:COUPling {DC|AC|GND}
  :CHANnel<n>:COUPling?
- Functional description：
  This command is used to set the channel coupling mode. DC represents the AC and DC component of input signal can be through; AC represents the DC component of the input signal will be blocked; GND represents cut-off the input signal.

  <n>: {1|2|3|4}, it represents {CH1|CH2|CH3|CH4}.
- Return format：
  Query AC, DC or GND.
- For example：

| | |
|---|---|
| :CHAN1:COUP DC | Set the coupling mode of CH1 to DC. |
| :CHAN1:COUP? | Query returns DC. |

:CHANnel<n>:DISPlay

- Command format：
  :CHANnel<n>:DISPlay { {1|ON} | {0|OFF} }
  :CHANnel<n>:DISPlay?
- Functional description：
  This command is used to turn on/off the specified channel.
  <n>: {1|2|3|4}, it represents {CH1|CH2|CH3|CH4}.
- Return format：
  Query returns 1 or 0，it represents ON or OFF separately.
- For example：
  :CHAN1:DISP ON　　　　Turn on CH1.
  :CHAN1:DISP?
  Query returns 1, it represents the CH1 is turned on.

:CHANnel<n>:INVert

- Command format：
  :CHANnel<n>:INVert { {1|ON} | {0|OFF} }
  :CHANnel<n>:INVert?
- Functional description：
  This command is used to turn on/off the waveform phase reversed function.
  ON: turn on the waveform phase reversed function.
  OFF: restore the waveform to normal display.
  <n>: {1|2|3|4}, it represents {CH1|CH2|CH3|CH4}.
- Return format：
  Query returns 1 or 0, it represents ON or OFF.
- For example：
  :CHAN1:INV OFF　　　　Turn off the phase reversed of CH1.
  :CHAN1:INV?
  Query returns 0, it represents the phase reversed of CH1 is turned off.

:CHANnel<n>:PROBe:TYPe

- Command format：
  :CHANnel<n>:PROBe:TYPe　{ VOLTs | AMPeres }
  :CHANnel<n>:PROBe:TYPe?
- Functional description：
  This command is used to set the probe type. VOLTs represents voltage. AMPeres represents current.
  <n>: {1|2|3|4}, it represents {CH1|CH2|CH3|CH4}.

- **Return format:**
  Query returns the probe type of the oscilloscope.
- **For example:**

| | |
|---|---|
| :CHAN1:PROB:TYP AMPeres | Set the probe type of CH1 to current probe. |
| :CHAN1:PROB:TYP? | Query returns AMPeres. |

### :CHANnel<n>:PROBe

- **Command format:**
  :CHANnel<n>:PROBe { <probe> }
  :CHANnel<n>:PROBe?
- **Functional description:**
  This command is used to set the probe attenuation factor, the range can set to 0.001X~20000X.
  <probe>: Self-defined probe attenuation factor.
  <n>: {1|2|3|4}, it represents {CH1|CH2|CH3|CH4}.
- **Return format:**
  Query returns the probe attenuation factor of the oscilloscope and the currently probe attenuation factor in scientifc notation, the unit is X.
- **For example:**

| | |
|---|---|
| :CHAN1:PROB 10X | Set the probe attenuation factor of CH1 to 10X. |
| :CHAN1:PROB? | Query returns 1.000000e+01. |

### :CHANnel<n>:OFFSet

- **Command format:**
  :CHANnel<n>:OFFSet <offset>
  :CHANnel<n>:OFFSet?
- **Functional description:**
  This command is used to set the waveform offset on vertical direction.
  <n>: {1|2|3|4|5|6|7|8|9}, it represents {CH1|CH2|CH3|CH4|MATH|REFA|REFB|REFC|REFD} respectively.
- **Return format:**
  Query returns the setting value of offset in scientific notation, unit is V.
- **For example:**

| | |
|---|---|
| :CHAN1:OFFS 20V | Set the vertical offset of channel 1 to 20 V. |
| :CHAN1:OFFS? | Query returns 2.000e001. |

### :CHANnel<n>:SCALe

- **Command format:**
  :CHANnel<n>:SCALe {<scale> | UP | DOWN}
  :CHANnel<n>:SCALe?
- **Functional description:**
  This command is used to set volts/div scale on vertical direction.
  <scale>: Volts/div scale value;
  UP: Increase one scale based on the current scale of the oscilloscope;
  DOWN: Decrease one scale based on the current scale of the oscilloscope.
  <n>: {1|2|3|4|5|6|7|8|9}, it represents {CH1|CH2|CH3|CH4|MATH|REFA|REFB|REFC|REFD} respectively.
- **Return format:**
  Query returns the current volts/div scale vaule in scientific notation, unit is V.
- **For example:**

| | |
|---|---|
| :CHAN1:SCAL 20 V | Set the volts/div scale of channel 1 to 20 V. |
| :CHAN1:SCAL? | Query returns 2.000e001. |
| :CHAN1:SCAL UP | Increase one scale based on 20 V volts/div scale. |

### :CHANnel<n>:UNITs

- **Command format:**
  :CHANnel<n>:UNITs {VOLTs|AMPeres|WATTs|UNKNown}
  :CHANnel<n>:UNITs?
- **Functional description:**
  This command is used to set the channel unit to VOLTs, AMPeres, WATTs or UNKNown.<n>: {1|2|3|4}, it represents {CH1|CH2|CH3|CH4}.
- **Return format:**
  Query returns VOLTs, AMPeres, WATTs or UNKNown.
- **For example:**

| | |
|---|---|
| :CHAN1:UNIT VOLT | Set the unit of CH1 to VOLTs. |
| :CHAN1:UNIT? | Query returns VOLTs. |

### :CHANnel<n>:VERNier

- **Command format:**
  :CHANnel<n>:VERNier { {1|ON} | {0|OFF} }
  :CHANnel<n>:VERNier?
- **Functional description:**
  This command is used to set scale adjusting mode. If scale adjusting mode sets to ON, which is fine tuning. Fine tuning can be further subdivided within coarse tuning range for improving vertical resolution; if scale adjusting mode sets to OFF, which is coarse tuning. It can adjust vertical sensitivity by system 1-2-5.
  <n>: {1|2|3|4}, it represents {CH1|CH2|CH3|CH4}.
- **Return format:**
  Query returns 1 or 0, it represents ON or OFF.
- **For example:**

| | |
|---|---|
| :CHAN1:VERN ON | Turn on fine tuning of channel 1. |
| :CHAN1:VERN? | Query returns 1. |

### :CHANnel<n>:SELect

- **Command format:**
  :CHANnel<n>:SELect
  :CHANnel<n>:SELect?
- **Functional description:**
  This command is used to select channel.
  <n>: {1|2|3|4|5|6|7|8|9}, it represents {CH1|CH2|CH3|CH4|MATH|REFA|REFB|REFC|REFD} respectively.
- **Return format:**
  Query returns 1 or 0.
- **For example:**

| | |
|---|---|
| :CHAN1:SELect | Select CH1. |
| :CHAN1:SELect? | Query returns 1, it represents the channel is selected. |

## TIMebase Command

This command is to change the horizontal scale (timebase) of the current channel and horizontal position of the trigger in memory (trigger offset). Changing the horizontal scale may casue the waveform extended or compressed relative to the center of the screen; changing the horizontal position may casue waveform position away from the center of screen.

### :TIMebase:MODE

- **Command format:**

:TIMebase:MODE {MAIN | WINDow}
:TIMebase:MODE?

- **Functional description:**
  This command is used to set timebase mode, MAIN（main timebase）or WINDow（<Zoomed> timebase）.
- **Return format:**
  Query returns MAIN or WINDow.
- **For example:**

| | |
|---|---|
| :TIM:MODE MAIN | Set timebase mode to MAIN. |
| :TIM:MODE? | Query returns MAIN. |

### :TIMebase:OFFSet

- **Command format:**
  :TIMebase:OFFSet <offset>
  :TIMebase:OFFSet?
- **Functional description:**
  This command is used to adjust the offset of MAIN (main timebase), which is the offset of the waveform position from the center of the screen.
- **Return format:**
  Query returns <offset> value in scientific notation, unit is s.
- **For example:**

| | |
|---|---|
| :TIM:OFFS 1s | Set the offset of MAIN to 1s. |
| :TIM:OFFS? | Query returns 1.000e000. |

### :TIMebase:WINDow:OFFSet

- **Command format:**
  :TIMebase:WINDow:OFFSet <offset>
  :TIMebase:WINDow:OFFSet?
- **Functional description:**
  This command is used to adjust the offset of WINDow (<Zoomed> timebase), which is offset of the waveform position from the center of the screen.
- **Return format:**
  Query returns <offset> value in scientific notation, unit is s.
- **For example:**

| | |
|---|---|
| :TIM:WIND:OFFS 1 | Set the offset of WINDow to 1s. |
| :TIM:WIND:OFFS? | Query returns 1.000e000. |

### :TIMebase:SCALe

- **Command format:**
  :TIMebase:SCALe {<scale> | UP | DOWN}
  :TIMebase:SCALe?
- **Functional description:**
  This command is used to set timebase scale of MAIN (main timebase), which is s/div.
  <scale>: Timebase scale value;
  UP: Increase one scale based on the current scale of the oscilloscope;
  DOWN: Decrease one scale based on the current scale of the oscilloscope.
- **Return format:**
  Query returns < scale> value in scientific notation, unit is s/div.
- **For example:**

| | |
|---|---|
| :TIM:SCAL 2 | Set the offset of MAIN to 2 s/div. |
| :TIM:SCAL? | Query returns 2.000e000. |

### :TIMebase:WINDow:SCALe

- **Command format:**
  :TIMebase:WINDow:SCALe < scale >
  :TIMebase:WINDow:SCALe?
- **Functional description:**
  This command is used to set timebase scale of WINDow (<Zoomed> timebase), which is s/div.
- **Return format:**
  Query returns < scale> value in scientific notation, the unit is s/div.
- **For example:**

| | |
|---|---|
| :TIM:WIND:SCAL 2 | Set the offset of WINDow to 2 s/div. |
| :TIM:WIND:SCAL? | Query returns 2.000e000. |

### :TIMebase:HOLDoff

- **Command format:**
  :TIMebase:HOLDoff <time>
  :TIMebase:HOLDoff?
- **Functional description:**
  This command is used to set trigger holdoff time, which can set the range to 100ns~10s.
- **Return format:**
  Query returns the value of trigger holdoff time in scientific notation, unit is s.
- **For example:**

| | |
|---|---|
| :TIM:HOLD 1s | Set trigger holdoff time to 1s. |
| :TIM:HOLD? | Query returns 1.000e000. |

## FUNCtion Command

This command is used to display the operation result of the waveform of CH1, CH2. The operation is add, subtract, multiply, divide, and, or, not, XOR and FFT. Set filter to use the expression to operating.

### :FUNCtion:MATH:MODE

- **Command format:**
  FUNCtion:MATH:MODE {MATH| FFT| LOGic| FILTer| ADVance}
  FUNCtion:MATH:MODE?
- **Functional description:**
  This command is used to select MATH mode.
- **Return format:**
  Query returns {MATH|FFT|LOGic|FILTer|ADVance}.
- **For example:**

| | |
|---|---|
| FUNC:MATH:MODE FFT | Set MATH mode to FFT mode. |
| FUNC:MATH:MODE? | Query returns FFT. |

### :FUNCtion:OPERation

- **Command format:**
  :FUNCtion:OPERation {ADD | SUBTract | MULTiply | DIVide | AND | OR | NOT | XOR }
  :FUNCtion:OPERation?
- **Functional description:**
  This command is used to set functional operator, which includes the basic and logical operation. That is add, subtract, multiply and divide, and, or, not and XOR.

- **Return format:**
  Query returns {ADD | SUBTract | MULTiply | DIVide | AND | OR | NOT | XOR }.
- **For example:**

| | |
|---|---|
| :FUNCtion:OPERation ADD | Use add operator: src1+src2. |
| :FUNCtion:OPERation? | Query returns ADD. |

## :FUNCtion:DIGital<n>:THReshold

- **Command format:**
  :FUNCtion:DIGital<n>:THReshold <value>
  :FUNCtion:DIGital<n>:THReshold?
- **Functional description:**
  This command is used to set the logical threshold value of the specified channel, if it greater than the threshold value, then take value as 1, if it less than the threshold value, then take value as 0. n takes value as 1 or 2.
- **Return format:**
  Query returns 1.000e000, unit is V.
- **For example:**

| | |
|---|---|
| :FUNC:DIG1:THR 1 V | Set the logical threshold value of CH1 to 1 V. |
| :FUNC:DIG1:THR? | Query returns 1.000e000. |

## :FUNCtion:SOURce<m>

- **Command format:**
  :FUNCtion:SOURce<m> {CHANnel1| CHANnel2| CHANnel3| CHANnel4}
  :FUNCtion:SOURce<m>?
- **Functional description:**
  SOURce<m> represents source 1 or source 2, <m> takes value as 1 or 2.
  SOURce1 is used to select the first source for operator mathematical functions. And which can be a signle source for Filter, FFT.
  SOURce2 is used to select the second source for operator mathematical functions. It's not suitable for single source of Filter, FFT.
  <value> represents CHANnel<n>, <n> takes value from 1/2/3/4{CH1/ CH2/ CH3/ CH4}.
- **Return format:**
  Query returns CHANnel1, CHANnel2, CHANnel3 or CHANnel4.
- **For example:**

| | |
|---|---|
| :FUNCtion:SOUR1 CHAN1 | Set CH1 as the first source. |
| :FUNCtion:SOUR1? | Query returns CHANnel1. |
| :FUNCtion:SOUR2 CHAN2 | Set CH2 as the second source. |
| :FUNCtion:SOUR2? | Query returns CHANnel2. |
| :FUNCtion:OPERation ADD | Add source 1 and source 2. |

## :FUNCtion:FFT:WINDow

- **Command format:**
  :FUNCtion:FFT:WINDow {RECTangular|HANNing|HAMMing|BMAN|FLATtop}
  :FUNCtion:FFT:WINDow?
- **Functional description:**
  FFT adds window to intercept signal. RECT, HANN, HAMM, BMAN, FLAT is rectangular window, Hanning window, Hamming window, BlacKman window and flat-top window.
- **Return format:**
  Query returns { RECTangular|HANNing|HAMMing|BMAN|FLATtop }.
- **For example:**

| | |
|---|---|
| :FUNCtion:SOUR1 CHAN1 | Set CH1 as the source. |
| :FUNC:FFT:WIND HAMM | Add Hamming window. |
| :FUNC:FFT:WIND? | Query returns HAMMing. |

## :FUNCtion:FFT:DISPlay

- Command format:
  :FUNCtion:FFT:DISPlay {FULL| SPLIt| WATerfall1| WATerfall2| INDependent}
  :FUNCtion:FFT:DISPlay?
- Functional description:
  This command is used to set FFT display mode.
- Return format:
  Query returns {FULL| SPLIt| WATerfall1| WATerfall2| INDependent }. FULL is full display. SPLIt is split screen display. INDPendent represents independent display FFT. WATerfall1 is waterfall curve 1. WATerfall2 is waterfall curve 2.
- For example:

| | |
|---|---|
| :FUNCtion:FFT:DISPlay FULL | Set FFT to full display. |
| :FUNCtion:FFT:DISPlay? | Return FULL. |

## :FUNCtion:FFT:WATerfall:SLICe

## :FUNCtion:FFT:WATerfall:SLICe

- Command format:
  :FUNCtion:FFT:WATerfall:SLICe <value>
  :FUNCtion:FFT:WATerfall:SLICe ?
- Functional description:
  This command is used to select the segment of FFT waterfall curve, the selection range is 1~200.
- Return format:
  Query returns which frame segment is currently selected.
- For example:
  :FUNCtion:FFT:WATerfall:SLICe 100
  Set FFT waterfall curve to select 100 frame segment.
  :FUNCtion:FFT:WATerfall:SLICe ?        Query returns 100.

## :FUNCtion:FFT:POINts

- Command format:
  :FUNCtion:FFT:POINts {8K| 16K| 32K| 64K| 128K | 256K | 512K | 1M}
  :FUNCtion:FFT:POINts?
- Functional description:
  This command is used to set FFT point.
- Return format:
  Query returns {8K| 16K| 32K| 64K| 128K | 256K | 512K | 1M}.
- For example:

| | |
|---|---|
| :FUNCtion:FFT:POINts 8 K | Set FFT point to 8 K. |
| :FUNCtion:FFT:POINts? | Return 8 K. |

## :FUNCtion:FFT:VTYPe

- Command format:
  :FUNCtion:FFT:VTYPe {VRMS|DBRMS}
  :FUNCtion:FFT:VTYPe?
- Functional description:
  This command is used to select the unit of FFT vertical direction to dBRMS or VRMS. dBRMS represents power root mean square. VRMS represents voltage root mean square.
- Return format:
  Query returns {VRMS|DBRMS}.
- For example:

|  |  |
|---|---|
| :FUNCtion:SOUR1 CHAN1 | Set CH1 as the source. |
| :FUNC:FFT:VTYP VRMS | Set the unit of FFT vertical direction to VRMS. |
| :FUNC:FFT:VTYP? | Query returns VRMS. |

## :FUNCtion:FFT:FREQuency

- Command format:
  :FUNCtion:FFT:FREQuency?
- Functional description:
  This command is used to acquire the center freuquency of spectrum waveform after FFT.
- Return format:
  Query returns the center frequency of spectrum waveform, the unit is Hz.
- For example:

|  |  |
|---|---|
| :FUNCtion:FFT:FREQuency? | Query returns 1.000e003. |

## :FUNCtion:FFT:FREQuency:STARt

- Command format:
  :FUNCtion:FFT:FREQuency:STARt <freq>
  :FUNCtion:FFT:FREQuency:STARt?
- Functional description:
  This command is used to set the start frequency of FFT.
- Return format:
  Query returns 1.000e003, the unit is Hz.
- For example:

|  |  |
|---|---|
| :FUNCtion:FFT:FREQuency:STARt 1 kHz | Set the start frequency to 1 kHz. |
| :FUNC:FFT:FREQ:STARt? | Query returns 1.000e003. |

## :FUNCtion:FFT:FREQuency:END

- Command format:
  :FUNCtion:FFT:FREQuency:END <freq>
  :FUNCtion:FFT:FREQuency:END?
- Functional description:
  This command is used to set the end frequency of FFT.
- Return format:
  Query returns 1.000e003, the unit is Hz.
- For example:

|  |  |
|---|---|
| :FUNCtion:FFT:FREQuency:END 1 kHz | Set the end frequency of FFT to 1 kHz. |
| :FUNC:FFT:FREQ:END? | Query returns 1.000e003. |

## :FUNCtion:FFT:FREQuency:CENTer

- Command format:
  :FUNCtion:FFT:FREQuency:CENTer <freq>
  :FUNCtion:FFT:FREQuency:CENTer?
- Functional description:
  This command is used to set the center frequency of FFT.
- Return format:
  Query returns 1.000e003, the unit is Hz.
- For example:

|  |  |
|---|---|
| :FUNCtion:FFT:FREQuency:CENTer 1 kHz | Set the center frequency to 1 kHz. |
| :FUNC:FFT:FREQ:CENTer? | Query returns 1.000e003. |

## :FUNCtion:FFT:FREQuency:BW

- Command format:
  :FUNCtion:FFT:FREQuency:BW <freq>
  :FUNCtion:FFT:FREQuency:BW?
- Functional description:
  This command is used to set the frequency bandwidth of FFT.
- Return format:
  Query returns 1.000e003, the unit is Hz.
- For example:

| | |
|---|---|
| :FUNCtion:FFT:FREQuency:BW 1 kHz | Set the frequency bandwidth to 1 kHz. |
| :FUNC:FFT:FREQ:BW? | Query returns 1.000e003. |

## :FUNCtion:FFT:FREQuency:TRACk

- Command format:

  :FUNCtion:FFT:FREQuency:TRACk {{1 | ON} | {0 | OFF}}
  :FUNCtion:FFT:FREQuency:TRACk?
- Functional description:
  This command is used to set the frequency track switch of FFT.
- Return format:
  Query returns the status of the frequency track switch. 0 represents track is enabled, 1 represents is track is disabled.
- For example:

| | |
|---|---|
| :FUNCtion:FFT:FREQuency:TRACk ON | Turn on the frequency track switch. |
| :FUNCtion:FFT:FREQuency:TRACk? | Query returns 1. |

## :FUNCtion:FFT:DETEction:REALTime

- Command format:
  :FUNCtion:FFT:DETEction:REALTime {PPEAK|NPEAK| AVERage| SAMPle}
  :FUNCtion:FFT:DETEction:REALTime?
- Functional description:
  This command is used to set the demodulation mode for real-time spectrum.
  PPEAK: Take the maximum value within the range of each sampling point.
  NPEAK: Take the minimum value within the range of each sampling point.
  AVERage: Take the average value within the range of each sampling point.
  SAMPle: Take the value of first point within the range of each sampling point.
- Return format:
  Query returns the demodulation mode of real-time spectrum.
- For example:
  :FUNCtion:FFT:DETEction:REALTime PPEAK
  Set the demodulation mode of real-time spectrum to +peak detection.
  :FUNCtion:FFT:DETEction:REALTime? Query returns PPEAK.

## :FUNCtion:FFT:DETEction:AVERage

- Command format:
  :FUNCtion:FFT:DETEction:AVERage {OFF|PPEAK|NPEAK| AVERage|SAMPle}
  :FUNCtion:FFT:DETEction:AVERage?
- Functional description:
  This command is used to set the demodulation mode for the average spectrum.
  OFF: turn off the average spectrum
  PPEAK: Take the maximum value within the range of each sampling point.
  NPEAK: Take the minimum value within the range of each sampling point..

AVERage: Take the average value within the range of each sampling point.
SAMPle: Take the value of first point within the range of each sampling point.

- Return format:
  Query returns the demodulation mode of the average spectrum.
- For example:
  :FUNCtion:FFT:DETEction:AVERage PPEAK
  Set the demodulation mode of the average spectrum to +peak detection.
  :FUNCtion:FFT:DETEction:AVERage? Query returns PPEAK.

## :FUNCtion:FFT:DETEction:AVERage:COUNt

- Command format:
  :FUNCtion:FFT:DETEction:AVERage:COUNt <value>
  :FUNCtion:FFT:DETEction:AVERage:COUNt?
- Functional description:
  This command is used to set the average number of times for average spectrum, the range can set to 2~512.
- Return format:
  Query the average number of times of average spectrum.
- For example:
  :FUNCtion:FFT:DETEction:AVERage:COUNt 56
  Set the average number of times of average spectrum to 56.
  :FUNCtion:FFT:DETEction:AVERage:COUNt? Query returns 56.

## :FUNCtion:FFT:DETEction:MAXHold

- Command format:
  :FUNCtion:FFT:DETEction:MAXHold {OFF|PPEAK|NPEAK|AVERage|SAMPle}
  :FUNCtion:FFT:DETEction:MAXHold?
- Functional description:
  This command is used to set the demodulation mode for the maximum hold spectrum.
  OFF: turn off the average spectrum
  PPEAK: Take the maximum value within the range of each sampling point.
  NPEAK: Take the minimum value within the range of each sampling point.
  AVERage: Take the average value within the range of each sampling point.
  SAMPle: Take the value of first point within the range of each sampling point.
- Return format:
  Query returns the demodulation mode of the maximum hold spectrum.
- For example:
  :FUNCtion:FFT:DETEction:MAXHold PPEAK
  Set the demodulation mode of the maximum hold spectrum to +peak detection.
  :FUNCtion:FFT:DETEction:MAXHold?
  Query returns PPEAK.

## :FUNCtion:FFT:DETEction:MINHold

- Command format:
  :FUNCtion:FFT:DETEction:MINHold {OFF|PPEAK|NPEAK| AVERage|SAMPle}
  :FUNCtion:FFT:DETEction:MINHold?
- Functional description:
  This command is used to set the demodulation mode for the minimum hold spectrum.
  OFF: Turn off the average spectrum
  PPEAK: Take the minimum value within the range of each sampling point.
  NPEAK: Take the minimum value within the range of each sampling point.
  AVERage: Take the average value within the range of each sampling point.
  SAMPle: Take the value of first point within the range of each sampling point.
- Return format:

Query returns the demodulation mode of the minimum hold spectrum.

- For example：
  :FUNCtion:FFT:DETEction:MINHold PPEAK
  Set the demodulation mode of spectrum to +peak detection.
  :FUNCtion:FFT:DETEction:MINHold? Query returns PPEAK.

## :FUNCtion:FFT:DETEction:RESet

- Command format：
  :FUNCtion:FFT:DETEction:RESet
- Functional description：
  This command is used to reset the average value, the maiximum and minimum hold spectrum.
- For example：
  :FUNCtion:FFT:DETEction:RESet Reset each spectrum.

## :FUNCtion:FFT:MARK:TYPE

- Command format：
  :FUNCtion:FFT:MARK:TYPE {OFF|AUTO|THReshold| MANUal}
  :FUNCtion:FFT:MARK:TYPE?
- Functional description：
  This command is used to set mark type for the spectrum.
  OFF: Turn off the spectrum marker.
  AUTO: Auto spectrum marker.
  THReshold: Threshold spectrum marker.
  MANUal: Manual spectrum marker.
- Return format：
  Query returns the current selected marker type of spectrum.
- For example：
  :FUNCtion:FFT:MARK:TYPE AUTO Set marker type of the spectrum to AUTO.
  :FUNCtion:FFT:MARK:TYPE? Query returns AUTO.

## :FUNCtion:FFT:MARK:SOURce

- Command format：
  :FUNCtion:FFT:MARK:SOURce {REALtime|AVERage|MAXHold|MINHold}
  :FUNCtion:FFT:MARK:SOURce?
- Functional description：
  This command is used to set mark source of the spectrum marker, this instruction is the common instruction.
  REALtime：Mark the real-time spectrum.
  AVERage：Mark the average spectrum.
  MAXHold：Mark the maximum hold spectrum.
  MINHold：Mark the minimum hold spectrum.
- Return format：
  Query returns the current selected mark source.
- For example：
  :FUNCtion:FFT:MARK:SOURce AVERage
  Set mark source of the spectrum marker to average spectrum.
  :FUNCtion:FFT:MARK:SOURce? Query returns AVERage.

## :FUNCtion:FFT:MARK:POINts

- Command format：
  :FUNCtion:FFT:MARK:POINts <value>

:FUNCtion:FFT:MARK:POINts ?

- **Functional description:**
  Set mark point of of the spectrum marker.
  <value>: the value of mark point, the range can set to to 1~50.
- **Return format:**
  Query returns mark point of the spectrum marker.
- **For example:**

| | |
|---|---|
| :FUNCtion:FFT:MARK:POINts 20 | Set mark point of of the spectrum marker to 20. |
| :FUNCtion:FFT:MARK:POINts ? | Query returns 20. |

## :FUNCtion:FFT:MARK:EVENt

- **Command format:**
  :FUNCtion:FFT:MARK:EVENt {1|ON}|{0|OFF}
  :FUNCtion:FFT:MARK:EVENt?
- **Functional description:**
  This command is used to turn on/off mark list of the spectrum marker.
- **Return format:**
  Query the status of mark list, 1 represents ON, 0 represents OFF.
- **For example:**

| | |
|---|---|
| :FUNCtion:FFT:MARK:EVENt ON | Turn on mark list. |
| :FUNCtion:FFT:MARK:EVENt? | Query returns 1. |

## :FUNCtion:FFT:MARK:DATA?

- **Command format:**
  :FUNCtion:FFT:MARK:DATA?
- **Functional description:**
  This command is used to read the mark event list data in FFT of the oscilloscope.
- **Return format:**
  Query returns the mark event list data in FFT. Returned data is conform with Appendix 2：IEEE 488.2 Binary Data Format.
- **For example:**
  :FUNCtion:FFT:MARK:DATA? Query returns

```
#9000000089FFT,
ID,Freq,Amp,
1,1.000E+003,7.800E-001,
2,2.000E+003,7.900E-001,
3,3.000E+003,7.700E-001,
4,4.000E+003,7.300E-001,
5,5.000E+003,7.400E-001,
```

  FFT represents FFT mode, the event table data in CSV format is followed behind. The specified format of the event table data is automatically adapted by different devices. The data are separated by commas and will automatically line wrap according to the decoding list.

## :FUNCtion:FFT:MARK:THReshold:LEVel

- **Command format:**
  :FUNCtion:FFT:MARK:THReshold:LEVel <value>
  :FUNCtion:FFT:MARK:THReshold:LEVel?
- **Functional description:**
  This command is used to set threshold voltage value for threshold spectrum mark.
  <value>: Threshold voltage value, the unit is V when vertical direction is Vrms; the unit is dB when vertical direction is dBVrms; if the unit is not matched with the vertical unit, the setting is invalid.

- **Return format:**
  Query returns the threshold value of spectrum mark in scientific notation, the unit is related to :FUNCtion:FFT:VTYPe.
- **For example:**
  :FUNCtion:FFT:MARK:THReshold:LEVel -12.5dB
  Set the threshold of spectrum mark to -12.5dB.
  :FUNCtion:FFT:MARK:THReshold:LEVel?
  Query returns -1.250e-001.
  :FUNCtion:FFT:MARK:THReshold:LEVel 0.15V.
  Set the threshold of spectrum mark to 0.15V.
  :FUNCtion:FFT:MARK:THReshold:LEVel?
  Query returns 1.500e-001.

## :FUNCtion:FFT:MARK:MANUal:PEAK

- **Command format:**
  :FUNCtion:FFT:MARK:MANUal:PEAK
- **Functional description:**
  This command is used to move the marker to the maximum peak.
- **For example:**

| | |
|---|---|
| :FUNCtion:FFT:MARK:MANUal:PEAK | Move the marker to the maximum peak. |

## :FUNCtion:FILTer:TYPE

- **Command format:**
  :FUNCtion:FILTer:TYPE {LP|HP|BP|BS}
  :FUNCtion:FILTer:TYPE?
- **Functional description:**
  This command is used to set the filter type. LP, HP, BP, BS respectively represents low-pass filter, high-pass filter, band-pass filter and band-limit filter.
- **Return format:**
  Query returns LP, HP, BP and BS.
- **For example:**

| | |
|---|---|
| :FUNCtion:SOUR1 CHAN1 | Set CH1 as the source. |
| :FUNC:FILT:TYPE BP | Set the filter type to band-pass filter. |
| :FUNC:FILT:TYPE? | Query returns BP. |

## :FUNCtion:FILTer:FREQuency:HIGH

- **Command format:**
  :FUNCtion:FILTer:FREQuency:HIGH < freq>
  :FUNCtion:FILTer:FREQuency:HIGH?
- **Functional description:**
  This command is used to set the upper limit cut-off frequency value of filter. It is suitable for high-pass filter, band-pass filter and band-limit filter.
- **Return format:**
  Query returns 1.000e003, the unit is Hz.
- **For example:**
  :FUNCtion:SOUR1 CHAN1
  Set CH1 as the source.
  :FUNC:FILT:FREQ:HIGH 1 kHz
  Set the upper limit cut-off frequnecy value to 1 kHz.
  :FUNC:FILT:FREQ:HIGH?
  Query returns 1.000e003.

## :FUNCtion:FILTer:FREQuency:LOW

- Command format:
  :FUNCtion:FILTer:FREQuency:LOW <freq>
  :FUNCtion:FILTer:FREQuency:LOW?
- Functional description:
  This command is used to set the low limit cut-off frequency value of filter. It is suitable for high-pass filter, band-pass filter and band-limit filter.
- Return format:
  Query returns 6.000e001, the unit is Hz.
- For example:
  :FUNC:SOUR1 CHAN1
  Set channel 1 as the source.
  :FUNC:FILT:FREQ:LOW 60 Hz
  Set the low limit cut-off frequnecy value to 60 Hz.
  :FUNC:FILT:FREQ:LOW?
  Query returns 6.000e001.

## :FUNCtion:LOGic:INVert

- Command format:
  :FUNCtion:LOGic:INVert {{1|ON}|{0|OFF}}
  :FUNCtion:LOGic:INVert?
- Functional description:
  This command is used to turn on/off logic invert function.
- Return format:
  Query returns 1 or 0, it represents ON or OFF.
- For example:
  :FUNCtion:LOGic:INVert OFF
  Turn off logic invert function.
  :FUNCtion:LOGic:INVert?
  Query returns 0, it represents logic invert function is tured off.

## :FUNCtion:EXPRession

- Command format:
  :FUNCtion:EXPRession <expression>
- Functional description:
  Use free combination expression to peform mathematical calculation.
  Expression format refer to Advance option in MATH menu of the oscilloscope, <expression> is belong to ASCII character string parameter.
- For example:
  :FUNCtion:EXPRession "CH1*CH2"
  It represents multiply channel 1 with channel 2.

## MEASure Command

The command is for the basic measurement operation; all parameter measurement can get the test value without open measurement function; by default, it will turn on measurement and acquire the test value autommaticaally. In general, test result is urned in scientific notation.

### :MEASure:ALL

- **Command format:**
  :MEASure:ALL {{1 | ON} | {0 | OFF}}
  :MEASure:ALL?
- **Functional description:**
  This command is used to turn on/off all measurement functions.
- **Return format:**
  Query returns whether all measurement functions is enabled.
- **For example:**

| | |
|---|---|
| :MEASure:ALL ON | Turn on all measurement functions. |
| :MEASure:ALL? | Query returns 1. |

### :MEASure:CLEar

- **Command format:**
  :MEASure:CLEar
- **Functional description:**
  This command is used to clear all the current measured parameter values, and refresh the measured data.
- **For example:**

| | |
|---|---|
| :MEAS:CLE | Clear all the current measured parameter values. |

### :MEASure:SOURce

- **Command format:**
  :MEASure:SOURce <source>
  :MEASure:SOURce?
- **Functional description:**
  This command is used to select measuring source. <source> is CHANnel<n>, n takes value from 1, 2, 3, 4
- **Return format:**
  Query returns {CHANnel1 | CHANnel2 | CHANnel3 | CHANnel4 | MATH }.
- **For example:**

| | |
|---|---|
| :MEAS:SOUR CHAN1 | Select CH1 as measuring source. |
| :MEAS:SOUR? | Return CHANnel1. |

### :MEASure:SLAVe:SOURce

- **Command format:**
  :MEASure:SLAVe:SOURce <source>
  :MEASure:SLAVe:SOURce?
- **Functional description:**
  This command is used to select slave source. <source> is CHANnel<n>, n takes value from 1, 2, 3, 4.
- **Return format:**
  Query returns {CHANnel1 | CHANnel2 | CHANnel3 | CHANnel4 | MATH }.
- **For example:**

| | |
|---|---|
| :MEAS:SLAV:SOUR CHAN1 | Select CH1 as measuring source. |
| :MEAS:SLAV:SOUR? | Return CHANnel1. |

## :MEASure:STATistic:DISPlay

- **Command format:**
  :MEASure:STATistic:DISPlay { {1|ON} | {0|OFF} }
  :MEASure:STATistic:DISPlay?
- **Functional description:**
  This command is used to turn on/off measure statistic function.
- **Return format:**
  Query returns 1 or 0, it represents ON or OFF.
- **For example:**

| | |
|---|---|
| :MEASure:STATistic:DISPlay ON | Turn on measure statistic function. |
| :MEASure:STATistic:DISPlay? | Query returns 1. |

## :MEASure:STATistic:RESet

- **Command format:**
  :MEASure:STATistic:RESet
- **Functional description:**
  This command is used to clear historical statistic data and restart to count.
- **For example:**

| | |
|---|---|
| :MEASure:STATistic:RESet | Clear historical statistic data and restart to count. |

## :MEASure:STATistic:ITEM

- **Command format:**
  :MEASure:STATistic:ITEM <item>
  :MEASure:STATistic:ITEM? <type>,<item>
- **Functional description:**
  This command is used to open statistic function for the sepecified arbitrary waveform parameter and query the statistic result of arbitrary waveform parameter.
  <item> represents waveform parameter,
  {VMAX|VMIN|VPP|VTOP|VBASe|VAMPlitude|VAVerage|CYCVAVerage|VMIDdle|VRMS|ACRMS|CYCRMS|AREa|CYCAREa|FREQuency|PHASe|PERiod|OVERshoot|PREShoot|RISetime|FALLtime|PWIDth|NWIDth|PDUTy|NDUTy|PDELay|NDELay|PULSes|FRR|FRF|FFR|FFF|LRR|LRF|LFR|LFF}.
  <type> represents statistic type,
  {MAXimum|MINimum|CURRent|AVERages|DEViation}, it represents maximum, minimum, current value, averaged value and deviation respectively.
- **Return format:**
  Query returns the statistic results in scientific results.
- **For example:**
  :MEASure:STATistic:ITEM  CYCVAV
  Open cycle average measure statistic function.
  :MEASure:STATistic:ITEM?  MAX,CYCVAV
  Return the maximum of cycle average statistic 1.120e000.

## :MEASure:WINDow

- **Command format:**
  :MEASure:WINDow {SCReen | CURSor}
  :MEASure:WINDow?
- **Functional description:**
  This command is used to set the window area of measurement. SCReen is full screen. CURSor is cursor area.
  Use command :CURSor:CURAX and :CURSor:CURBX to adjust the vertical range of cursor line when the measurement area is cursor area.

- **Return format:**
  Query returns {SCReen | CURSor}.
- **For example:**

| | |
|---|---|
| :MEASure:WINDow CURSor | Set the window area of measurement to cursor area. |
| :MEASure:WINDow? | Query returns CURSor |

## :MEASure:THReshold:DEFault

- **Command format:**
  :MEASure:THReshold:DEFault <source>
- **Functional description:**
  This command is used to set default value of threshold. The lower limit of threshold is 10%, the middle limit of threshold is 50%, and the upper limit of threshold is 90%.
  <source> represents {CHANnel1| CHANnel2| CHANnel3| CHANnel4 | MATH}.
- **For example:**

| | |
|---|---|
| :MEASure:THReshold:DEFault CHANnel1 | Set threshold of CH1 to default value. |

## :MEASure:THReshold

- **Command format:**
  :MEASure:THReshold <source>, [<lower>], [< middle>], [<upper>]
  :MEASure:THReshold? <source>
- **Functional description:**
  This command is used to set the upper, middle, lower limit of threshold for <source> in user-defined measurement mode. The upper, middle, lower limit of threshold can set independently. Threshold value will affect all parameters measurements of time, delay, phase and duty cycle.
  <source> represents {CHANnel1| CHANnel2| CHANnel3| CHANnel4 | MATH}.
  <lower> take value from 5% - 93%; <middle> take value from 6% - 94%; <upper> take value from 7% - 95%.
- **Return format:**
  Query returns the upper, middle, lower limit of threshold in scientific notation. The unit is %.
- **For example:**
  :MEASure:THReshold CHANnel1,20,40,80
  Set the upper, middle, lower limit of threshold for CH1.
  :MEASure:THReshold? CHANnel1
  Query returns 2.000e001, 4.000e001, 8.000e001.
  :MEASure:THReshold CHANnel1,40
  Set the middle limit of threshold for CH1 to 40%.

## :MEASure:PDUTy?

- **Command format:**
  :MEASure:PDUTy? [<source>]
- **Functional description:**
  This command is used to measure positive duty cycle of the specified channel waveform. <source> takes value from CHANnel1, CHANnel2, CHANnel3, CHANnel4 or MATH. Omitting represents the current channel.
- **Return format:**
  Query returns 5.000e001, the unit is %.

## :MEASure:NDUTy?

- **Command format:**
  :MEASure:NDUTy? [<source>]
- **Functional description:**
  This command is used to measure negative duty cycle of the specified channel waveform. <source> takes value from CHANnel1, CHANnel2, CHANnel3, CHANnel4 or MATH. Omitting represents the current channel.Omitting represents the current channel.

- **Return format:**
  Query returns 5.000e001, the unit is %.

## :MEASure:PDELay?

- **Command format:**
  :MEASure:PDELay? [<source1>, <source2>]
- **Functional description:**
  This command is used to measure time delay of <source1> and <source2> with respect to rising edge. <source> takes value from CHANnel1, CHANnel2, CHANnel3, CHANnel4 or MATH.
- **Return format:**
  Query returns -1.000e-004, the unit is s.
- **For example:**
  :MEASure:PDEL? CHAN1,CHAN2
  Measuring time delay with repect to rising edge.

## :MEASure:NDELay?

- **Command format:**
  :MEASure:NDELay? [<source1>, <source2>]
- **Functional description:**
  This command is used to measure time delay of <source1> and <source2> with respect to falling edge. <source> takes value from CHANnel1, CHANnel2, CHANnel3, CHANnel4 or MATH.
- **Return format:**
  Query returns -1.000e-004, the unit is s.
- **For example:**
  Measuring time delay with repect to rising edge.
  :MEASure:NDEL? CHAN1,CHAN2

## :MEASure:PHASe?

- **Command format:**
  :MEASure:PHASe? [<source1>, <source2>]
- **Functional description:**
  This command is used to timing measure the amount of time of <source1> which exceeds or lags with respect to <source2>, expressed in degrees, with 360° as a period. <source> takes value from CHANnel1, CHANnel2, CHANnel3, CHANnel4 or MATH.
- **Return format:**
  Query returns 1.000e001, the unit is degree.
- **For example:**
  Measuring the amount of time of <source1> which exceeds or lags with respect to <source2>.
  :MEASure:PHAS? CHAN1,CHAN2

## :MEASure:VPP?

- **Command format:**
  :MEASure:VPP? [<source>]
- **Functional description:**
  This command is used to measure peak-to-peak value of the specified channel waveform. <source> takes value from CHANnel1, CHANnel2, CHANnel3, CHANnel4 or MATH. Omitting represents the current channel.Omitting represents the current channel.
- **Return format:**
  Query returns 3.120e000, the unit is V.

## :MEASure:VMAX?

- **Command format:**
  :MEASure:VMAX? [<source>]
- **Functional description:**
  This command is used to measure the maximum value of the specified channel waveform. <source> takes value from CHANnel1, CHANnel2, CHANnel3, CHANnel4 or MATH. Omitting represents the current channel.Omitting represents the current channel.
- **Return format:**
  Query returns 2.120e000, the unit is V.

## :MEASure:VMIN?

- **Command format:**
  :MEASure:VMIN? [<source>]
- **Functional description:**
  This command is used to measure the minimum value of the specified channel waveform. <source> takes value from CHANnel1, CHANnel2, CHANnel3, CHANnel4 or MATH. Omitting represents the current channel.
- **Return format:**
  Query returns -2.120e000, unit is V.

## :MEASure:VAMPlitude?

- **Command format:**
  :MEASure:VAMPlitude? [<source>]
- **Functional description:**
  This command is used to measure amplitude value of the specified channel waveform. <source> takes value from CHANnel1, CHANnel2, CHANnel3, CHANnel4 or MATH. Omitting represents the current channel.
- **Return format:**
  Query returns 3.120e000, the unit is V.

## :MEASure:VTOP?

- **Command format:**
  :MEASure:VTOP? [<source>]
- **Functional description:**
  This command is used to measure the top value of the specified channel waveform.<source> takes value from CHANnel1, CHANnel2, CHANnel3, CHANnel4 or MATHOmitting represents the current channel.
- **Return format:**
  Query returns 3.120e000, the unit is V.

## :MEASure:VBASe?

- **Command format:**
  :MEASure:VBASe? [<source>]
- **Functional description:**
  This command is used to measure the bottom value of the specified channel waveform. <source> takes value from CHANnel1, CHANnel2, CHANnel3, CHANnel4 or MATHOmitting represents the current channel.
- **Return format:**
  Query returns -3.120e000, the unit is V.

## :MEASure:VMIDdle?

- **Command format:**
  :MEASure:VMIDdle? [<source>]

- Functional description:
  This command is used to measure the middle value of the specified channel waveform. <source> takes value from CHANnel1, CHANnel2, CHANnel3, CHANnel4 or MATH. Omitting represents the current channel.
- Return format:
  Query returns 0.120e000, the unit is V.

:MEASure:VAVerage?

- Command format:
  :MEASure:VAVerage? [<interval>], [<source>]
- Functional description:
  This command is used to measure the average value of the specified channel waveform. <source> is the specified channel and takes value from CHANnel1, CHANnel2, CHANnel3, CHANnel4 or MATH. If there is no assigned <source>, then the current channel as the default; <interval> specifies the measuring interval, and take value as CYCLe, DISPlay. CYCLe represents integer cycle. DISPlay represents full screen. If there is no assigned <interval>, then DISPlay as the default.
- Return format:
  Query returns 1.120e000, the unit is V.

:MEASure:VRMS?

- Command format:
  :MEASure:VRMS? [<interval>], [<source>]
- Functional description:
  This command is used to measure the root mean sqaure value of the specified channel waveform. <source> is the specified channel and takes value from CHANnel1, CHANnel2, CHANnel3, CHANnel4 or MATH. If there is no assigned <source>, then the current channel as the default; <interval> specifies the measuring interval, and take value as CYCLe, DISPlay. CYCLe represents integer cycle. DISPlay represents full screen. If there is no assigned <interval>, then DISPlay as the default.
- Return format:
  Query returns 1.120e000, the unit is V.

:MEASure:ACRMs?

- Command format:
  :MEASure:ACRMs? [<source>]
- Functional description:
  This command is used to measure AC RMS of the specified channel waveform. This command is used to measure the area of the specified channel waveform. <source> is the specified channel and takes value from CHANnel1, CHANnel2, CHANnel3, CHANnel4 or MATH. If there is no assigned <source>, then the current channel as the default.
- Return format:
  Query returns 1.230e000, the unit is V.

:MEASure:AREa?

- Command format:
  :MEASure:AREa? [<interval>], [<source>]
- Functional description:
  This command is used to measure the area of the specified channel waveform. <source> is the specified channel and takes value from CHANnel1, CHANnel2, CHANnel3, CHANnel4 or MATH. If there is no assigned <source>, then the current channel as the default; <interval> specifies the measuring interval, and take value as CYCLe, DISPlay. CYCLe represents integer cycle. DISPlay represents full screen. If there is no assigned <interval>, then DISPlay as the default.
- Return format:
  Query returns 3.456e002, the unit is Vs.

## :MEASure:OVERshoot?

- **Command format:**
  :MEASure:OVERshoot? [<source>]
- **Functional description:**
  This command is used to measure overshoot of the specified channel waveform. <source> takes value from CHANnel1, CHANnel2, CHANnel3, CHANnel4 or MATH. Omitting represents the current channel.
- **Return format:**
  Query returns 1.230e002, unit is %.

## :MEASure:PREShoot?

- **Command format:**
  :MEASure:PREShoot? [<source>]
- **Functional description:**
  This command is used to measure preshoot of the specified channel waveform. <source> takes value from CHANnel1, CHANnel2, CHANnel3, CHANnel4 or MATH. Omitting represents the current channel.
- **Return format:**
  Query returns 1.230e002, unit is %.

## :MEASure:FREQuency?

- **Command format:**
  :MEASure:FREQuency? [<source>]
- **Functional description:**
  This command is used to measure frequency of the specified channel waveform. <source> takes value from CHANnel1, CHANnel2, CHANnel3, CHANnel4 or MATH. Omitting represents the current channel.
- **Return format:**
  Query returns 2.000e003, the unit is Hz.

## :MEASure:RISetime?

- **Command format:**
  :MEASure:RISetime? [<source>]
- **Functional description:**
  This command is used to measure rising time of the specified channel waveform. <source> takes value from CHANnel1, CHANnel2, CHANnel3, CHANnel4 or MATH. Omitting represents the current channel.
- **Return format:**
  Query returns 5.000e-005, the unit is s.

## :MEASure:FALLtime?

- **Command format:**
  :MEASure:FALLtime? [<source>]
- **Functional description:**
  This command is used to measure falling time of the specified channel waveform. <source> takes value from CHANnel1, CHANnel2, CHANnel3, CHANnel4 or MATH. Omitting represents the current channel.
- **Return format:**
  Query returns 5.000e-005, the unit is s.

## :MEASure:PERiod?

- **Command format:**
  :MEASure:PERiod? [<source>]
- **Functional description:**

This command is used to measure period of the specified channel waveform. <source> takes value from CHANnel1, CHANnel2, CHANnel3, CHANnel4 or MATH. Omitting represents the current channel.

- Return format:

  Query returns 5.000e-003, the unit is s.

:MEASure:PWIDth?

- Command format:

  :MEASure:PWIDth? [<source>]
- Functional description:

  This command is used to measure positive pulse width of the specified channel waveform. <source> takes value from CHANnel1, CHANnel2, CHANnel3, CHANnel4 or MATH. Omitting represents the current channel.
- Return format:

  Query returns 5.000e-003, the unit is s.

:MEASure:NWIDth?

- Command format:

  :MEASure:NWIDth? [<source>]
- Functional description:

  This command is used to measure negative pulse width of the specified channel waveform. <source> takes value from CHANnel1, CHANnel2, CHANnel3, CHANnel4 or MATH. Omitting represents the current channel.
- Return format:

  Query returns 5.000e-003, the unit is s.

:MEASure:PULSes?

- Command format:

  :MEASure:PULSes? [<source>]
- Functional description:

  This command is used to measure the number of positive pulse of the specified channel. <source> takes value from CHANnel1, CHANnel2, CHANnel3, CHANnel4 or MATH. Omitting represents the current channel.
- Return format:

  Query returns 2.000e+000, it represents 2 pulses.

:MEASure:FRR?

- Command format:

  :MEASure:FRR? <source1>, <source2>
- Functional description:

  This command is used to measure the time between <source1> and the first rising edge of <source2>. <source> takes value from CHANnel1, CHANnel2, CHANnel3, CHANnel4 or MATH.
- Return format:

  Query returns 5.000e-003, the unit is s.

:MEASure:FRF?

- Command format:

  :MEASure:FRF? <source1>, <source2>
- Functional description:

  This command is used to measure the time between the first rising edge of <source1> and the first falling edge of <source2>. <source> takes value from CHANnel1, CHANnel2, CHANnel3, CHANnel4 or MATH. Omitting represents the current channel.
- Return format:

  Query returns 5.000e-003, the unit is s.

## :MEASure:FFR?

- **Command format:**
  :MEASure:FFR? <source1>, <source2>
- **Functional description:**
  This command is used to measure the time between the first falling edge of <source1> and the first rising edge of <source2>.<source> takes value from CHANnel1, CHANnel2, CHANnel3, CHANnel4 or MATH.
- **Return format:**
  Query returns 5.000e-003, the unit is s.

## :MEASure:FFF?

- **Command format:**
  :MEASure:FFF? <source1>, <source2>
- **Functional description:**
  This command is used to measure the time between <source1> and the first falling edge of <source2>. <source> takes value from CHANnel1, CHANnel2, CHANnel3, CHANnel4 or MATH.
- **Return format:**
  Query returns 5.000e-003, the unit is s.

## :MEASure:LRR?

- **Command format:**
  :MEASure:LRR? <source1>, <source2>
- **Functional description:**
  This command is used to measure the time between the first rising edge of <source1> and the last rising edge of <source2>. <source> takes value from CHANnel1, CHANnel2, CHANnel3, CHANnel4 or MATH.
- **Return format:**
  Query returns 5.000e-003, the unit is s.

## :MEASure:LRF?

- **Command format:**
  :MEASure:LRF? <source1>, <source2>
- **Functional description:**
  This command is used to measure the time between the first rising edge of <source1> and the last falling edge of <source2>. <source> takes value from CHANnel1, CHANnel2, CHANnel3, CHANnel4 or MATH.
- **Return format:**
  Query returns 5.000e-003, the unit is s.

## :MEASure:LFR?

- **Command format:**
  :MEASure:LFR? <source1>, <source2>
- **Functional description:**
  This command is used to measure the time between the first falling edge of <source1> and the last rising edge of <source2>. <source> takes value from CHANnel1, CHANnel2, CHANnel3, CHANnel4 or MATH.
- **Return format:**
  Query returns 5.000e-003, the unit is s.

## :MEASure:LFF?

- **Command format:**
  :MEASure:LFF? <source1>, <source2>
- **Functional description:**

This command is used to measure the time between the first falling edge of <source1> and the last falling edge of <source2>. <source> takes value from CHANnel1, CHANnel2, CHANnel3, CHANnel4 or MATH.

➢ Return format:
Query returns 5.000e-003, the unit is s.

## TRIGger Command

This command is used to control trigger sweep mode and trigger specifiedation. Trigger determines when the oscilloscope to start sampling data and display waveform.

### Trigger Control

#### :TRIGger:MODE

➢ Command format:
:TRIGger:MODE <mode>
:TRIGger:MODE?

➢ Functional description:
This command is used to set trigger mode and it can be automatically suitable for trigger mode of different model.
<mode> contains EDGE (edge trigger), PULSe (pulse width trigger), VIDeo (video trigger), SLOPe (slope trigger), RUNT (runt trigger), WINDow (over-amplitude trigger), DELay (delay trigger), TIMeout (timeout trigger), DURation (duration time trigger), SHOLd (setup hold trigger), NE (Nth edge trigger) and PATTern (pattern trigger).

➢ Return format:
Query returns the trigger mode.

➢ For example:

| | |
|---|---|
| :TRIGger:MODE NE | Set trigger mode to Nth edge trigger. |
| TRIGger:MODE? | Query returns NE. |

#### :TRIGger:FORCe

➢ Command format:
:TRIGger:FORCe

➢ Functional description:
This command is sutiable for the oscilloscope if there is no properly trigger condition. Executing this command to force to produce a trigger signal to generate input waveform and display it.

➢ For example:

| | |
|---|---|
| TRIG:FORC | Force trigger |

#### :TRIGger:SWEep

➢ Command format:
:TRIGger:SWEep {AUTO|NORMal|SINGle}
:TRIGger:SWEep?

➢ Functional description:
This command is used to select trigger sweep mode.
AUTO: In no trigger condition, the internal will generate a trigger signal to force trigger.
NORMal: It will be generated only when the trigger condition is met.
SINGle: Generating single trigger when the trigger condition is met and then stop.

➢ Return format:
Query returns trigger sweep mode {AUTO|NORMal|SINGle}.

- For example:

| | |
|---|---|
| :TRIGger:SWEep AUTO | Set channel 1 as AUTO trigger mode. |
| :TRIGger:SWEep? | Query returns AUTO. |

## :TRIGger:LEVel:ASETup

- Command format:
  :TRIGger:LEVel:ASETup
- Functional description:
  This command is used to set tigger level at the vertical midpoint of signal amplitude.
- For example:

| | |
|---|---|
| :TRIG:LEVel:ASETup | Set tigger level at the center. |

## :TRIGger:STATus?

- Command format:
  :TRIGger:STATus?
- Functional description:
  Query the current trigger status of the oscilloscope.
- Return format:
  Query returns STOP/ARMED/READY/TRIGED/AUTO/SCAN/RESET/REPLAY/WAIT.
- For example:

| | |
|---|---|
| :TRIGger:STATus? | Query returns AUTO. |

## :TRIGger:LEVel

- Command format:
  :TRIGger:LEVel <level>
  :TRIGger:LEVel?
- Functional description:
  This command is used to set trigger level value of normal trigger mode. Nuerical value of <level> must be set after the conversion according to the amplitude volts/div scale and screen information.
- Return format:
  Query returns the <level> setting value, the unit is V.
- For example:

| | |
|---|---|
| :TRIG:LEV 2 | Set trigger level of the trigger to 2 V. |
| :TRIG:LEV? | Query returns 2.000e000. |

## :TRIGger:LEVel:LOW

- Command format:
  :TRIGger:LEVel:LOW <level>
  :TRIGger:LEVel:LOW?
- Functional description:
  This command is used to set low level value of slop trigger. Nuerical value of <level> must be set after the conversion according to the amplitude volts/div scale and screen information.
- Return format:
  Query returns the <level> setting value, the unit is V.
- For example:

| | |
|---|---|
| :TRIG:LEV:LOW 2 | Set trigger level of the trigger to 2 V. |
| :TRIG:LEV:LOW? | Query returns 2.000e000. |

## :TRIGger:LEVel:HIGH

- **Command format:**
  :TRIGger:LEVel:HIGH  <level>
  :TRIGger:LEVel:HIGH?
- **Functional description:**
  This command is used to set high level value of slop trigger. Nuerical value of <level> must be set after the conversion according to the amplitude volts/div scale and screen information.
- **Return format:**
  Query returns the <level> setting value, the unit is V.
- **For example:**

| | |
|---|---|
| :TRIG:LEV:HIGH 2 | Set trigger level of the trigger to 2 V. |
| :TRIG:LEV:HIGH? | Query returns 2.000e000. |

## :TRIGger:SOURce

- **Command format:**
  :TRIGger:SOURce  <source>
  :TRIGger:SOURce?
- **Functional description:**
  This command is used to set signle trigger source, input channel (CHANnel1, CHANnel 2, CHANnel, CHANnel), external trigger (EXT), AC Line (main electricity). EDGE/ PULSe only support AC Line and EXT source.
  <source> represents trigger source, and takes value from CHANnel<n>|EXT |ACLINE, <n> takes value from 1, 2, 3 or 4.
- **Return format:**
  Query returns trigger source {CHANnel1|CHANnel2|CHANnel3|CHANnel4|EXT|ACLINE}.
- **For example:**

| | |
|---|---|
| :TRIGger:SOUR CHAN1 | Set CH1 as edge trigger. |
| :TRIGger:SOUR? | Query returns CHANnel1. |

## :TRIGger:COUPling

- **Command format:**
  :TRIGger:COUPling {DC|AC|LF|HF|NOISE}
  :TRIGger:COUPling?
- **Functional description:**
  This command is used to set coupling mode, DC (direct current), AC (alternating current), LF (low frequency reject), HF (high frequency reject), NOISE (noise reject). Only VIDeo does not support.
- **Return format:**
  Query returns coupling mode {DC|AC|LF|HF|NOISE}.
- **For example:**

| | |
|---|---|
| :TRIGger:COUPling AC | Set edge trigger as AC. |
| :TRIGger:COUPling? | Query returns AC. |

## Edge Trigger

### :TRIGger:EDGE:SLOPe

- **Command format:**
  :TRIGger:EDGE:SLOPe {POSitive|NEGative|ALTernation}
  :TRIGger:EDGE:SLOPe?
- **Functional description:**
  This command is used to set edge trigger type, which is POSitive (rising edge), NEGative (falling edge) and ALTernation (rising/falling edge).
- **Return format:**
  Query returns edge type of trigger source {POSitive | NEGative | ALTernation}.
- **For example:**

| | |
|---|---|
| :TRIGger:EDGE:SLOP POS | Set trigger edge to rising edge. |
| :TRIGger:EDGE:SLOP? | Query returns POSitive. |

## Pulse Width Trigger

### :TRIGger:PULSe:QUALifier

- **Command format:**
  :TRIGger:PULSe:QUALifier {GREaterthan | LESSthan | INRange}
  :TRIGger:PULSe:QUALifier?
- **Functional description:**
  This command is used to set the setting condition of pulse time, which is GREaterthan (greater than), LESSthan (less than), EQUal (equal to) and INRange (between).
- **Return format:**
  Query returns {GREaterthan | LESSthan | EQUal| INRange}.
- **For example:**

| | |
|---|---|
| :TRIGger:PULSe:QUALifier GRE | Set pulse condition to GREaterthan. |
| :TRIGger:PULSe:QUALifier? | Query returns GREaterthan. |

### :TRIGger:PULSe:POLarity

- **Command format:**
  :TRIGger:PULSe:POLarity {POSitive | NEGative}
  :TRIGger:PULSe:POLarity?
- **Functional description:**
  This command is used to set pulse polarity, which is POSitive and NEGative.
- **Return format:**
  Query returns { POSitive | NEGative }.
- **For example:**

| | |
|---|---|
| :TRIGger:PULSe:POL POS | Set pulse polarity to POSitive. |
| :TRIGger:PULSe:POL? | Query returns POSitive. |

### :TRIGger:PULSe:TIME:UPPer

- **Command format:**
  :TRIGger:PULSe:TIME:UPPer <time>
  :TRIGger:PULSe:TIME:UPPer?

- **Functional description:**
  This command is used to set the upper limit of pulse trigger time.
- **Return format:**
  Query returns the current time interval, the unit is s.
- **For example:**

| | |
|---|---|
| :TRIGger:PULSe:TIME:UPPer 1 | Set the upper limit of pulse trigger time to 1s. |
| :TRIGger:PULSe:TIME:UPPer? | Query returns 1.000e000. |

### :TRIGger:PULSe:TIME:LOWer

- **Command format:**
  :TRIGger:PULSe:TIME:LOWer <time>
  :TRIGger:PULSe:TIME:LOWer?
- **Functional description:**
  This command is used to set the lower limit of pulse trigger time.
- **Return format:**
  Query returns the current time interval, the unit is s.
- **For example:**

| | |
|---|---|
| :TRIGger:PULSe:TIME:LOWer 1 | Set the lower limit of pulse trigger time to 1s. |
| :TRIGger:PULSe:TIME:LOWer? | Query returns 1.000e0 |

## Video Trigger

### :TRIGger:VIDeo:MODE

- **Command format:**
  :TRIGger:VIDeo:MODE { ODD | EVEN | LINE| ALINes}
  :TRIGger:VIDeo:MODE?
- **Functional description:**
  This command is used to set synchronization mode for video trigger, which includes ODD, EVEN, LINE (specified line), ALINes (all lines).
- **Return format:**
  Query returns { ODD | EVEN | LINE| ALIN}.
- **For example:**

| | |
|---|---|
| :TRIGger:VIDeo:MODE ODD | Set synchronization mode of video trigger to ODD. |
| :TRIGger:VIDeo:MODE? | Query returns ODD. |

### :TRIGger:VIDeo:STANdard

- **Command format:**
  :TRIGger:VIDeo:STANdard { NTSC | PAL | SECAM }
  :TRIGger:VIDeo:STANdard?
- **Functional description:**
  This command is used to set video standard.
- **Return format:**
  Query returns { NTSC | PAL | SECAM }.
- **For example:**

| | |
|---|---|
| :TRIGger:VIDeo:STANdard NTSC | Set video standard to NTSC. |
| :TRIGger:VIDeo:STANdard? | Query returns NTSC. |

### :TRIGger:VIDEO:LINE

- **Command format:**
  :TRIGger:VIDEO:LINE <value>
  :TRIGger:VIDEO:LINE?
- **Functional description:**
  This command is used to set the assigned line. <value> represents the assigned line, the range is relative to video standard.
- **Return format:**
  Query returns the current assigned line.
- **For example:**

| | |
|---|---|
| :TRIG:VIDEO:LINE 50 | Set the assigned line of video synchronization to 50. |
| :TRIG:VIDEO:LINE? | Query returns 50. |

## Slope Trigger

### :TRIGger:SLOPe:QUALifier

- **Command format:**
  :TRIGger:SLOPe:QUALifier {GREaterthan | LESSthan | INRange}
  :TRIGger:SLOPe:QUALifier?
- **Functional description:**
  This command is used to set the setting condition for slop time, which includes GREaterthan (greater than), LESSthan (less than) and INRange (between).
- **Return format:**
  Query returns {GREaterthan | LESSthan | INRange}.
- **For example:**

| | |
|---|---|
| :TRIGger:SLOPe:QUALifier GRE | Set slop condition to GREaterthan. |
| :TRIGger:SLOPe:QUALifier? | Query returns GREaterthan. |

### :TRIGger:SLOPe:SLOPe

- **Command format:**
  :TRIGger:SLOPe:SLOPe {POSitive|NEGative}
  :TRIGger:SLOPe:SLOPe?
- **Functional description:**
  This command is used to set slop trigger type, which includes POSitive (rising) and NEGative (falling).
- **Return format:**
  Query returns {POSitive|NEGative}.
- **For example:**

| | |
|---|---|
| :TRIGger:SLOPe:SLOPe POS | Set slop trigger as POSitive. |
| :TRIGger:SLOPe:SLOPe? | Query returns POSitive. |

### :TRIGger:SLOPe:TIME:UPPer

- **Command format:**
  :TRIGger:SLOPe:TIME:UPPer <time>
  :TRIGger:SLOPe:TIME:UPPer?
- **Functional description:**
  This command is used to set the upper limit of slope trigger time.

- **Return format:**
  Query returns the current time interval, the unit is s.
- **For example:**

| | |
|---|---|
| :TRIGger:SLOPe:TIME:UPPer 1 | Set the upper limit of slope trigger time to 1s. |
| :TRIGger:SLOPe:TIME:UPPer? | Query returns 1.000e000. |

## :TRIGger:SLOPe:TIME:LOWer

- **Command format:**
  :TRIGger:SLOPe:TIME:LOWer <time>
  :TRIGger:SLOPe:TIME:LOWer?
- **Functional description:**
  This command is used to set the lower limit of slope trigger time.
- **Return format:**
  Query returns the current time interval, the unit is s.
- **For example:**

| | |
|---|---|
| :TRIGger:SLOPe:TIME:LOWer 1 | Set the lower limit of slope trigger time to 1s. |
| :TRIGger:SLOPe:TIME:LOWer? | Query returns 1.000e000. |

## :TRIGger:SLOPe:THReshold

- **Command format:**
  :TRIGger:SLOPe:THReshold {LOW|HIGH|LH}
  :TRIGger:SLOPe:THReshold?
- **Functional description:**
  This command is used to set threshold mode of slop trigger.
- **Return format:**
  Query returns {LOW|HIGH|LH}.
- **For example:**

| | |
|---|---|
| :TRIGger:SLOPe:THR HIGH | Set slope trigger threshold to HIGH. |
| :TRIGger:SLOPe:THR? | Query returns HIGH. |

## :TRIGger:SLOPe:RATe:LOWer?

- **Command format:**
  :TRIGger:SLOPe:RATe:LOWer?
- **Functional description:**
  Set the lower limit of the current slop trigger, the unit is V/s.
- **Return format:**
  Query returns the lower limit of slope trigger in scientific notation.
- **For example:**

| | |
|---|---|
| :TRIGger:SLOPe:RATe:LOWer? | Query returns 3.210E+000. |

## :TRIGger:SLOPe:RATe:UPPer?

- **Command format:**
  :TRIGger:SLOPe:RATe:UPPer?
- **Functional description:**
  Query the upper limit of the current slop trigger, the unit is V/s.
- **Return format:**
  Query returns the upper limit of slope trigger in scientific notation.
- **For example:**

| | |
|---|---|
| :TRIGger:SLOPe:RATe:UPPer? | Query returns 3.210E+000. |

## Runt Trigger

### :TRIGger:RUNT:QUALifier

- **Command format:**
  :TRIGger:RUNT:QUALifier {GREaterthan | LESSthan | INRange | NONE}
  :TRIGger:RUNT:QUALifier?
- **Functional description:**
  This command is used to set the setting condition of runt level time, which includes GREaterthan (greater than), LESSthan (less than), INRange (within the range) and NONE (random).
- **Return format:**
  Query returns {GREaterthan | LESSthan | EQUal | NONE}.
- **For example:**

| | |
|---|---|
| :TRIGger:RUNT:QUALifier GRE | Set trigger condition to GREaterthan. |
| :TRIGger:RUNT:QUALifier? | Query returns GREaterthan. |

### :TRIGger:RUNT:POLarity

- **Command format:**
  :TRIGger:RUNT:POLarity {POSitive | NEGative}
  :TRIGger:RUNT:POLarity?
- **Functional description:**
  This command is used to set the pulse polarity of runt level, which includes POSitive (positive pulse width) and NEGative (negative pulse width).
- **Return format:**
  Query returns {POSitive | NEGative}.
- **For example:**

| | |
|---|---|
| :TRIGger:RUNT:POL POS | Set the pulse polarity to POSitive. |
| :TRIGger:RUNT:POL? | Query returns POSitive. |

### :TRIGger:RUNT:LEVel

- **Command format:**
  :TRIGger:RUNT:LEVel {LOW | HIGH}
  :TRIGger:RUNT:LEVel?
- **Functional description:**
  This command is used to set trigger level mode for runt trigger.
- **Return format:**
  Query returns {LOW | HIGH}.

- **For example:**

| | |
|---|---|
| :TRIGger:RUNT:LEV HIGH | Set runt level to HIGH. |
| :TRIGger:RUNT:LEV? | Query returns HIGH. |

### :TRIGger:RUNT:TIME:UPPer

- **Command format:**
  :TRIGger:RUNT:TIME:UPPer <time>
  :TRIGger:RUNT:TIME:UPPer?
- **Functional description:**
  This command is used to set the upper limit of runt level trigger time.
- **Return format:**

Query returns the current upper limit of time, the the unit is s.

- For example：

  :TRIGger:RUNT:TIME:UPPer 1

  Set the upper limit of runt level trigger time to 1s.

  :TRIGger:RUNT:TIME:UPPer?

  Query returns 1.000e000.

### :TRIGger:RUNT:TIME:LOWer

- Command format：

  :TRIGger:RUNT:TIME:LOWer <time>

  :TRIGger:RUNT:TIME:LOWer?
- Functional description：

  This command is used to set the lower limit of runt level trigger time.
- Return format：

  Query returns the current lower limit of time, the unit is s.
- For example：

| | |
|---|---|
| :TRIGger:RUNT:TIME:LOWer 1 | Set the lower limit of runt level trigger time to 1s. |
| :TRIGger:RUNT:TIME:LOWer? | Query returns 1.000e000. |

## Over-amplitude Trigger

### :TRIGger:WINDow:SLOPe

- Command format：

  :TRIGger:WINDow:SLOPe {POSitive|NEGative|ALTernation}

  :TRIGger:WINDow:SLOPe?
- Functional description：

  This command is used to set edge type of trigger source, which is POSitive (rising edge), NEGative (falling edge) and ALTernation (rising/falling edge).
- Return format：

  Query returns edge type of trigger souece {POSitive|NEGative|ALTernation}.
- For example：

| | |
|---|---|
| :TRIGger:WINDow:SLOP POS | Set window trigger to rising edge. |
| :TRIGger:WINDow:SLOP? | Query returns POS. |

### :TRIGger:WINDow:LEVel

- Command format：

  :TRIGger:WINDow:LEVel {LOW | HIGH | LH}

  :TRIGger:WINDow:LEVel?
- Functional description：

  This command is used to set level mode for window trigger.
- Return format：

  Query returns {LOW | HIGH | LH}.
- For example：

| | |
|---|---|
| :TRIGger:WINDow:LEV HIGH | Set window trigger to HIGH. |
| :TRIGger:WINDow:LEV? | Query returns HIGH. |

### :TRIGger:WINDow:TIME

- **Command format:**
  :TRIGger:WINDow:TIME <time>
  :TRIGger:WINDow:TIME?
- **Functional description:**
  This command is used to set time interval of window trigger.
- **Return format:**
  Query returns the current time interval, the unit is s.
- **For example:**

| | |
|---|---|
| :TRIGger:WINDow:TIME 1 | Set time interval of window trigger to 1s. |
| :TRIGger:WINDow:TIME? | Query returns 1.000e000. |

### :TRIGger:WINDow:POSition

- **Command format:**
  :TRIGger:WINDow:POSition {ENTER|EXIT|TIME}
  :TRIGger:WINDow:POSition?
- **Functional description:**
  This command is used to set window trigger position.
- **Return format:**
  Query returns {ENTER|EXIT|TIME}.
- **For example:**

| | |
|---|---|
| :TRIGger:WINDow:POS TIME | Set window trigger position to TIME. |
| :TRIGger:WINDow:POS? | Query returns TIME. |

## Delay Trigger

### :TRIGger:DELay:ARM:SOURce

- **Command format:**
  :TRIGger:DELay:ARM:SOURce {CHANnel1| CHANnel2| CHANnel3| CHANnel4}
  :TRIGger:DELay:ARM:SOURce?
- **Functional description:**
  This command is used to set focus source for delay trigger.
- **Return format:**
  Query returns {CHANnel1| CHANnel2| CHANnel3| CHANnel4}.
- **For example:**

| | |
|---|---|
| :TRIGger:DELay:ARM:SOUR CHAN1 | Set CH1 as focus source. |
| :TRIGger:DELay:ARM:SOUR? | Query returns CHANnel1. |

### :TRIGger:DELay:ARM:SLOPe

- **Command format:**
  :TRIGger:DELay:ARM:SLOPe {NEGative | POSitive}
  :TRIGger:DELay:ARM:SLOPe?
- **Functional description:**
  This command is used to set edge type for trigger source, which includes POSitive (rising edge) and NEGative (falling edge).
- **Return format:**
  Query returns {NEGative | POSitive}.
- **For example:**

```
:TRIGger:DELay:ARM:SLOPe NEG          Set edge type of trigger source to NEGative.
:TRIGger:DELay:ARM:SLOPe?             Query returns NEGative.
```

### :TRIGger:DELay:TRIGger:SOURce

➢ Command format:

:TRIGger:DELay:TRIGger:SOURce {CHANnel1| CHANnel2| CHANnel3| CHANnel4}

:TRIGger:DELay:TRIGger:SOURce?

➢ Functional description:

This command is used to set trigger source for delay trigger.

➢ Return format:

Query returns {CHANnel1| CHANnel2| CHANnel3| CHANnel4}.

➢ For example:

```
:TRIGger:DELay:TRIGger:SOUR CHAN1          Set CH1 as trigger source.
:TRIGger:DELay:TRIGger:SOUR?               Query returns CHANnel1.
```

### :TRIGger:DELay:TRIGger:SLOPe

➢ Command format:

:TRIGger:DELay:TRIGger:SLOPe {NEGative | POSitive}

:TRIGger:DELay:TRIGger:SLOPe?

➢ Functional description:

This command is used to set edge type for trigger source, which includes POSitive (rising edge) and NEGative (falling edge).

➢ Return format:

Query returns {NEGative | POSitive}.

➢ For example:

```
:TRIGger:DELay:TRIGger:SLOPe NEG          Set edge type of trigger source to NEGative.
:TRIGger:DELay:TRIGger:SLOPe?             Query returns NEGative.
```

### :TRIGger:DELay:QUALifier

➢ Command format:

:TRIGger:DELay:QUALifier { GREaterthan | LESSthan | INRange | OUTRange }

:TRIGger:DELay:QUALifier?

➢ Functional description:

This command is used to set time interval condition for delay trigger, which includes GREaterthan (greater than), LESSthan (less than), INRange (within in the range) and OUTRange (out of the range).

➢ Return format:

Query returns { GREaterthan | LESSthan | INRange | OUTRange }.

➢ For example:

```
:TRIGger:DELay:QUALifier GRE          Set slope condition to GREaterthan.
:TRIGger:DELay:QUALifier?             Query returns GREaterthan.
```

### :TRIGger:DELay:TIME:UPPer

➢ Command format:

:TRIGger:DELay:TIME:UPPer <time>

:TRIGger:DELay:TIME:UPPer?

➢ Functional description:

This command is used to set the upper time limit for delay trigger.

➢ Return format:

Query returns the current upper time limit, the unit is s.

- **For example:**

| | |
|---|---|
| :TRIGger:DELay:TIME:UPPer 1 | Set the upper time limit of trigger delay to 1s. |
| :TRIGger:DELay:TIME:UPPer? | Query returns 1.000e000. |

### :TRIGger:DELay:TIME:LOWer

- **Command format:**
  ```
  :TRIGger:DELay:TIME:LOWer     <time>
  :TRIGger:DELay:TIME:LOWer?
  ```
- **Functional description:**
  This command is used to set the lower limit of delay trigger time.
- **Return format:**
  Query returns the current lower limit of time, the unit is s.
- **For example:**

| | |
|---|---|
| :TRIGger:DELay:TIME:LOWer 1 | Set the lower limit of delay trigger time to 1s. |
| :TRIGger:DELay:TIME:LOWer? | Query returns 1.000e000. |

### :TRIGger:DELay:SELect

- **Command format:**
  ```
  :TRIGger:DELay:SELect     <SOURce<n>>
  :TRIGger:DELay:SELect
  ```
- **Functional description:**
  This command is used to switch the selected source. SOURce<n> represents source, n takes value from 1, 2.
  SOURce1 represents focus source; SOURce2 represents trigger source.
- **Return format:**
  Query returns { SOURce1 | SOURce2 }.
- **For example:**

| | |
|---|---|
| :TRIGger:DELay:SELect SOURce1 | Set to select the foucs trigger. |
| :TRIGger:DELay:SELect? | Query returns SOURce1. |

## Timeout Trigger

### :TRIGger:TIMEOUT:TIME

- **Command format:**
  ```
  :TRIGger:TIMEOUT:TIME     <time>
  :TRIGger:TIMEOUT:TIME?
  ```
- **Functional description:**
  This command is used to set time interval for timeout tigger.
- **Return format:**
  Query returns the current time interval, the unit is s.
- **For example:**

| | |
|---|---|
| :TRIGger:TIMEOUT:TIME 1 | Set time interval of timeout tigger to 1s. |
| :TRIGger:TIMEOUT:TIME? | Query returns 1.000e000. |

### :TRIGger:TIMEOUT:SLOPe

- **Command format:**

:TRIGger:TIMEOUT:SLOPe {POSitive|NEGative|ALTernation}
:TRIGger:TIMEOUT:SLOPe?

- Functional description:
  This command is used to set edge type for trigger source, which includes POSitive (rising edge), NEGative (falling edge) and ALTernation (rising/falling edge).
- Return format:
  Query returns edge type of trigger source {POSitive|NEGative|ALTernation}.
- For example:

| | |
|---|---|
| :TRIGger:TIMEOUT:SLOP POS | Set edge trigger to POSitive. |
| :TRIGger:TIMEOUT:SLOP? | Query returns POSitive. |

## Duration Trigger

### :TRIGger:DURation:PATTern

- Command format:
  :TRIGger:DURation:PATTern { HIGH | LOW | X }
  :TRIGger:DURation:PATTern?
- Functional description:
  This command is used to set pattern for duration trigger, which is HIGH (pattern value is 1), LOW (pattern value is 0) and X (channel is invalid).
- Return format:
  Query returns { HIGH | LOW | X }.
- For example:

| | |
|---|---|
| :TRIGger:DURation:PATTern HIGH | Set pattern of duration trigger to 1. |
| :TRIGger:DURation:PATTern? | Query returns HIGH. |

### :TRIGger:DURation:QUALifier

- Command format:
  :TRIGger:DURation:QUALifier { GREaterthan | LESSthan | INRange }
  :TRIGger:DURation:QUALifier?
- Functional description:
  This command is used to set time interval condition for delay trigger, which includes GREaterthan (greater than), LESSthan (less than) and INRange (within the range).
- Return format:
  Query returns { GREaterthan | LESSthan | INRange }.
- For example:

| | |
|---|---|
| :TRIGger:DURation:QUALifier GRE | Set slope condition to GREaterthan. |
| :TRIGger:DURation:QUALifier? | Query returns GREaterthan. |

### :TRIGger:DURation:TIME:LOWer

- Command format:
  :TRIGger:DURation:TIME:LOWer <time>
  :TRIGger:DURation:TIME:LOWer?
- Functional description:
  This command is used to set the lower limit of duration trigger time. It can be set when time interval condition is GREaterthan.
- Return format:
  Query returns the current time limit, the unit is s.
- For example:

| | |
|---|---|
| :TRIGger:DURation:TIME:LOWer 1 | Set the lower limit of duration trigger time to 1s. |

```
:TRIGger:DURation:TIME:LOWer?        Query returns 1.000e000.
```

### :TRIGger:DURation:TIME:UPPer

- **Command format:**
  :TRIGger:DURation:TIME:UPPer <time>
  :TRIGger:DURation:TIME:UPPer?
- **Functional description:**
  This command is used to set the upper limit of duration trigger time. It can be set when time interval condition is LESSthan (less than).
- **Return format:**
  Query returns the current upper limit of time, the unit is s.
- **For example:**

```
:TRIGger:DURation:TIME:UPPer 1       Set the upper time limit of duration trigger to 1s.
:TRIGger:DURation:TIME:UPPer?        Query returns 1.000e000.
```

## Setup & Hold Trigger

### :TRIGger:SHOLd:DATA:SOURce

- **Command format:**
  :TRIGger:SHOLd:DATA:SOURce {CHANnel1| CHANnel2| CHANnel3| CHANnel4}
  :TRIGger:SHOLd:DATA:SOURce?
- **Functional description:**
  This command is used to set data source for setup&hold trigger.
- **Return format:**
  Query returns { CHANnel1| CHANnel2| CHANnel3| CHANnel4 }.
- **For example:**

```
:TRIGger:SHOLd:DATA:SOUR CHAN1       Set CH1 as data source.
:TRIGger:SHOLd:DATA:SOUR?            Query returns CHANnel1.
```

### :TRIGger:SHOLd:CLOCK:SOURce

- **Command format:**
  :TRIGger:SHOLd:CLOCK:SOURce { CHANnel1| CHANnel2| CHANnel3| CHANnel4 }
  :TRIGger:SHOLd:CLOCK:SOURce?
- **Functional description:**
  This command is used to set clock source for setup&hold trigger.
- **Return format:**
  Query returns { CHANnel1| CHANnel2| CHANnel3| CHANnel4 }.
- **For example:**

```
:TRIGger:SHOLd:CLOCK:SOUR CHAN1      Set CH1 as clock source.
:TRIGger:SHOLd:CLOCK:SOUR?           Query returns CHANnel1.
```

### :TRIGger:SHOLd:SLOPe

- **Command format:**
  :TRIGger:SHOLd:SLOPe {POSitive|NEGative}
  :TRIGger:SHOLd:SLOPe?
- **Functional description:**
  This command is used to set edge type for setup&hold trigger, which includes POSitive (rising edge) and NEGative (falling edge).

- **Return format:**
  Query returns {POSitive|NEGative}.
- **For example:**

| | |
|---|---|
| :TRIGger:SHOLd:SLOPe POS | Set setup&hold trigger to POSitive. |
| :TRIGger:SHOLd:SLOPe? | Query returns POSitive. |

## :TRIGger:SHOLd:PATTern

- **Command format:**
  :TRIGger:SHOLd:PATTern { HIGH | LOW }
  :TRIGger:SHOLd:PATTern?
- **Functional description:**
  This command is used to set pattern for setup&hold trigger, which includes HIGH (pattern value is 1) and LOW (pattern value is 0).
- **Return format:**
  Query returns { HIGH | LOW }.
- **For example:**

| | |
|---|---|
| :TRIGger:SHOLd:PATTern HIGH | Set pattern of setup&hold trigger to 1. |
| :TRIGger:SHOLd:PATTern? | Query returns HIGH. |

## :TRIGger:SHOLd:MODE

- **Command format:**
  :TRIGger:SHOLd:MODE { SETup | HOLD }
  :TRIGger:SHOLd:MODE?
- **Functional description:**
  This command is used to set time mode for setup&hold trigger, which includes SETup (setup time), HOLD (hold time).
- **Return format:**
  Query returns { SETup | HOLD }.
- **For example:**

| | |
|---|---|
| :TRIGger:SHOLd:MODE HOLD | Set time mode to HOLD. |
| :TRIGger:SHOLd:MODE? | Query returns HOLD. |

## :TRIGger:SHOLd:TIME

- **Command format:**
  :TRIGger:SHOLd:TIME <time>
  :TRIGger:SHOLd:TIME?
- **Functional description:**
  This command is used to set time interval for setup&hold trigger.
- **Return format:**
  Query returns the current time interval, the unit is s.
- **For example:**

| | |
|---|---|
| :TRIGger:SHOLd:TIME 1 | Set time interval of setup&hold trigger to 1s. |
| :TRIGger:SHOLd:TIME? | Query returns 1.000e000. |

## :TRIGger:SHOLd:SELect

- **Command format:**
  :TRIGger:SHOLd:SELect <SOURce<n>>

:TRIGger:SHOLd:SELect

- **Functional description:**
  This command is used to switch the selected source. SOURce<n> represents source, n takes value from 1, 2.
  SOURce1 represents data source; SOURce2 represents clock source.
- **Return format:**
  Query returns { SOURce1 | SOURce2 }.
- **For example:**

| | |
|---|---|
| :TRIGger:SHOLd:SELect SOURce1 | Set the selected focus source. |
| :TRIGger:SHOLd:SELect? | Query returns SOURce1. |

## Nth-edge Trigger

### :TRIGger:NEDGE:SLOPe

- **Command format:**
  :TRIGger:NEDGE:SLOPe {POSitive|NEGative }
  :TRIGger:NEDGE:SLOPe?
- **Functional description:**
  This command is used to set the edge type of trigger, which includes POSitive (rising edge) and NEGative (falling edge).
- **Return format:**
  Query returns the edge type of trigger {POSitive|NEGative}.
- **For example:**

| | |
|---|---|
| :TRIGger:NEDGE:SLOP POS | Set edge trigger to POSitive. |
| :TRIGger:NEDGE:SLOP? | Query returns POSitive. |

### :TRIGger:NEDGE:TIME

- **Command format:**
  :TRIGger:NEDGE:TIME <time>
  :TRIGger:NEDGE:TIME?
- **Functional description:**
  This command is used to set time interval for Nth-edge trigger.
- **Return format:**
  Query returns the current time interval, the unit is s.
- **For example:**

| | |
|---|---|
| :TRIGger:NEDGE:TIME 1 | Set time interval of Nth edge trigger to 1s. |
| :TRIGger:NEDGE:TIME? | Query returns 1.000e000. |

### :TRIGger:NEDGE:VALue

- **Command format:**
  :TRIGger:NEDGE:VALue <value>
  :TRIGger:NEDGE:VALue?
- **Functional description:**
  This command is used to set Nth-edge value, <value> is integer value and the range can set to 0~65535.
- **Return format:**
  Query returns the current Nth-edge value.
- **For example:**

| | |
|---|---|
| :TRIGger:NEDGE:VALue 100 | Set Nth-edge value to 100. |
| :TRIGger:NEDGE:VALue? | Query returns 100. |

## Pattern Trigger

- **Command format:**
  :TRIGger:PATTern:PATTern { HIGH | LOW | X | POSitive | NEGative }
  :TRIGger:PATTern:PATTern?
- **Functional description:**
  This command is used to set pattern trigger, which includes HIGH (pattern value is 1), LOW (pattern value is 0), X (channel is invalid), POSitive (rising edge), NEGative (falling edge).
- **Return format:**
  Query returns { HIGH | LOW | X | POSitive | NEGative }.
- **For example:**

| | |
|---|---|
| :TRIGger:PATTern:PATTern HIGH | Set patter trigger to 1. |
| :TRIGger:PATTern:PATTern? | Query returns HIGH. |

## CURSor Command

This command is uset to set cursor parameter to measure the waveform data on the screen.

### :CURSor:MODE

- **Command format:**
  :CURSor:MODE { TRACK | INDependent }
  :CURSor:MODE?
- **Functional description:**
  This command is used to set cursor mode, which includes TRACK, INDependent.
- **Return format:**
  Query returns { TRACK | INDependent }.
- **For example:**

| | |
|---|---|
| :CURSor:MODE TRACK | Set cursor mode to TRACK. |
| :CURSor:MODE? | Query returns TRACK. |

### :CURSor:TYPE

- **Command format:**
  :CURSor:TYPE {AMPlitude | TIME | SCReen | CLOSe }
  :CURSor:TYPE?
- **Functional description:**
  This command is used to set cursor type for cursor mode, which includes AMPlitude, TIME, SCReen, CLOSe.
- **Return format:**
  Query returns {AMPlitude | TIME | SCReen | CLOSe }.
- **For example:**

| | |
|---|---|
| :CURSor:TYPE AMP | Set cursor type to AMPlitude. |
| :CURSor:TYPE? | Query returns AMPlitude. |

### :CURSor:SOURce

- **Command format:**
  :CURSor:SOURce <source>
  :CURSor:SOURce?
- **Functional description:**
  This command is used to set cursor source for manual cursor mode.

<source> takes value from {CHANnel<n>|MATH| REFA | REFB | REFC | REFD }， n takes value from 1, 2, 3, 4.

- **Return format：**
  Query returns {CHANnel<n>|MATH| REFA | REFB | REFC | REFD}.

- **For example：**

| | |
|---|---|
| :CURSor:SOURce CHAN1 | Set CH1 as cursor source. |
| :CURSor:SOURce? | Query returns CHANnel1. |

## :CURSor:CURAX

- **Command format：**
  :CURSor:CURAX <value>
  :CURSor:CURAX?
- **Functional description：**
  This command is use to set the horizontal position of cursor line A. Timebase YT mode range is form left to right [0,699], timebase XY mode range is from left to right [28,227].
- **Return format：**
  Query returns the horizontal position of cursor line A.
- **For example：**

| | |
|---|---|
| :CURSor:CURAX 50 | Set the horizontal position of manual cursor line A position to 50. |
| :CURSor:CURAX? | Query returns 50. |

## :CURSor:CURAY

- **Command format：**
  :CURSor:CURAY <value>
  :CURSor:CURAY?
- **Functional description：**
  This command is use to set the vertical position of cursor line A. The range is form up to down [28,227].
- **Return format：**
  Query returns the vertical position of cursor line A.
- **For example：**

| | |
|---|---|
| :CURSor:CURAY 50 | Set the vetical position of manual cursor line A position to 50. |
| :CURSor:CURAY? | Query returns 50. |

## :CURSor:CURBX

- **Command format：**
  :CURSor:CURBX <value>
  :CURSor:CURBX?
- **Functional description：**
  This command is use to set the horizontal position of cursor line B. Timebase YT mode range is form left to right [0,699], timebase XY mode range is from left to right [28,227].
- **Return format：**
  Query returns the horizontal position of cursor line B.
- **For example：**

| | |
|---|---|
| :CURSor:CURBX 50 | Set the vetical position of manual cursor line B position to 50. |
| :CURSor:CURBX? | Query returns 50. |

## :CURSor:CURBY

- **Command format：**
  :CURSor:CURBY <value>

:CURSor:CURBY?

- Functional description:
  This command is use to set the vertical position of cursor line B. The range is form top to bottom [28,227].
- Return format:
  Query returns the vertical position of cursor line B.
- For example:
  :CURSor:CURBY 50 Set the vetical position of manual cursor line B position to 50.
  :CURSor:CURBY? Query returns 50.

## :CURSor:AXValue?

- Command format:
  :CURSor:AXValue?
- Functional description:
  Query X value at cursor A, unit is determined by the amplitude unit of the current selected channel.
- Return format:
  Query returns X value at the current cursor A in scientific notation.
- For example:
  :CURSor:AXV? Query returns 2.000000E+02.

## :CURSor:AYValue?

- Command format:
  :CURSor:AYValue?
- Functional description:
  Query Y value at cursor A, unit is determined by the amplitude unit of the current selected channel.
- Return format:
  Query returns Y value at the current cursor A in scientific notation.
- For example:
  :CURSor:AYV? Query returns 2.000000E+02.

## :CURSor:BXValue?

- Command format:
  :CURSor:BXValue?
- Functional description:
  Query X value at cursor B, unit is determined by the amplitude unit of the current selected channel.
- Return format:
  Query returns X value at the current cursor B in scientific notation.
- For example:
  :CURSor:BXV? Query returns 2.000000E+02.

## :CURSor:BYValue?

- Command format:
  :CURSor:BYValue?
- Functional description:
  Query Y value at cursor B, unit is determined by the amplitude unit of the current selected channel.
- Return format:
  Query returns Y value at the current cursor B in scientific notation.
- For example:
  :CURSor:BYV? Query returns 2.000000E+02.

## :CURSor:XDELta?

- Command format:
  :CURSor:XDELta?
- Functional description:
  Query the X difference△X between cursor A and cursor B in cursor tracking measurement.
- Return format:
  Query returns the current X difference between cursor A and cursor B in scientific notation.
- For example:

| | |
|---|---|
| :CURSor:XDEL? | Query returns 2.000000E+02 |

## :CURSor:YDELta?

- Command format:
  :CURSor:YDELta?
- Functional description:
  Query the Y difference△Y between cursor A and cursor B in cursor tracking measurement. The unit is the same as the unit of the currently selected channel.
- Return format:
  Query returns the current Y difference between cursor A and cursor B in scientific notation.
- For example:

| | |
|---|---|
| :CURSor:YDEL? | Query returns 2.000000E+02 |

## :CURSor:XY?

- Command format:
  :CURSor:XY?
- Functional description:
  Query all the measurement parameter of cursor measurement command :DISPlay:TYPE in XY mode.
- Return format:
  Query returns all the measurement parameter in XY mode. The numerical value with the standard unit and returned in scientific notation. The return order refers to Appendix 3：Cursor Measurement in XY Mode . The order is from up to down in sequence.
- For example:

| | |
|---|---|
| :CURSor:XY? | Query returns all cursor measurement parameter in XY12 mode, such as "1.0E+02, 3.0E+01,………" |

## :CURSor:XUNITs

- Command format:
  :CURSor:XUNITs {SECond | HZ}
  :CURSor:XUNITs?
- Functional description:
  This command is used to set the cursor's horizontal unit. SECond represents second. Hz represents Hertz
- Return format:
  Query returns {SECond | HZ}.
- For example:

| | |
|---|---|
| :CURSor:XUNITs SECond | Set the cursor' s horizontal unit to second. |
| :CURSor:XUNITs? | Query returns SECond. |

## FILE Command

This command is used to set reference waveform and storage function.

### :FILE:LOAD

- ➢ **Command format：**

  :FILE:LOAD <filename>, [<source>], [<disk>]
- ➢ **Functional description：**

  This command is used to load waveform to the reference channel or set the data.

  <filename> represents the filename or file path, the filename must be character string data with double quotation mark, such as "test.csv".
  - ■ The file name of *.csv or *.wav represent that load waveform data of a certain file into the reference channel, it matched with the oscilloscope's suffix name.
  - ■ The file name of *.dat represent that load setting data of a certain file into the oscilloscope, it matched with the oscilloscope's suffix name.

  <source > represents reference channel { REFA | REFB | REFC | REFD }, optional parameter，it only valid when loading waveform data.
  - ■ REFA represents reference channel A
  - ■ REFB represents reference channel B
  - ■ REFC represents reference channel C
  - ■ REFD represents reference channel D

  <disk> represents memory medium { FLASH | UDISK }, optional parameter，omitting represents internal data of FLASH.
  - ■ FLASH represents internal data.
  - ■ UDISK represents U flash disk data.
- ➢ **For example：**

  FILE:LOAD "test.wav",REFA,UISK

  Load test.wav waveform data from USB into reference channel A.

  FILE:LOAD "system-set-up01.dat"

  Load position 1 configuration data from internal medium into the oscilloscope.

Notes：When the oscilloscope can not self-defined the file name and suffix

- ● The file name of internal storage setting must be"system-set-up01.set"~ "system-set-up255.set", the maximum limit is 255 files.
- ● The file name of internal storage bsv file must be"wave01.bsv"~" wave255.bsv", the maximum limit is 255 files.

### :FILE:SAVE

- ➢ **Command format：**

  :FILE:SAVE <filename>[,<source>][,<disk>][,<startframe>,< endframe>,<framerate>]
- ➢ **Functional description：**

  This command is used to save the channel waveform or set the data or recording data into file.

  <filename> represents the filename or file path, the filename must be character string data with double quotation mark, such as "test.csv".
  - ■ The file name of *.csv or *.wav represent that to save the waveform of a certain file into the file with suffix name, it matched with the oscilloscope's suffix name.
  - ■ The file name of *.dat represent that save the setting data of a certain file into the file, it matched with the oscilloscope's suffix name.
  - ■ The file name of *.mp4 represent that save a recording video to the file, it matched with the oscilloscope's suffix name.
  - ■ The file name of *.bmp or *.jpeg or *.png represent that save a picture to the file, it matched with the oscilloscope's suffix name.

  <source > represents physical channel { CHANnel1| CHANnel2| CHANnel3| CHANnel4 }, optional parameter，it only valid when loading waveform data.

- CHANnel1 represents CH1.
- CHANnel2 represents CH2.
- CHANnel3 represents CH3.
- CHANnel4 represents CH4.

<disk> represents storage medium { FLASH | UDISK }, optional parameter，omitting represents internal data of FLASH.

- FLASH represents internal data.
- UDISK represents U flash disk data.

<startframe> represents the start frame of saved recoding video.
<endframe> represents the end frame of saved recoding video.
<framerate> represents the frame rate of saved recoding video.

➢ For example：
FILE:SAVE "test.csv",CHANnel1,UDISK
Save CH1 waveform data as test.csv format into USB.
FILE:SAVE "system-set-up01.dat"
The configuration data of oscilloscope save as 1 position of internal medium.
FILE:SAVE "wave01.wav",CHANnel1,FLASH
Save CH1 waveform data into internal medium.
FILE:SAVE "wave01.wav",CHANnel1
Save CH1 waveform data into internal medium.
FILE:SAVE "system-set-up01.dat",FLASH
The configuration data of oscilloscope save as internal medium.
FILE:SAVE "system-set-up01.dat"
The configuration data of oscilloscope save as 1 position of internal medium.
FILE:SAVE "test.mp4",1,100,10
Save recording video in start frame 1, end frame 100, frame rate 10 to test.mp4 file in USB.
FILE:SAVE "test.png", UDISK
Save picture data in png format to test.png file in USB.

Notes：When the oscilloscope can not self-defined the file name and suffix

- The file name of internal storage setting must be"system-set-up01.set"~ "system-set-up255.set", maximum limit 255 files.
- The file name of internal storage bsv file must be"wave01.bsv"~" wave255.bsv", maximum limit 255 files.

## REFerence Command

This command is used to set reference waveform function of the oscilloscope.

### :REFerence:CLEar

➢ Command format：
:REFerence:CLEar {REFA|REFB|REFC|REFD|ALL}

➢ Functional description：
This command is used to clear reference channel waveform , ALL indicates clearing all reference channel waveforms .

➢ For example：

| | |
|---|---|
| :REFerence:CLEar REFA | Clear waveform of reference channel A. |

### :REFerence:LOAD

➢ Command format：
:REFerence:LOAD <source>[,<ref>]

➢ Functional description：
This command is used to quickly load the physical channel waveform into the reference channel. If the reference channel is omitted, it will be loaded into the currently selected reference channel.
<source>: CHANnel<n>, takes value from 1, 2, 3, 4.

<ref>：{ REFA | REFB | REFC | REFD }，Representing four reference channels A, B, C, and D respectively.

- For example：
  :REFerence:LOAD CHANnel1,REFB
  Loading the waveform of CH1 into the reference channel B.

## RECord Command

This command is used to set recording waveform function of the oscilloscope.

### :RECord:ENABle

- Command format：
  :RECord:ENABle { {1|ON} | {0|OFF} }
  :RECord:ENABle?
- Functional description：
  This command is used to turn on/off recording waveform function.
- Return format：
  Query returns 1 or 0, it represents ON or OFF.
- For example：
  :RECord:ENABle ON
  Turn on recording waveform function.
  RECord:ENABle?
  Query returns 1, it represents recording waveform function is turned on.

### :RECord:STARt

- Command format：
  :RECord:STARt { {1|ON} | {0|OFF} }
  :RECord:STARt?
- Functional description：
  This command is used to start/stop recording waveform.
- Return format：
  Query returns 1 or 0, it represents ON or OFF.
- For example：
  :RECord:STARt ON
  Start to recording waveform.
  :RECord:STARt?
  Query returns 1, it represents the recording waveform is started.

### :RECord:FAST

- Command format：
  :RECord:FAST { {1|ON} | {0|OFF} }
  :RECord:FAST?
- Functional description：
  This command is used to turn on/off quick recoding function.
- Return format：
  Query returns 1 or 0, it represents ON or OFF.
- For example：
  :RECord:FAST ON
  Turn on quick recording function.
  :RECord:FAST?
  Query returns 1, it represents quick recording function is turned on.

## :RECord:INTerval

- Command format:
  :RECord:INTerval <time>
  :RECord:INTerval?
- Functional description:
  This command is used to set time interval for recording waveform.
- Return format:
  Query returns time interval of recording waveform in scientific notation, the unit is s.
- For example:

| | |
|---|---|
| :RECord:INTerval 200ns | Set play delay time of recording waveform to 200ns. |
| :RECord:INTerval? | Query returns 2.000e-004. |

## :RECord:PLAY

- Command format:
  :RECord:PLAY { {1|ON} | {0|OFF} }
  :RECord:PLAY?
- Functional description:
  This command is used to start or stop play recording waveform.
- Return format:
  Query returns 1 or 0, it represents ON or OFF.
- For example:
  :RECord:PLAY ON
  Start to play recording waveform.
  :RECord:PLAY?
  Query returns 1, it represent that the recording waveform is start to play.

## :RECord:PLAY:DELay

- Command format:
  :RECord:PLAY:DELay <time>
  :RECord:PLAY:DELay?
- Functional description:
  This command is used to set delay time for play recording waveform.
- Return format:
  Query returns delay time of play recording waveform in scientific notation, the unit is s.
- For example:
  :RECord:PLAY:DELay 20ms
  Set delay time for play recording waveform to 20ms.
  :RECord:PLAY:DELay?
  Query returns 2.000e-002.

## :RECord:CURRent

- Command format:
  :RECord:CURRent <value>
  :RECord:CURRent?
- Functional description:
  This command is used to set or query the current frame of recording waveform.
- Return format:
  Query returns the curren frame of recording waveform, it is integer data.
- For example:

| | |
|---|---|
| :RECord:CURRent 100 | Set the current frame of recording waveform to 100. |
| :RECord:CURRent? | Query returns 100. |

### :RECord:FRAMes

- **Command format:**
  :RECord:FRAMes <value>
  :RECord:FRAMes?
- **Functional description:**
  This command is used to set or query the frame number of recording waveform.
- **Return format:**
  Query returns the frame of recording waveform, it is integer data.
- **For example:**

| | |
|---|---|
| :RECord:FRAMes 400 | Set the frame number of recording waveform to 400. |
| :RECord:FRAMes? | Query returns 400. |

## DVM Command

This command is used to set digital voltage meter.

### :DVM:ENABle

- **Command format:**
  :DVM:ENABle { {1|ON} | {0|OFF} }
  :DVM:ENABle?
- **Functional description:**
  This command is used to set or query the status of digital voltage meter function (ON or OFF).
- **Return format:**
  Query returns 1 or 0, it represents ON or OFF.
- **For example:**

| | |
|---|---|
| :DVM:ENABle ON | Turn on digital voltage meter. |
| :DVM:ENABle? | Query returns 1. |

### :DVM:SOURCe

- **Command format:**
  :DVM:SOURCe <source>
  :DVM:SOURCe?
- **Functional description:**
  This command is used to set or query the information source of digital voltage meter.
  <source>: CHANnel<n>, n takes value from 1, 2, 3, 4.
- **Return format:**
  Query returns { CHANnel1 | CHANnel2| CHANnel3| CHANnel4 }.
- **For example:**

| | |
|---|---|
| :DVM:SOURCe CHANnel1 | Set source as CH1. |
| :DVM:SOURCe? | Query returns CHANnel1. |

### :DVM:MODE

- **Command format:**
  :DVM:MODE {ACRMs|DC|DCRMs}
  :DVM:MODE?
- **Functional description:**
  This command is used to set or query the mode of digital voltage meter.
- **Return format:**
  Query returns {ACRMs|DC|DCRMs}.
- **For example:**

| | |
|---|---|
| :DVM:MODE DC | Set the mode of digital voltage meter to DC. |
| :DVM:MODE? | Query returns DC. |

## :DVM:CURRent?

- **Command format:**
  : DVM:CURRent?
- **Functional description:**
  This command is used to query the current measured voltage value of digital voltage meter.
- **Return format:**
  Query returns the current measured voltage value of digital voltage meter in scientific notation, unit is determined by:DVM:MODE.
- **For example:**

| | |
|---|---|
| :DVM:CURRent? | The current measured voltage value is 3.000e-003. |

## :DVM:LIMit:ENABle

- **Command format:**
  :DVM:LIMit:ENABle { {1|ON} | {0|OFF} }
  :DVM:LIMit:ENABle?
- **Functional description:**
  This command is used to turn on/off the limit alarm function.
- **Return format:**
  Query returns 1 or 0, it represents ON or OFF.
- **For example:**

| | |
|---|---|
| :DVM:LIMit:ENABle ON | Turn on the limit alarm function. |
| :DVM:LIMit:ENABle? | Query returns 1. |

## :DVM:LIMit:QUALifier

- **Command format:**
  :DVM:LIMit:QUALifier {GREaterthan | LESSthan | INRange | OUTRange}
  :DVM:LIMit:QUALifier?
- **Functional description:**
  This command is used to set the limit alarm condition for digital voltage meter, which includes GREaterthan, LESSthan, INRange (between), OUTRange.
- **Return format:**
  Query returns { GREaterthan | LESSthan | INRange | OUTRange }.
- **For example:**

| | |
|---|---|
| :DVM:LIMit:QUALifier GRE | Set the limit alarm condition to GREaterthan. |
| :DVM:LIMit:QUALifierr? | Query returns GREaterthan. |

## :DVM:LIMit:UPPer

- **Command format:**
  :DVM:LIMit:UPPer <upper>
  :DVM:LIMit:UPPer?
- **Functional description:**
  This command is used to set the upper limit of alarm for digital voltage meter.
- **Return format:**
  Query returns the upper limit in scientific notation, the unit is V.
- **For example:**

| | |
|---|---|
| :DVM:LIMit:UPPer 2V | Set the upper limit to 2V. |
| :DVM:LIMit:UPPer? | Query returns 2.000e000. |

## :DVM:LIMit:LOWer

- **Command format:**

:DVM:LIMit:LOWer <lower >
:DVM:LIMit:LOWer?

- **Functional description:**
  This command is used to set the lower limit of alarm for digital voltage meter.
- **Return format:**
  Query returns the lower limit in scientific notation, the unit is V.
- **For example:**

| | |
|---|---|
| :DVM:LIMit:LOWer 1V | Set the lower limit to 1V. |
| :DVM:LIMit:LOWer? | Query returns 1..000e000. |

## PF Command

This command is used to set Pass/Fail test function.

### :PF:ENABle

- **Command format:**
  :PF:ENABle { {1|ON} | {0|OFF} }
  :PF:ENABle?
- **Functional description:**
  This command is used to set or query Pass/Fail test function (ON or OFF).
- **Return format:**
  Query returns 1 or 0, it represents ON or OFF.
- **For example:**

| | |
|---|---|
| :PF:ENABle ON | Turn on Pass/Fail test function. |
| :PF:ENABle? | Query returns 1. |

### :PF:SOURCe

- **Command format:**
  :PF:SOURCe <source>
  :PF:SOURCe?
- **Functional description:**
  This command is used to set or query the measurement source of Pass/Fail test.
  <source>: CHANnel<n>, takes value from 1, 2, 3, 4.
- **Return format:**
  Query returns { CHANnel1 | CHANnel2| CHANnel3| CHANnel4 }.
- **For example:**

| | |
|---|---|
| :PF:SOURCe CHANnel1 | Set the measurement source as CH1. |
| :PF:SOURCe? | Query returns CHANnel1. |

### :PF:OPERate

- **Command format:**
  :PF:OPERate {RUN|STOP}
  :PF:OPERate?
- **Functional description:**
  This command is used to run or stop Pass/Fail test.
- **Return format:**
  Query returns {RUN|STOP}.
- **For example:**

| | |
|---|---|
| :PF:OPERate RUN | Run Pass/Fail test.. |
| :PF:OPERate? | Query returns RUN. |

### :PF:OUTPut

- **Command format:**
  :PF:OUTPut {PASS|FAILED}
  :PF:OUTPut?
- **Functional description:**
  This command is used to set or query the output of Pass/Fail test.
- **Return format:**
  Query returns {PASS|FAILED}.
- **For example:**

| | |
|---|---|
| :PF:OUTPut PASS | Set the output of Pass/Fail test to PASS. |
| :PF:OUTPut? | Query returns PASS. |

### :PF:STOP:TYPe

- **Command format:**
  :PF:STOP:TYPe {PCOUNT|FCOUNT}
  :PF:STOP:TYPe?
- **Functional description:**
  This command is used to set or query stop type of Pass/Fail test.
  PCOUNT represents the number of pass; FCOUNT represents the number of fail.
- **Return format:**
  Query returns {PCOUNT|FCOUNT}.
- **For example:**

| | |
|---|---|
| :PF:STOP:TYPe PCOUNT | Set stop type of Pass/Fail test to PCOUNT. |
| :PF:STOP:TYPe? | Query returns PCOUNT. |

### :PF:STOP:QUALifier

- **Command format:**
  :PF:STOP:QUALifier {LEQual | GEQual}
  :PF:STOP:QUALifier?
- **Functional description:**
  This command is used to set or query the stop condition of Pass/Fail test.
  GEQual represents greater than or equal to; LEQual represents less than or equal to.
- **Return format:**
  Query returns {LEQual | GEQual}.
- **For example:**

| | |
|---|---|
| :PF:STOP:QUALifier GEQual | Set the stop condition of Pass/Fail test to ≥. |
| :PF:STOP:QUALifier? | Query returns GEQual. |

### :PF:STOP:THReshold

- **Command format:**
  :PF:STOP:THReshold <value>
  :PF:STOP:THReshold?
- **Functional description:**
  This command is used to set or query the stop threshold of Pass/fail test.

<value>: stop threshold, range is 1~30000, the specified range will self-adapting according to the oscilloscope.

- **Return format:**
  Query returns stop threshold, it is integer data.
- **For example:**

| | |
|---|---|
| :PF:STOP:THReshold 100 | Set the stop threshold of Pass/fail test to 100. |
| :PF:STOP:THReshold? | Query returns 1.00. |

## :PF:TEMPlate:SOURce

- Command format：
  :PF:TEMPlate:SOURce <source>
  :PF:TEMPlate:SOURce?
- Functional description：
  This command is used to set or query the reference channel of template setting of Pass/fail test.
  <source>: CHANnel<n>, n takes value from 1, 2, 3, 4.
- Return format：
  Query returns { CHANnel1 | CHANnel2| CHANnel3| CHANnel4 }.
- For example：
  :PF:TEMPlate:SOURce CHANnel1
  Set the reference channel of template setting as CH1.
  :PF:TEMPlate:SOURce?
  Query returns CHANnel1.

## :PF:TEMPlate:X

- Command format：
  :PF:TEMPlate:X <value>
  :PF:TEMPlate:X?
- Functional description：
  This command is used to set or query the horizontal tolerance for template setting of Pass/fail test.
  <value>: horizontal tolerance, range is 1~100, the specified range will self-adapting according to the oscilloscope.
- Return format：
  Query returns horizontal tolerance of template setting, it is integer data.
- For example：

| | |
|---|---|
| :PF:TEMPlate:X 50 | Set horizontal tolerance of template setting to 50. |
| :PF:TEMPlate:X? | Query returns 50. |

## :PF:TEMPlate:Y

- Command format：
  :PF:TEMPlate:Y <value>
  :PF:TEMPlate:Y?
- Functional description：
  This command is used to set or query the vertical tolerance for template setting of Pass/fail test.
  <value>: vertical tolerance, range is 1~100, the specified range will self-adapting according to the oscilloscope.
- Return format：
  Query returns vertical tolerance of template setting, it is integer data.
- For example：

| | |
|---|---|
| :PF:TEMPlate:Y 50 | Set vertical tolerance of template setting to 50. |
| :PF:TEMPlate:Y? | Query returns 50. |

## :PF:RESult?

- Command format：
  :PF:RESult?
- Functional description：
  This command is used to query the statistical result of Pass/Fail test.
  Return data format: <pass>, <failed>, <total>, <pass> represents the number of pass; <failed> represents the number of fail; <total> present the total number.
- Return format：

Query returns the statistical result of Pass/Fail test.

➢ For example:

| | |
|---|---|
| :PF:RESult? | Query returns 35, 42, 77. |

## ACQuire Command

This command is used to set the sampling mode for the oscilloscope.

### :ACQuire:TYPE

➢ Command format:
:ACQuire:TYPE {NORMal | AVERage | PEAKdetect | HRESolution }
:ACQuire:TYPE?

➢ Functional description:
This command is used to set sampling mode of the oscilloscope, which is NORMal, AVERage, PEAKdetect, HRESolution.

➢ Return format:
Query returns {NORMal | AVERage | PEAKdetect | HRESolution }.

➢ For example:

| | |
|---|---|
| :ACQ:TYPE AVER | Set sampling mode to AVERage. |
| :ACQ:TYPE? | Query returns AVERage. |

### :ACQuire:AVERages:COUNt

➢ Command format:
:ACQuire:AVERages:COUNt <count>
:ACQuire:AVERages:COUNt?

➢ Functional description:
This command is used to set the number of average sampling mode, <count> stepped as Nth power of 2, takes value form 2~8192, 1≤N≤30.

➢ Return format:
Query returns the current number of average sampling mode.

➢ For example:

| | |
|---|---|
| :ACQ:AVER:COUN 32 | Set the number of average sampling mode to 32. |
| :ACQ:AVER:COUN? | Query returns 32. |

### :ACQuire:MEMory:DEPTh

➢ Command format:
:ACQuire:MEMory:DEPTh { AUTO | 7K | 70K | 700K | 7M | 28M | 56M }
:ACQuire:MEMory:DEPTh?

➢ Functional description:
This command is used to set storage depth of sampling mode, it will self-adapting according to the storage depth of different mode.

➢ Return format:
Query returns { AUTO | 7K | 70K | 700K | 7M | 28M | 56M }.

➢ For example:

| | |
|---|---|
| :ACQ:MEM:DEPT AUTO | Set storage depth to AUTO. |
| :ACQ:MEM:DEPT? | Query returns AUTO. |

## DISPlay Command

This command is used to set or query the diplay function or data of the oscilloscope.

## :DISPlay:DATA:FORMat

- **Command format:**
  :DISPlay:DATA:FORMat { BMP | JPEG | PNG}
  :DISPlay:DATA:FORMat?
- **Functional description:**
  This command is used to query the current image and the data format of local saved image.
- **Return format:**
  Query returns the image format { BMP | JPEG | PNG}.
- **For example:**

| | |
|---|---|
| :DISPlay:DATA:FORMat PNG | Set the image data that with PNG. |
| :DISPlay:DATA:FORMat? | Query returns PNG. |

## :DISPlay:DATA?

- **Command format:**
  :DISPlay:DATA?
- **Functional description:**
  This command is used to query the current image data, and return the image data with BMP format by default.
  The format of returned image is decide by the command :DISPlay:DATA:FORMat.
- **Return format:**
  Query returns the image data, returned data is conform to Appendix 2: IEEE 488.2 Binary System Data Format.
- **For example:**

| | |
|---|---|
| :DISPlay:DATA? | Query returns the image format. |

## :DISPlay:FORMat

- **Command format:**
  :DISPlay:FORMat { VECTors | DOTS }
  :DISPlay:FORMat?
- **Functional description:**
  This command is used to set diplay format for sampling point, which is VECTors (vector display), DOTS (direct display).
- **Return format:**
  Query returns { VECTors | DOTS }.
- **For example:**

| | |
|---|---|
| :DISPlay:FORMat VECT | Set diplay format of sampling point to VECTors. |
| :DISPlay:FORMat | Query returns VECTors. |

## :DISPlay:GRID

- **Command format:**
  :DISPlay:GRID {FULL|HALF|CROSS|NONE}
  :DISPlay:GRID?
- **Functional description:**
  This command is used to set diplay format of grid.
- **Return format:**
  Query returns {FULL|HALF|CROSS|NONE}.
- **For example:**
  :DISPlay:GRID CROSS
  Set the display format of grid to CROSS, which not display separate grids.
  :DISPlay:GRID?
  Query returns CROSS.

## :DISPlay:GRID:BRIGhtness

- **Command format:**
  :DISPlay:GRID:BRIGhtness <count>
  :DISPlay:GRID:BRIGhtness?
- **Functional description:**
  This command is used to set the grid brightness, <count> takes value from 1~100, the larger the number, the brighter the grid.
- **Return format:**
  Query returns the current grid brightness.
- **For example:**

| | |
|---|---|
| :DISPlay:GRID:BRIGhtness 50 | Set the grid brightness to 50. |
| :DISPlay:GRID:BRIGhtness? | Query returns 50. |

## :DISPlay:GRAD:TIME

- **Command format:**
  :DISPlay:GRAD:TIME {DSO|MINimum|50ms|100ms|200ms|500ms|1s|2s|5s|10s|20s|INFinite}
  :DISPlay:GRAD:TIME?
- **Functional description:**
  This command is used to set the persistence time. DSO represents the persistence is closed.
- **Return format:**
  Query returns {DSO|MINimum|50ms|100ms|200ms|500ms|1s|2s|5s|10s|20s|INFinite}.
- **For example:**

| | |
|---|---|
| :DISPlay:GRAD:TIME 50ms | Set the persistence time to 50ms. |
| :DISPlay:GRAD:TIME? | Query returns 50ms. |

## :DISPlay:COLOR

- **Command format:**
  :DISPlay:COLOR { {1|ON} | {0|OFF} }
  :DISPlay:COLOR?
- **Functional description:**
  This command is used to turn on/off color temperature display.
- **Return format:**
  Query returns 1 or 0, it represents ON or OFF.
- **For example:**

| | |
|---|---|
| :DISPlay:COLOR ON | Turn on color temperature display. |
| :DISPlay:COLOR? | Query returns 1, it represents color display is turned on. |

## :DISPlay:COLOR:INVERt

- **Command format:**
  :DISPlay:COLOR:INVERt { {1|ON} | {0|OFF} }
  :DISPlay:COLOR:INVERt?
- **Functional description:**
  This command is used to turn on/off inverse color.
- **Return format:**
  Query returns 1 or 0, it represents ON or OFF.
- **For example:**
  :DISPlay:COLOR:INVERt ON
  Turn on inverse color.
  :DISPlay:COLOR:INVERt?
  Query returns 1, it represents inverse color display is turned on.

## :DISPlay:WAVE:BRIGhtness

- **Command format:**
  :DISPlay:WAVE:BRIGhtness <count>
  :DISPlay:WAVE:BRIGhtness?
- **Functional description:**
  This command is used to set the waveform brightness, <count> takes value from 1~100, larger the number, the brighter the waveform.
- **Return format:**
  Query returns the current waveform brightness.
- **For example:**

| | |
|---|---|
| :DISPlay:WAVE:BRIGhtness 50 | Set the waveform brightness to 50. |
| :DISPlay:WAVE:BRIGhtness? | Query returns 50. |

## :DISPlay:TRANsparent

- **Command format:**
  :DISPlay:TRANsparent { {1|ON} | {0|OFF} }
  :DISPlay:TRANsparent?
- **Functional description:**
  This command is used to turn on/off display transparency.
- **Return format:**
  Query returns 1 or 0, it represents ON or OFF.
- **For example:**
  :DISPlay:TRANsparent ON
  Turn on display transparency.
  :DISPlay:TRANsparent?
  Query returns 1, it represents display transparency is turned on.

## :DISPlay:CLEar

- **Command format:**
  :DISPlay:CLEar
- **Functional description:**
  This command is used to clear and refresh the waveform on the screen. If there is reference waveform, then clear and refresh the reference waveform.

## :DISPlay:TYPE

- **Command format:**
  :DISPlay:TYPE {XY12|XY13|XY14| XY23|XY24|XY34|YT}
  :DISPlay:TYPE?
- **Functional description:**
  This command is used to set timebase display type XY12 (X-Y mode: display the amplitude value of CH1 on the horizontal axis, display the amplitude value of CH2 on the vertical axis). XY34 (X-Y mode: display the amplitude value of CH3 on the horizontal axis, display the amplitude value of CH4 on the vertical axis). YT (Y-T mode: display the relative relation of vertical voltage and horizontal time).
- **Return format:**
  Query returns {XY12|XY13|XY14| XY23|XY24|XY34|YT}.
- **For example:**

| | |
|---|---|
| :DISP:TYPE YT | Set timebase format to YT mode. |
| :DISP:TYPE? | Query returns YT. |

## WAVeform Command

This command is used to read the waveform data and relative parameter on the screen of the oscilloscope.

### :WAVeform:MODE

- **Command format:**
  :WAVeform:MODE {NORMal | RAW}
  :WAVeform:MODE?
- **Functional description:**
  NORMal: read the current waveform data, the count of waveform data is fixed count.
  RAW: read the waveform data from internal storage, the count of waveform data is related to storage depth.
  Notes: This instruction resets the start, cut-off and waveform point. Data in internal storage can be read only when the oscilloscope in stop status. This instruction is invalid in MATH channel.
- **Return format:**
  Query returns {NORMal | RAW}.
- **For example:**

| | |
|---|---|
| :WAVeform:MODE RAW | Set the read mode of waveform data to RAW. |
| :WAVeform:MODE? | Query returns RAW. |

### :WAVeform:FORMat

- **Command format:**
  :WAVeform:FORMat { WORD | BYTE | ASCII }
  :WAVeform:FORMat?
- **Functional description:**
  The default waveform data format is AD waveform point data.
  BYTE: return AD data, a waveform data take a byte (that is 8 bits).
  WORD: return AD data, a waveform data take two bytes (that is 16 bits), low 8 bits is valid, high 8 bits is 0.
  ASCII: returned waveform with the actual voltage value of each waveform point in scientific notation, and each voltage value is separated by commas. It is conform with Appendix 2：IEEE 488.2 Binary Data Format.
  Such as #412342.00000E+01,2.20000E+01, 2.30000E+01..........\n.
- **Return format:**
  Query returns { WORD | BYTE | ASCII }.
- **For example:**

| | |
|---|---|
| :WAVeform:FORMat BYTE | Return format of waveform AD data is single byte mode. |
| :WAVeform:FORMat? | Query returns BYT. |

### :WAVeform:STARt

- **Command format:**
  :WAVeform:STARt <start>
  :WAVeform:STARt?
- **Functional description:**
  This command is used to set or query the start position of waveform data reading, <start> is integer data type.
  NORMal: 1~1400.
  RAW: 1 to the maximum storage depth point.
- **Return format:**
  Query returns the start position.
- **For example:**

| | |
|---|---|
| :WAVeform:STARt 200 | Set the start position of waveform data reading to 200. |
| :WAVeform:STARt? | Query returns 200. |

## :WAVeform:STOP

- **Command format:**
  :WAVeform:STOP <stop>
  :WAVeform:STOP?
- **Functional description:**
  This command is used to set or query the cut-off position of waveform data reading, < stop> is integer data type.
  NORMal: < stop> range is 1~1400.
  RAW: < stop> range is 1 to the maximum storage depth point.
- **Return format:**
  Query returns the cut-off position.
- **For example:**

| | |
|---|---|
| :WAVeform:STOP 400 | Set the end point of waveform data reading to 400. |
| :WAVeform:STOP? | Query returns 400. |

## :WAVeform:SOURce

- **Command format:**
  :WAVeform:SOURce {CHANnel<n>| MATH}
  :WAVeform:SOURce?
- **Functional description:**
  This command is used to set the signal source of waveform data is to be queried. If this command is not sent, it means query the waveform data of the current channel.
- **Return format:**
  Query returns { CHANnel1 | CHANnel2| CHANnel3| CHANnel4 | MATH }.
- **For example:**
  :WAVeform:SOURce CHAN1
  Set the signal source of waveform data to be queried as CH1.
  :WAVeform:SOURce?
  Query returns CHANnel1.

## :WAVeform:POINts

- **Command format:**
  :WAVeform:POINts <points>
  :WAVeform:POINts?
- **Functional description:**
  This command is used to set the waveform point which need to returned, the default value is 0.
- **Return format:**
  Query returns the waveform point that need to be returned.
- **For example:**
  :WAVeform:POINts 120
  Set returned waveform point to 120.
  :WAVeform:POINts?
  Query returns 120.

## :WAVeform:DATA?

- **Command format:**
  :WAVeform:DATA?
- **Functional description:**
  This command is used to read the waveform data from the specified data source.
- **Return format:**
  WAVeform:POINts is the specified quantity of waveform data. Waveform data source is related to: WAVeform:SOURce. Data format is related to WAVeform:FORMat. Returned format is conform with

Appendix 2：IEEE 488.2 Binary Data Format.

- For example：
  - ◆ The data flow for obtaining screen waveform as the following

  :WAVeform:SOURce CHAN1
  Set singal source of query waveform data to be queried to CH1.
  :WAVeform:MODE NORMal
  Set to read waveform data display on the screen.
  :WAVeform:FORMat BYTE
  Return format of waveform data is AD single byte mode.
  :WAVeform:DATA?
  Obtain the waveform data.

  - ◆ The data flow for obtaining waveform data from internal memory, this flow is only valid in stop status.

  :WAVeform:SOURce CHAN1
  Set singal source of query waveform data to be queried to CH1.
  :WAVeform:MODE RAW
  Set to read waveform data from internal memory.
  :WAVeform:FORMat BYTE
  Return format of waveform data is AD single byte mode.
  :WAVeform:POINts 5000 Read waveform point from internal memory is 5000.
  Cycle read internal waveform data.
  {
  :WAVeform:DATA?
  Acquire a piece of waveform data from internal memory.
  :WAVeform:START? Start position of reading waveform data, -1 represents to the last point.
  }
  Explanation: when reading internal data in batch, the read back data of each time is only the data of an area in internal memory, and the waveform data between two adjacent pieces is continuous. Each piece of data conforms to Appendix 2：IEEE 488.2 Binary Data Format.

## :WAVeform:PREamble?

- Command format：
  :WAVeform:PREamble?
- Functional description：
  Query returns the waveform configuration parameter of the current system.
- Return format：
  Query return is is sparated by comma ",".
  Return data format: format, Type, Points, Count, Xinc, Xor, Xref, Yinc, Yor, Yref.
  Format: BYTE (0), WORD (1), ASCII (2).
  Type: NORMAL (0), PEAK (1), AVER (2), HRESolution(3).
  Points: Waveform data point that need to be returned.
  Count: it is average time in average sampling, it is 1 in other mode.
  Xinc: The time difference between in two point of waveform data source in X direction.
  Xor: The relative time of trigger point.
  Xref: X reference.
  Yinc: Unit of voltage in Y direction.
  Yor: Y direction relative to Zero position of YREF.
  Yref: Reference value in Y direction, channel zero level ADC value.
- For example：
  :WAVeform:PREamble? Return 1,0,0,1,8.000e-009,-6.000e-006,0,4.000e-002,10,128.

## :WAVeform:XINCrement?

- **Command format:**
  :WAVeform:XINCrement?
- **Functional description:**
  This command is used to query time interval between two adjacent points of the current selected channel source in X direction.
  Returned value is related to the current data reading mode.
  In NORMal mode, XINCrement=TimeScale/waveform point of time scale in horizontal poistion (50)
  In RAW mode, XINCrement=1/SampleRate
- **Return format:**
  Query returns number of timebase, the unit is s.
- **For example:**
  :WAV:XINC? Query returns 3.000e-003.

## :WAVeform:XORigin?

- **Command format:**
  :WAVeform:XORigin?
- **Functional description:**
  This command is used to query the start time of waveform data of the current selected channel source in X direction.
  Returned value is related to the current data reading mode, time of trigger point is zero, before the trigger point is negative.
  In NORMal mode, return start time of waveform data displayed on the screen: XORigin = -1*TimeScale*7
  In RAW mode, return start time of waveform data in internal memory, XORigin = -1*(SamplePoints/SampleRate)/2
- **Return format:**
  Query returns time value, the unit is s.
- **For example:**
  :WAV:XOR? Query returns 3.000e-002.

## :WAVeform:XREFerence?

- **Command format:**
  :WAVeform:XREFerence?
- **Functional description:**
  This command is used to query the reference time benchmark of waveform point of the current selected channel source in X direction, it's always zero
- **Return format:**
  Query reference time benchmark, query returns 0.
- **For example:**
  :WAV:XREF? Query returns 0.

## :WAVeform:YINCrement?

- **Command format:**
  :WAVeform:YINCrement?
- **Functional description:**
  This command is used to query the unit of voltage of current selected channel source in Y direction, unit is the same with the current amplitude unit.
  Returned value is related to the current data reading mode:
  YINCrement = VerticalScale/ waveform point of amplitude scale in vertical position (25)
- **Return format:**
  Query returns unit voltage value in Y direction.

- **For example:**
  :WAV:YINC? Query returns 2.000e000.

### :WAVeform:YORigin?

- **Command format:**
  :WAVeform:YORigin?
- **Functional description:**
  This command is used to query the current selected channel source in Y direction relative to vertical offset of the vertical reference position.
  Returned value is related to the current data reading mode, up is positive and down is negative with the reference as the base:
  YORigin = VerticalOffset/YINCrement
- **Return format:**
  Query returns the vertical offset, it is integer type.
- **For example:**
  :WAV:YOR? Query returns 1.0.

### :WAVeform:YREFerence?

- **Command format:**
  :WAVeform:YREFerence?
- **Functional description:**
  This command is used to query the vertical reference position of the current selected channel source in Y direction, and ADC value of channel zero level.
  Returned value is related to the current data reading mode: YREFerence fixed as 128.
- **Return format:**
  Query returns the reference position, it is integer type.
- **For example:**
  :WAV:YREF? Query returns 1.28.

## SBUS Command

This command is used to set RS232, SPI, I$^2$C bus decoding and trigger parameter.

### Basic Attribute

#### :SBUS:DISPlay

- **Command format:**
  :SBUS:DISPlay {{1|ON}|{0|OFF}}
  :SBUS:DISPlay?
- **Functional description:**
  This command is used to turn on/off bus decoding status of the oscilloscope.
- **Return format:**
  Query returns 1 or 0, it represents ON or OFF.
- **For example:**
  :SBUS:DISPlay ON Turn on bus decoding status to display decoding wavefrom.
  :SBUS:DISPlay? Query returns 1.

#### :SBUS:MODE

- **Command format:**
  :SBUS:MODE {RS232 | I$^2$C | SPI }

:SBUS:MODE?

- **Functional description:**
  This command is used to select bus decoding and trigger mode.
- **Return format:**
  Query returns { RS232 | $I^2C$ | SPI }.
- **For example:**

| | |
|---|---|
| :SBUS:MODE $I^2C$ | Select $I^2C$ bus decoding mode. |
| :SBUS:MODE? | Query returns $I^2C$. |

## :SBUS:BASE

- **Command format:**
  :SBUS:BASE {ASCII | BINary | HEX | DEC}
  :SBUS:BASE?
- **Functional description:**
  This command is used to set bus display format for the oscilloscope.
- **Return format:**
  Query returns {ASCII | BINary | HEX | DEC}.
- **For example:**

| | |
|---|---|
| :SBUS:BASE BIN | Set bus decoding to binary display format. |
| :SBUS:BASE? | Query returns BINary. |

## :SBUS:EVENt

- **Command format:**
  :SBUS:EVENt { {1|ON} | {0|OFF} }
  :SBUS:EVENt?
- **Functional description:**
  This command is used to turn on/off bus event of the oscilloscope.
- **Return format:**
  Query returns 1 or 0, it represents ON or OFF.
- **For example:**

| | |
|---|---|
| :SBUS:EVENt ON | Turn on bus event. |
| :SBUS:EVENt? | Query returns 1. |

## :SBUS:DATA?

- **Command format:**
  :SBUS:DATA?
- **Functional description:**
  This command is used read the data of decoding event table of the oscilloscope.
- **Return format:**
  Query returns the data of decoding event table, returned data is conform with Appendix 2: IEEE 488.2 Binary Data Format.
- **For example:**
  :SBUS:DATA? Query returns

```
#9000000089RS232,
TIME,DATA,CHECK,
-1us,0,0,
-890.5ns,1,0,
-403.4ns,0,0,
9.8ns,1,0,
531.7ns,0,0,
```

RS232 represents decoding type (it may be $I^2C$, SPI or CAN), the event table data in CSV format is followed behind. The specified format of the event table data is automatically adapted by different devices. The data are separated by commas and will automatically line wrap according to the decoding list, the data value is related to the setting system display.

## :SBUS:TRIGger:MODE

- **Command format:**
  :SBUS:TRIGger:MODE {RS232 | I2C | SPI }
  :SBUS:TRIGger:MODE?
- **Functional description:**
  This command is used to set bus trigger mode.
- **Return format:**
  Query returns {RS232 | I2C | SPI }。
- **For example:**

| | |
|---|---|
| :SBUS:TRIGger:MODE I2C | Set trigger mode to I2C |
| :SBUS:TRIGger:MODE? | Query returns I2C |

## :SBUS:VERTical:POSition

- **Command format:**
  :SBUS:VERTical:POSition <value>
  :SBUS:VERTical:POSition?
- **Functional description:**
  This command is used to set trigger vertical position value of the oscilloscope. Parameter is integer type, step is 6, range is [-160,160]. The center of screen is zero point, up is positive, down is negative.
- **Return format:**
  Query returns vertical positon value.
- **For example:**

| | |
|---|---|
| :SBUS:VERTical:POSition 10 | Open bus vertical position value is 10. |
| :SBUS:VERTical:POSition? | Query returns 1.0 |

## :SBUS:TRIGger:SWEep

- **Command format:**
  :SBUS:TRIGger:SWEep {AUTO|NORMal|SINGle}
  :SBUS:TRIGger:SWEep?
- **Functional description:**
  This command is used to select bus trigger sweep mode.
  AUTO: In no trigger condition, the internal will produce trigger signal to force trigger.
  NORMal: It will be generated only trigger condition is met.
  SINGle: It will be generated one time and stop when trigger condition is met.
- **Return format:**
  Query returns trigger sweep mode { AUTO|NORMal|SINGle }.
- **For example:**

| | |
|---|---|
| :SBUS:TRIGger:SWEep AUTO | Set bus trigger sweep mode to AUTO. |
| :SBUS:TRIGger:SWEep? | Query returns AUTO. |

## RS232

### :SBUS:RS232:BAUDrate

- **Command format:**
  :SBUS:RS232:BAUDrate <baudrate>
  :SBUS:RS232:BAUDrate?
- **Functional description:**
  This command is used to set RS232 decoding baud rate of the oscilloscope. Parameter is integer type, range is 120~5000000.
- **Return format:**
  Query returns baud rate.
- **For example:**

| | |
|---|---|
| :SBUS:RS232:BAUDrate 500 | Set RS232 baud rate to 500b/s. |
| :SBUS:RS232:BAUDrate? | Query returns 500. |

### :SBUS:RS232:BITorder

- **Command format:**
  :SBUS:RS232:BITorder {LSBFirst | MSBFirst}
  :SBUS:RS232:BITorder?
- **Functional description:**
  This command is used to set RS232 bus decoding bit order of the oscilloscope, which includes LSBFirst and MSBFirst.
- **Return format:**
  Query returns {LSBFirst | MSBFirst}.
- **For example:**

| | |
|---|---|
| :SBUS:RS232:BITorder LSBF | Set RS232 bit order to LSB. |
| :SBUS:RS232:BITorder? | Query returns LSBFirst. |

### :SBUS:RS232:SOURce

- **Command format:**
  :SBUS:RS232:SOURce { CHANnel1 | CHANnel2| CHANnel3| CHANnel4 }
  :SBUS:RS232:SOURce?
- **Functional description:**
  This command is used to set RS232 bus decoding source.
- **Return format:**
  Query returns { CHANnel1 | CHANnel2| CHANnel3| CHANnel4}.
- **For example:**

| | |
|---|---|
| :SBUS:RS232:SOURce CHANnel1 | Set CH1 as RS232 bus decoding source. |
| :SBUS:RS232:SOURce? | Query returns CHANnel1. |

### :SBUS:RS232:POLarity

- **Command format:**
  :SBUS:RS232:POLarity { POSitive | NEGative }
  :SBUS:RS232:POLarity?
- **Functional description:**
  This command is used to set RS232 bus decoding polarity of the oscilloscope, which includes POSitive and NEGative.
- **Return format:**

Query returns { POSitive | NEGative }.

- **For example:**

| | |
|---|---|
| :SBUS:RS232:POLarity POSitive | Set RS232 bus decoding polarity to POSitive. |
| :SBUS:RS232:POLarity? | Query returns POSitive. |

### :SBUS:RS232:PARity

- **Command format:**
  :SBUS:RS232:PARity {EVEN | ODD | NONE}
  :SBUS:RS232:PARity?
- **Functional description:**
  This command is used to set RS232 bus parity of the oscilloscope.
- **Return format:**
  Query returns {EVEN | ODD | NONE}.
- **For example:**

| | |
|---|---|
| :SBUS:RS232:PARity ODD | Set RS232 bus parity to ODD. |
| :SBUS:RS232:PARity? | Query returns ODD. |

### :SBUS:RS232:DATA:BIT

- **Command format:**
  :SBUS:RS232:DATA:BIT {5 | 6 | 7 | 8 }
  :SBUS:RS232:DATA:BIT?
- **Functional description:**
  This command is used to set RS232 bus data bit of the oscilloscope.
- **Return format:**
  Query returns {5 | 6 | 7 | 8 }.
- **For example:**

| | |
|---|---|
| :SBUS:RS232:DATA:BIT 6 | Set RS232 data bit to 6. |
| :SBUS:RS232:DATA:BIT? | Query returns 6. |

### :SBUS:RS232:STOP:BIT

- **Command format:**
  :SBUS:RS232:STOP:BIT {1 | 2 }
  :SBUS:RS232:STOP:BIT?
- **Functional description:**
  This command is used to set RS232 bus stop bit of the oscilloscope.
- **Return format:**
  Query returns {1 | 2 }.
- **For example:**

| | |
|---|---|
| :SBUS:RS232:STOP:BIT 6 | Set RS232 stop bit to 6. |
| :SBUS:RS232:STOP:BIT? | Query returns 6. |

### :SBUS:RS232:DATA

- **Command format:**
  :SBUS:RS232:DATA <value>
  :SBUS:RS232:DATA?
- **Functional description:**
  This command is used to set RS232 bus trigger data of the oscilloscope. Binary character data presented by parameter 0 or 1, its range is related to the value set by instruction :SBUS:RS232:DATA:BIT, which is [0~2^databit – 1].

- Return format:
  Query returns data with binary data format.
- For example:

| | |
|---|---|
| :SBUS:RS232:DATA "01111111" | Set Set RS232 bus data to 0x7F. |
| :SBUS:RS232:DATA? | Query returns 01111111. |

### :SBUS:RS232:QUALifier

- Command format:
  :SBUS:RS232:QUALifier {BEGFrame | ERRFrame | ECCError | DATA}
  :SBUS:RS232:QUALifier?
- Functional description:
  This command is used to set RS232 bus trigger condition of the oscilloscope.
- Return format:
  Query returns {BEGFrame|ERRFrame|ECCError|DATA}.
- For example:

| | |
|---|---|
| :SBUS:RS232:QUALifier ERRF | Set RS232 bus condition to ERRFrame. |
| :SBUS:RS232:QUALifier? | Query returns ERRFrame. |

## I$^2$C

### :SBUS:I$^2$C:CLOCk:SOURce

- Command format:
  :SBUS:I$^2$C:CLOCk:SOURce { CHANnel1 | CHANnel2| CHANnel3| CHANnel4 }
  :SBUS:I$^2$C:CLOCk:SOURce?
- Functional description:
  This command is used to set I2C bus clock source of the oscilloscope.
- Return format:
  Query returns { CHANnel1 | CHANnel2| CHANnel3| CHANnel4}.
- For example:

| | |
|---|---|
| :SBUS:I$^2$C:CLOCk:SOURce CHANnel1 | Set CH1 as I$^2$C bus clock source. |
| :SBUS:I$^2$C:CLOCk:SOURce? | Query returns CHANnel1. |

### :SBUS:I$^2$C:DATA:SOURce

- Command format:
  :SBUS:I$^2$C:DATA:SOURce { CHANnel1 | CHANnel2| CHANnel3| CHANnel4}
  :SBUS:I$^2$C:DATA:SOURce?
- Functional description:
  This command is used to set I$^2$C bus data source of the oscilloscope.
- Return format:
  Query returns { CHANnel1 | CHANnel2| CHANnel3| CHANnel4 }.
- For example:

| | |
|---|---|
| :SBUS:I$^2$C:DATA:SOURce CHANnel1 | Set CH1 as I$^2$C bus data source. |
| :SBUS:I$^2$C:DATA:SOURce? | Query returns CHANnel1. |

### :SBUS:I$^2$C:ASIZe

- Command format:
  :SBUS:I$^2$C:ASIZe {7 | 10}

:SBUS:I$^2$C:ASIZe?

- **Functional description:**
  This command is used to set I$^2$C bus address bit wide of the oscilloscope.
- **Return format:**
  Query returns {7 | 10}.
- **For example:**

| | |
|---|---|
| :SBUS:I$^2$C:ASIZe 7 | Set I$^2$C bus address bit wide to 7. |
| :SBUS:I$^2$C:ASIZe? | Query returns 7. |

## :SBUS:I$^2$C:ADIRection

- **Command format:**
  :SBUS:I$^2$C:ADIRection { READ | WRITE }
  :SBUS:I$^2$C:ADIRection?
- **Functional description:**
  This command is used to set I$^2$C bus address direction of the oscilloscope.
- **Return format:**
  Query returns { READ | WRITE }.
- **For example:**

| | |
|---|---|
| :SBUS:I$^2$C:ADIRection READ | Set I$^2$C bus address direction to READ. |
| :SBUS:I$^2$C:ADIRection? | Query returns READ. |

## :SBUS:I$^2$C:ADDRess

- **Command format:**
  :SBUS:I$^2$C:ADDRess <value>
  :SBUS:I$^2$C:ADDRess?
- **Functional description:**
  This command is used to set I$^2$C bus address of the oscilloscope. Binary character string data presented by parameter is 0 or 1, its rang is related to the value set by instruction :SBUS: I$^2$C:ASIZe, which is [0~2^addressbit - 1].
- **Return format:**
  Query returns address value with binary character string.
- **For example:**

| | |
|---|---|
| :SBUS:I$^2$C:ADDRess "01111111" | Set I$^2$C bus address to 0x7F. |
| :SBUS:I$^2$C:ADDRess? | Query returns 01111111. |

## :SBUS:I$^2$C:QUALifier

- **Command format:**
  :SBUS:I$^2$C:QUALifier {STARt | RESTart | STOP | LOSS | ADDRess | DATA | ADATA}
  :SBUS:I$^2$C:QUALifier?
- **Functional description:**
  This command is used to set I$^2$C bus trigger condition of the oscilloscope.
- **Return format:**
  Query returns {STARt | RESTart | STOP | LOSS | ADDRess | DATA | ADATA}.
- **For example:**

| | |
|---|---|
| :SBUS:I$^2$C:QUALifier STOP | Set I$^2$C bus trigger condition to STOP. |
| :SBUS:I$^2$C:QUALifier? | Query returns STOP. |

### :SBUS:I²C:DATA:LEN

- **Command format:**
  :SBUS:$I^2C$:DATA:LEN <length>
  :SBUS:$I^2C$:DATA:LEN?
- **Functional description:**
  This command is used to set $I^2C$ bus trigger data length of the oscilloscope. It can takes value from 1~5.
- **Return format:**
  Query returns $I^2C$ bus trigger data length of the oscilloscope, it is integer data.
- **For example:**

| | |
|---|---|
| :SBUS:$I^2C$:DATA:LEN 2 | Set $I^2C$ bus trigger data length to 2 bytes. |
| :SBUS:$I^2C$:DATA:LEN? | Query returns 2. |

### :SBUS:I²C:DATA

- **Command format:**
  :SBUS:$I^2C$:DATA <value>
  :SBUS:$I^2C$:DATA?
- **Functional description:**
  This command is used to set $I^2C$ bus data. Binary character string data presented by parameter is 0 or 1, its data range is 0x0~0xFFFFFFFFFFFFFFFF.
- **Return format:**
  Query returns data with binary character string.
- **For example:**

| | |
|---|---|
| :SBUS:$I^2C$:DATA "1111111111111111" | Set $I^2C$ bus data to 0xFFFFF. |
| :SBUS:$I^2C$:DATA? | Query returns 1.111111111111111. |

## SPI

### :SBUS:SPI:CS:SOURce

- **Command format:**
  :SBUS:SPI:CS:SOURce { CHANnel1 | CHANnel2| CHANnel3| CHANnel4 }
  :SBUS:SPI:CS:SOURce?
- **Functional description:**
  This command is used to set SPI bus chip selection source of the oscilloscope.
- **Return format:**
  Query returns { CHANnel1 | CHANnel2| CHANnel3| CHANnel4}.
- **For example:**

| | |
|---|---|
| :SBUS:SPI:CS:SOURce CHANnel1 | Set CH1 as SPI bus chip selection source. |
| :SBUS:SPI:CS:SOURce? | Query returns CHANnel1. |

### :SBUS:SPI:CLOCk:SOURce

- **Command format:**
  :SBUS:SPI:CLOCk:SOURce { CHANnel1 | CHANnel2| CHANnel3| CHANnel4}
  :SBUS:SPI:CLOCk:SOURce?
- **Functional description:**
  This command is used to set SPI bus clock source of the oscilloscope.
- **Return format:**
  Query returns { CHANnel1 | CHANnel2| CHANnel3| CHANnel4 }.
- **For example:**

| | |
|---|---|
| :SBUS:SPI:CLOCk:SOURce CHANnel1 | Set CH1 as SPI bus clock source. |
| :SBUS:SPI:CLOCk:SOURce? | Query returns CHANnel1. |

## :SBUS:SPI:MOSI:SOURce

- **Command format:**
  :SBUS:SPI:MOSI:SOURce { CHANnel1 | CHANnel2| CHANnel3| CHANnel4 }
  :SBUS:SPI:MOSI:SOURce?
- **Functional description:**
  This command is used to set SPI bus master input slaver output source of the osciolloscope.
- **Return format:**
  Query returns { CHANnel1 | CHANnel2| CHANnel3| CHANnel4 }.
- **For example:**
  :SBUS:SPI:MOSI:SOURce CHANnel1
  Set CH1 as SPI bus master input slaver output source.
  :SBUS:SPI:MOSI:SOURce?
  Query returns CHANnel1.

## :SBUS:SPI:BITorder

- **Command format:**
  :SBUS:SPI:BITorder {LSBFirst | MSBFirst}
  :SBUS:SPI:BITorder?
- **Functional description:**
  This command is used to set SPI bus decoding byte order of the oscilloscope, which includes LSBFirst and MSBFirst.
- **Return format:**
  Query returns {LSBFirst | MSBFirst}.
- **For example:**

| | |
|---|---|
| :SBUS:SPI:BITorder LSBF | Set SPI bus decoding byte order to LSB. |
| :SBUS:SPI:BITorder? | Query returns LSBFirst. |

## :SBUS:SPI:CS:POLarity

- **Command format:**
  :SBUS:SPI:CS:POLarity {NEGative | POSitive}
  :SBUS:SPI:CS:POLarity?
- **Functional description:**
  This command is used to set SPI bus chip selection polarity of the oscilloscope, which includes POSitive and NEGative.
- **Return format:**
  Query returns {NEGative | POSitive}
- **For example:**

| | |
|---|---|
| :SBUS:SPI:CS:POLarity POSitive | Set SPI bus chip selection polarity to POSitive. |
| :SBUS:SPI:CS:POLarity? | Query returns POSitive. |

## :SBUS:SPI:CLOCk:POLarity

- **Command format:**
  :SBUS:SPI:CLOCk:POLarity {NEGative | POSitive}
  :SBUS:SPI:CLOCk:POLarity?
- **Functional description:**
  This command is used to set SPI bus clock polarity of the oscilloscope, which includes POSitive and

NEGative.

- Return format:
  Query returns {NEGative | POSitive}.
- For example:

| | |
|---|---|
| :SBUS:SPI:CLOCk:POLarity POSitive | Set SPI bus clock polarity to POSitive. |
| :SBUS:SPI:CLOCk:POLarity? | Query returns POSitive. |

## :SBUS:SPI:MOSI:POLarity

- Command format:
  :SBUS:SPI:MOSI:POLarity {NEGative | POSitive}
  :SBUS:SPI:MOSI:POLarity?
- Functional description:
  This command is used to set the polarity of SPI bus master output slaver intput source of the osciolloscope, which includes POSitive and NEGative.
- Return format:
  Query returns {NEGative | POSitive}.
- For example:
  :SBUS:SPI:MOSI:POLarity POSitive
  Set the polarity of SPI bus master output slaver intput source to POSitive.
  :SBUS:SPI:MOSI:POLarity?
  Query returns POSitive.

## :SBUS:SPI:WIDTh

- Command format:
  :SBUS:SPI:WIDTh <width>
  :SBUS:SPI:WIDTh?
- Functional description:
  This command is used to set SPI bus data bit width of the osciolloscope.
  <width> is integer data, the range is 4~32.
- Return format:
  Query returns SPI bus data bit width.
- For example:

| | |
|---|---|
| :SBUS:SPI:WIDTh 4 | Set SPI bus data bit width to 4. |
| :SBUS:SPI:WIDTh? | Query returns 4. |

## :SBUS:SPI:FRAMelen

- Command format:
  :SBUS:SPI:FRAMelen <len>
  :SBUS:SPI:FRAMelen?
- Functional description:
  This command is used to set the frame length for SPI bus data of the osciolloscope.
  <len> is integer data, the range is 1~32. Data bit width* data frame length can not over 128 bits
- Return format:
  Query returns the frame length bus data.
- For example:

| | |
|---|---|
| :SBUS:SPI:FRAMelen 1 | Set the frame length of SPI bus data to 1. |
| :SBUS:SPI:FRAMelen? | Query returns 1. |

## :SBUS:SPI:DATA

- **Command format:**
  :SBUS:SPI:DATA <value>
  :SBUS:SPI:DATA?
- **Functional description:**
  This command is used to set SPI bus DATA of the oscilloscope. Binary character string data presented by parameter is 0, 1 or X. X represents uncertainty, its data range is 0x0~0xFFFFFFFFFFFFFFFF.
- **Return format:**
  Query returns data with binary character string.
- **For example:**

| | |
|---|---|
| :SBUS:SPI:DATA "X00X00X1" | Set SPI bus data to X00X00X1. |
| :SBUS:SPI:DATA? | Query returns X00X00X1. |

## :SBUS:SPI:QUALifier

- **Command format:**
  :SBUS:SPI:QUALifier {IDLE|IDLEDATA|CS|CSDATA}
  :SBUS:SPI:QUALifier?
- **Functional description:**
  This command is used to set SPI bus condition of the oscilloscope.
- **Return format:**
  Query returns {IDLE|IDLEDATA|CS|CSDATA}.
- **For example:**

| | |
|---|---|
| :SBUS:SPI:QUALifier IDLE | Set SPI bus condition to IDLE. |
| :SBUS:SPI:QUALifier? | Query returns IDLE. |

## :SBUS:SPI:TRIGger:TIMeout

- **Command format:**
  :SBUS:SPI:TRIGger:TIMeout <vlaue>
  :SBUS:SPI:TRIGger:TIMeout?
- **Functional description:**
  This command is used to set SPI bus trigger timeout of the oscilloscope, parameter is integer type. <vlaue> equals to n * 4ns and value not exceed range [100ns,1s]. n takes value from [25,25*10^8].
- **Return format:**
  Query returns trigger timeout value in scientific notation, the unit is s.
- **For example:**

| | |
|---|---|
| :SBUS:SPI:TRIGger:TIMeout 100ns | Set SPI bus trigger timeout value to 100ns. |
| :SBUS:SPI:TRIGger:TIMeout? | Query returns 1..000e-007. |

# 3. Explanation of Programming

This chapter is to describe troubleshooting in process of programming. If you meet any of the following problems, please handle them according to the related instructions.

## Programming Preparation

Programming preparation is only applicable for using Visual Studio and LabVIEW development tools to programming under Windows operating system.

Firstly, user need to confirm that whether NI -VISA libray is installed (it can be download from the website https://www.ni.com/en-ca/support/downloads/drivers/download.ni-visa.html).

In this manual, the default installment path is C:\Program Files\IVI Foundation\VISA.

Build communication with PC via USB or LAN interface of the instrument, use USB data line to connect USB DEVICE port on the rear panel of the instrument with USB port of PC, or use LAN data line to connect LAN port on the rear panel of the instrument with LAN port of PC.

# 4. VISA Programming Example

There are some examples in this section. Throught these examples, user can know how to use VISA, and it can be combined with the command of programming manual to realize the control of the instrument. With these examples, user can develop more applications.

## VC++ Example

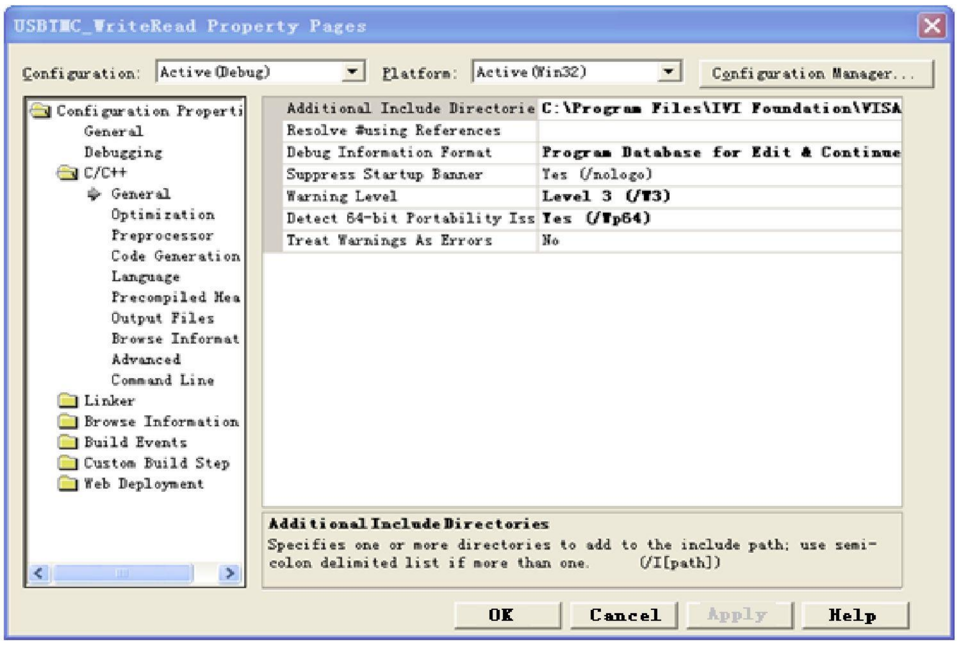
- Environment:Window system,Visual Studio
- Decription: Access the instrument via USBTMC and TCP/IP, and send "*IDN?" command on NI-VISA to query the device information.
- Steps

1. Open Visual Studio software to create a new VC++ win32 console project.
2. Set project environment that can adjust NI-VISA libray, which are static library and dynamic library.

a) Static library：
In NI-VISA installment path to find file visa.h, visatype.h and visa32.lib and copy them to the root path of VC++ project and add it to the project. Add two lines of code into file projectname.cpp as follows.

```
#include "visa.h"
#pragma comment(lib,"visa32.lib")
```

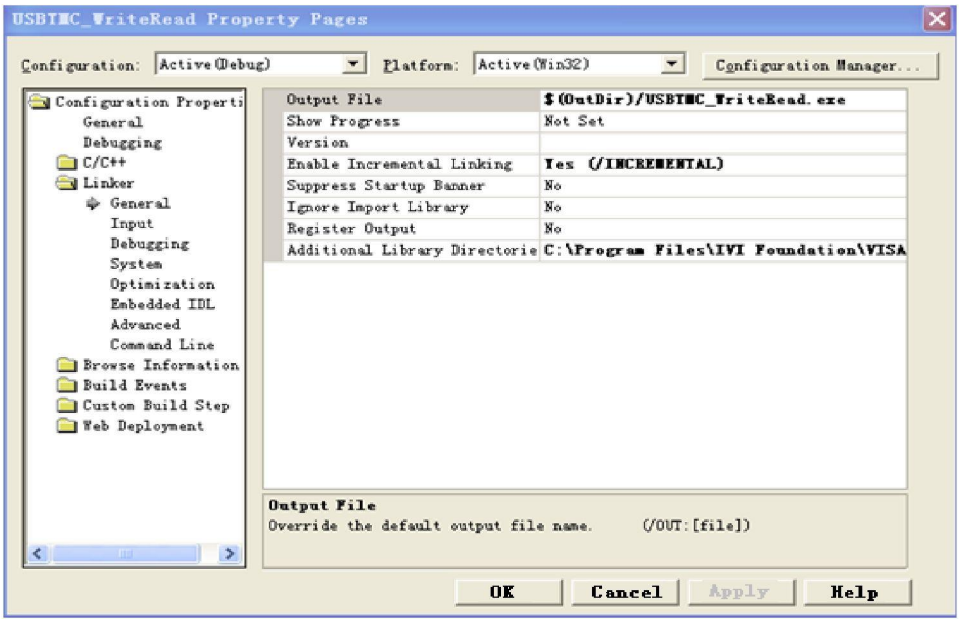
b) Dynamic library:
Press "project>>properties", select"c/c++---General" in attribute dialog on the leftside, set the value of "Additional Include Directories" as the installment path of NI-VIS (such as C:\ProgramFiles\IVI Foundation\VISA\WinNT\include), as shown in the following figure.



| | |
|---|---|
| Additional Include Directorie | C:\Program Files\IVI Foundation\VISA |
| Resolve #using References | |
| Debug Information Format | Program Database for Edit & Continue |
| Suppress Startup Banner | Yes (/nologo) |
| Warning Level | Level 3 (/W3) |
| Detect 64-bit Portability Iss | Yes (/Wp64) |
| Treat Warnings As Errors | No |

**Additional Include Directories**
Specifies one or more directories to add to the include path; use semi-colon delimited list if more than one. (/I[path])

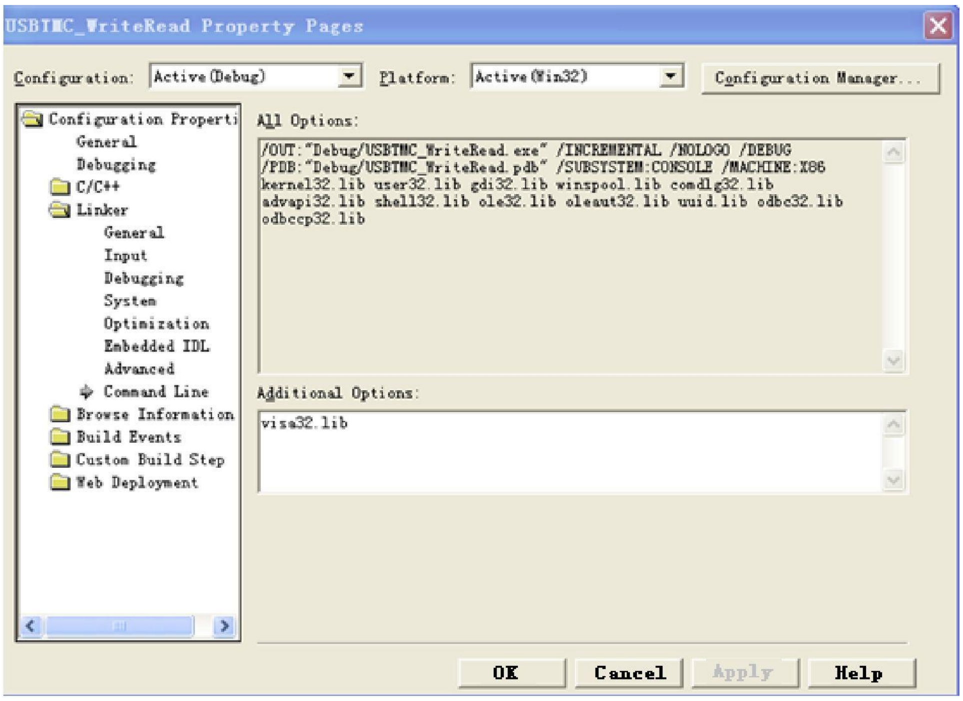
Select "Linker-General" in attribute dialog on the leftside, set the value of "Additional Library Directories" as the installment path of NI-VIS (such as C:\Program Files\IVI Foundation\VISA\WinNT\include), as shown in the following figure.



| | |
|---|---|
| Output File | $(OutDir)/USBTMC_WriteRead.exe |
| Show Progress | Not Set |
| Version | |
| Enable Incremental Linking | Yes (/INCREMENTAL) |
| Suppress Startup Banner | No |
| Ignore Import Library | No |
| Register Output | No |
| Additional Library Directorie | C:\Program Files\IVI Foundation\VISA |

**Output File**
Override the default output file name. (/OUT:[file])

Select “Linker-Command Line “in attribute dialog on the leftside, set the value of "Additional" as visa32.lib, as shown in the following figure.



USBTMC_WriteRead Property Pages

Configuration: Active(Debug)  Platform: Active(Win32)

All Options:

```
/OUT:"Debug/USBTMC_WriteRead.exe" /INCREMENTAL /NOLOGO /DEBUG
/PDB:"Debug/USBTMC_WriteRead.pdb" /SUBSYSTEM:CONSOLE /MACHINE:X86
kernel32.lib user32.lib gdi32.lib winspool.lib comdlg32.lib
advapi32.lib shell32.lib ole32.lib oleaut32.lib uuid.lib odbc32.lib
odbccp32.lib
```

Additional Options:

```
visa32.lib
```

Add file visa.h in projectname.cpp file

```
#include <visa.h>
```

**1.** Source Code

a) USBTMC Example

```
 int usbtmc_test()
 { /** This code demonstrates sending synchronous read & write commands
        * to an USB Test & Measurement Class (USBTMC) instrument using NI-VISA
        * The example writes the "*IDN?\n" string to all the USBTMC
        * devices connected to the system and attempts to read back
        * results using the write and read functions.
        * Open Resource Manager
        * Open VISA Session to an Instrument
        * Write the Identification Query Using viPrintf
        * Try to Read a Response With viScanf
        * Close the VISA Session*/
        ViSession defaultRM;
        ViSession instr;
        ViUInt32 numInstrs;
        ViFindList findList;
        ViStatus status;
        char instrResourceString[VI_FIND_BUFLEN];
        unsigned char buffer[100];
        int i;
        status = viOpenDefaultRM(&defaultRM);
        if (status < VI_SUCCESS)
        {
                printf("Could not open a session to the VISA Resource Manager!\n");
                return status;
        }
        /*Find all the USB TMC VISA resources in our system and store the number of resources in the system in
        numInstrs.*/
        status = viFindRsrc(defaultRM, "USB?*INSTR", &findList, &numInstrs, instrResourceString);
        if (status<VI_SUCCESS)
        {
                printf("An error occurred while finding resources. \nPress Enter to continue.");
                fflush(stdin);
                getchar();
```

```
		viClose(defaultRM);
		return status;
	}
	/** Now we will open VISA sessions to all USB TMC instruments.
	*	We must use the handle from viOpenDefaultRM and we must
	*	also use a string that indicates which instrument to open. This
	*	is called the instrument descriptor. The format for this string
	*	can be found in the function panel by right clicking on the
	*	descriptor parameter. After opening a session to the
	*	device, we will get a handle to the instrument which we
	*	will use in later VISA functions. The AccessMode and Timeout
	*	parameters in this function are reserved for future
	*	functionality. These two parameters are given the value VI_NULL. */
	for (i = 0; i < int(numInstrs); i++)
	{
		if (i > 0)
		{
			viFindNext(findList, instrResourceString);
		}
		status = viOpen(defaultRM, instrResourceString, VI_NULL, VI_NULL, &instr);
		if (status < VI_SUCCESS)
		{
			printf("Cannot open a session to the device %d. \n", i + 1);
			continue;
		}
		/** At this point we now have a session open to the USB TMC instrument.
		*We will now use the viPrintf function to send the device the string "*IDN?\n",
		*asking for the device's identification. */
		char * cmmand = "*IDN?\n";
		status = viPrintf(instr, cmmand);
		if (status < VI_SUCCESS)
		{
			printf("Error writing to the device %d. \n", i + 1);
			status = viClose(instr);
			continue;
		}
		/** Now we will attempt to read back a response from the device to
		*the identification query that was sent. We will use the viScanf
		*function to acquire the data.
		*After the data has been read the response is displayed. */
		status = viScanf(instr, "%t", buffer);
		if (status < VI_SUCCESS)
		{
			printf("Error reading a response from the device %d. \n", i + 1);
		}
		else
		{
			printf("\nDevice %d: %s\n", i + 1, buffer);
		}
		status = viClose(instr);
	}
	/*Now we will close the session to the instrument using viClose. This operation frees all
	system resources.*/
	status = viClose(defaultRM);
	printf("Press Enter to exit.");
	fflush(stdin);
	getchar();
	return 0;
}

int _tmain(int argc, _TCHAR* argv[ ])
{
	usbtmc_test();
	return 0;
```

```
}
```

b) TCP/IP Example

```
int tcp_ip_test(char *pIP)
{
    char outputBuffer[VI_FIND_BUFLEN];
    ViSession defaultRM, instr;
    ViStatus status;
    /* First we will need to open the default resource manager. */
    status = viOpenDefaultRM(&defaultRM);
    if (status < VI_SUCCESS)
    {
        printf("Could not open a session to the VISA Resource Manager!\n");
    }
    /* Now we will open a session via TCP/IP device */
    char head[256] = "TCPIP0::";
    char tail[] = "::inst0::INSTR";
    strcat(head, pIP);
    strcat(head, tail);
    status = viOpen(defaultRM, head, VI_LOAD_CONFIG, VI_NULL, &instr);
    if (status < VI_SUCCESS)
    {
        printf("An error occurred opening the session\n");
        viClose(defaultRM);
    }
    status = viPrintf(instr, "*idn?\n");
    status = viScanf(instr, "%t", outputBuffer);
    if (status < VI_SUCCESS)
    {
        printf("viRead failed with error code: %x \n", status);
        viClose(defaultRM);
    }
    else
    {
        printf("\nMesseage read from device: %*s\n", 0, outputBuffer);
    }
    status = viClose(instr);
    status = viClose(defaultRM);
    printf("Press Enter to exit.");
    fflush(stdin);
    getchar();
    return 0;
}
int _tmain(int argc, _TCHAR* argv[])
{
    printf("Please input IP address:");
    char ip[256];
    fflush(stdin);
    gets(ip);
    tcp_ip_test(ip);
    return 0;
}
```

## C#Example

- Environment:Window system, Visual Studio
- Description: Access the instrument via USBTMC and TCP/IP, and send "*IDN?" command on NI-VISA to query the device information.
- Steps:

1.Open Visual Studio software and create a new C# console project.
2.Add C# quote Ivi.Visa.dll and NationalInstruments.Visa.dll of VISA.
**3.Source Code**

a) USBTMC Example

```
class Program
{
        void usbtmc_test()
        {
            using (var rmSession = new ResourceManager())
            {
                var resources = rmSession.Find("USB?*INSTR");
                foreach (string s in resources)
                {
                    try
                    {
                        var mbSession = (MessageBasedSession)rmSession.Open(s);
                        mbSession.RawIO.Write("*IDN?\n");
                        System.Console.WriteLine(mbSession.RawIO.ReadString());
                    }
                    catch (Exception ex)
                    {
                        System.Console.WriteLine(ex.Message);
                    }
                }
            }
        }

        void Main(string[ ] args)
        {
            usbtmc_test();
        }
}
```

b) TCP/IP Example

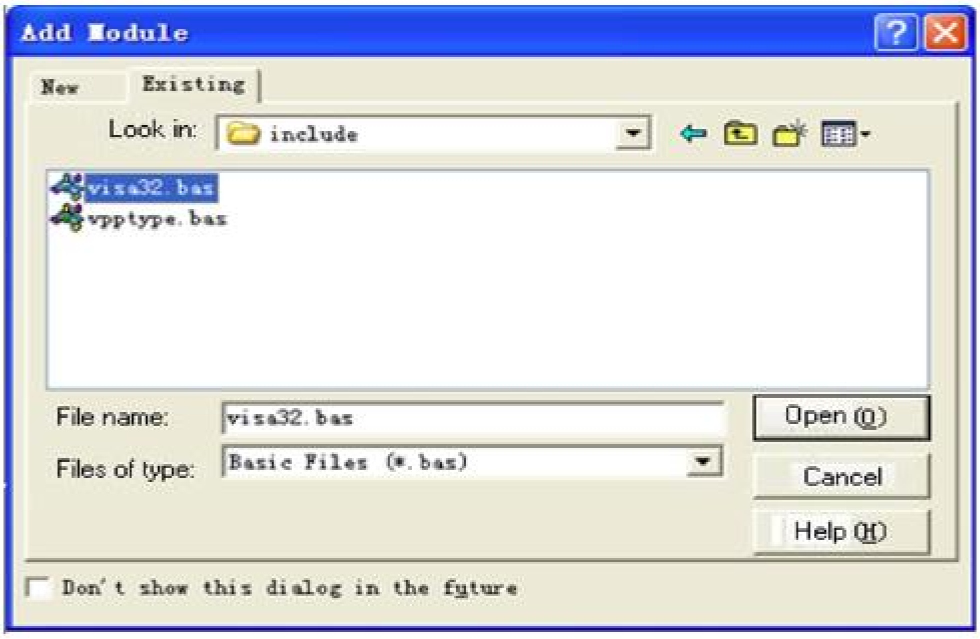
```
class Program
  {
        void tcp_ip_test(string ip)
        {
            using (var rmSession = new ResourceManager())
            {
                try
                {
                    var resource = string.Format("TCPIP0::{0}::inst0::INSTR", ip);
                    var mbSession = (MessageBasedSession)rmSession.Open(resource);
                    mbSession.RawIO.Write("*IDN?\n");
                    System.Console.WriteLine(mbSession.RawIO.ReadString());
                }
                catch (Exception ex)
                {
                    System.Console.WriteLine(ex.Message);
                }
            }
        }

        void Main(string[ ] args)
        {
            tcp_ip_test("192.168.20.11");
        }
  }
```

## VB Example

- Environment: Window systm, Microsoft Visual Basic 6.0.
- Description: Access the instrument via USBTMC and TCP/IP, and send "*IDN?" command on NI-VISA to query the device information.
- Steps

1. Open Visual Basic software and create a new standard application program project.

2. Set the project environment that can adjust NI-VISA libray, press Existing tab of Project>>Add Existing Item, in file "include" of NI-VISA installement path to find file visa32.bas and add this file, as shown in the following figure.



3. **Source Code**

a) USBTMC Example

```
PrivateFunction usbtmc_test() AsLong
' This code demonstrates sending synchronous read & write commands
' to an USB Test & Measurement Class (USBTMC) instrument using NI-VISA
' The example writes the "*IDN?\n" string to all the USBTMC
' devices connected to the system and attempts to read back
' results using the write and read functions.
' The general flow of the code is
' Open Resource Manager
' Open VISA Session to an Instrument
' Write the Identification Query Using viWrite
' Try to Read a Response With viRead
' Close the VISA Session

Const MAX_CNT = 200
Dim defaultRM AsLong
Dim instrsesn AsLong
Dim numInstrs AsLong
Dim findList AsLong
Dim retCount AsLong
Dim status AsLong
Dim instrResourceString AsString *VI_FIND_BUFLEN
Dim Buffer AsString * MAX_CNT
Dim i AsInteger

' First we must call viOpenDefaultRM to get the manager
' handle. We will store this handle in defaultRM.
status = viOpenDefaultRM(defaultRM)
If(status < VI_SUCCESS) Then
    resultTxt.Text = "Could not open a session to the VISA Resource Manager!"
    usbtmc_test = status
```

```
ExitFunction
EndIf

' Find all the USB TMC VISA resources in our system and store the
' number of resources in the system in numInstrs.
status = viFindRsrc(defaultRM, "USB?*INSTR", findList, numInstrs, instrResourceString)
If (status < VI_SUCCESS) Then
        resultTxt.Text = "An error occurred while finding resources."
        viClose(defaultRM)
        usbtmc_test = status
ExitFunction
EndIf

' Now we will open VISA sessions to all USB TMC instruments.
' We must use the handle from viOpenDefaultRM and we must
' also use a string that indicates which instrument to open.    This
' is called the instrument descriptor.    The format for this string
' can be found in the function panel by right clicking on the
' descriptor parameter. After opening a session to the
' device, we will get a handle to the instrument which we
' will use in later VISA functions.    The AccessMode and Timeout
' parameters in this function are reserved for future
' functionality.    These two parameters are given the value VI_NULL.
For i = 0 To numInstrs
If (i > 0) Then
        status = viFindNext(findList, instrResourceString)
EndIf
        status = viOpen(defaultRM, instrResourceString, VI_NULL, VI_NULL, instrsesn)
If (status < VI_SUCCESS) Then
        resultTxt.Text = "Cannot open a session to the device " + CStr(i + 1)
GoTo NextFind
EndIf

' At this point we now have a session open to the USB TMC instrument.
' We will now use the viWrite function to send the device the string "*IDN?",
' asking for the device's identification.
status = viWrite(instrsesn, "*IDN?", 5, retCount)
If (status < VI_SUCCESS) Then
        resultTxt.Text = "Error writing to the device."
        status = viClose(instrsesn)
GoTo NextFind
EndIf

' Now we will attempt to read back a response from the device to
' the identification query that was sent.    We will use the viRead
' function to acquire the data.
' After the data has been read the response is displayed.
status = viRead(instrsesn, Buffer, MAX_CNT, retCount)
If (status < VI_SUCCESS) Then
        resultTxt.Text = "Error reading a response from the device." + CStr(i + 1)
Else
        resultTxt.Text = "Read from device: " + CStr(i + 1) + " " + Buffer
EndIf
        status = viClose(instrsesn)
Next i

' Now we will close the session to the instrument using
' viClose. This operation frees all system resources.
status = viClose(defaultRM)
usbtmc_test = 0
EndFunction
```

b) TCP/IP Example

```
PrivateFunction tcp_ip_test(ByVal ip AsString) AsLong
Dim outputBuffer AsString * VI_FIND_BUFLEN
Dim defaultRM AsLong
Dim instrsesn AsLong
Dim status AsLong
Dim count AsLong

' First we will need to open the default resource manager.
status = viOpenDefaultRM(defaultRM)
If (status < VI_SUCCESS) Then
    resultTxt.Text = "Could not open a session to the VISA Resource Manager!"
    tcp_ip_test = status
ExitFunction
EndIf

' Now we will open a session via TCP/IP device
status = viOpen(defaultRM, "TCPIP0::" + ip + "::inst0::INSTR", VI_LOAD_CONFIG, VI_NULL, instrsesn)
If (status < VI_SUCCESS) Then
    resultTxt.Text = "An error occurred opening the session"
    viClose(defaultRM)
    tcp_ip_test = status
ExitFunction
EndIf
status = viWrite(instrsesn, "*IDN?", 5, count)
If (status < VI_SUCCESS) Then
    resultTxt.Text = "Error writing to the device."
EndIf
    status = viRead(instrsesn, outputBuffer, VI_FIND_BUFLEN, count)
If (status < VI_SUCCESS) Then
    resultTxt.Text = "Error reading a response from the device." + CStr(i + 1)
Else
    resultTxt.Text = "read from device:" + outputBuffer
EndIf
    status = viClose(instrsesn)
    status = viClose(defaultRM)
    tcp_ip_test = 0
EndFunction
```

## LabVIEW Example

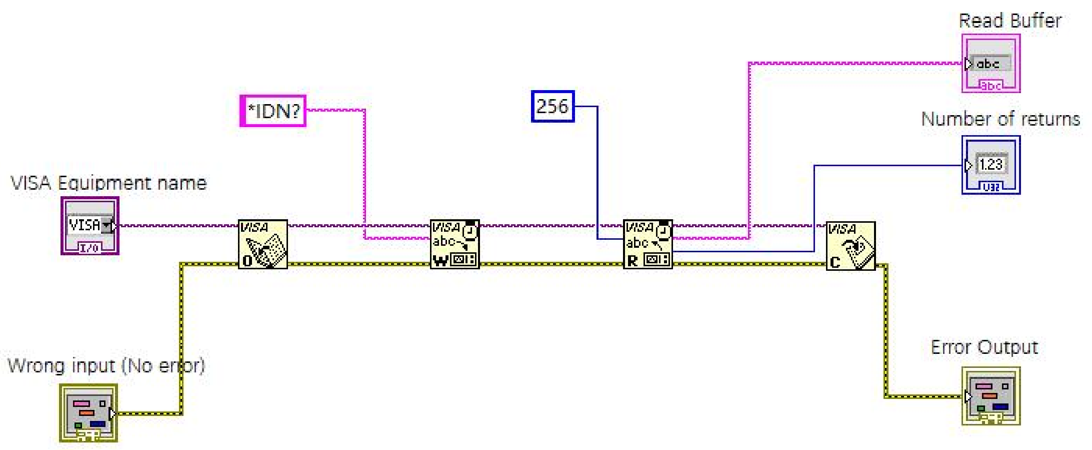
- Environment: Window system, LabVIEW
- Description: Access the instrument via USBTMC and TCP/IP, and send "*IDN?" command on NI-VISA to query the device information.
- Steps

1.Open LabVIEW software and create a VI file.

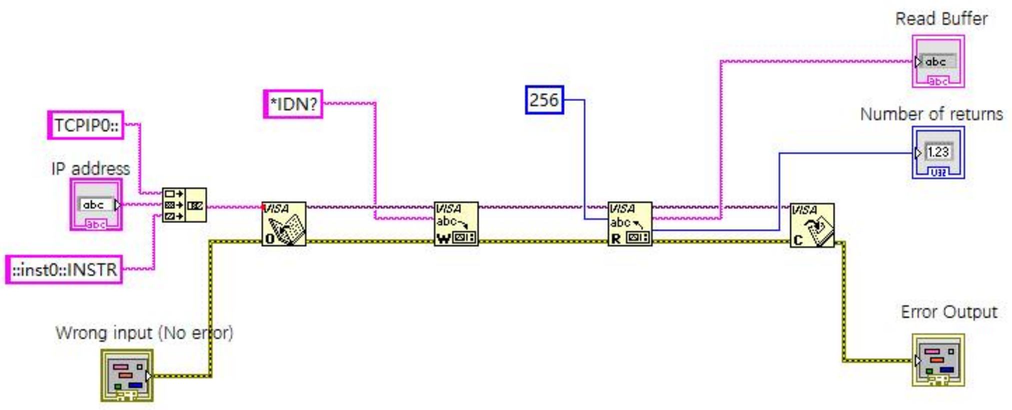
2.Add control, press the front panel interface, select and add VISA resource name, error input, error output and partial indetifier on control flow diagram.

3.Open diagram, press VISA resource name and then select and add function VISA Write, VISA Read, VISA Open and VISA Close on pop-out menu.

4.VI open a VISA session of USBTMC device and wrote *IDN? command and read back the response value. When all communication is complete, VI will close the VISA session.



5.Communication with the device via TCP/IP is similar with USBTMC, it need to set VISA write and read function to synchronous I/O, set LabVIEW to asynchronous IO by default. Right click on the node and select "Synchronous I/O Mode>>Synchronous" from shortcut menu to enable synchronous writing or reading of data, as shown in the following figure.



## MATLAB Example

- Environment: Window system, MATLAB
- Description: Access the instrument via USBTMC and TCP/IP, and send "*IDN?" command on NI-VISA to query the device information.
- Steps:

1. Open MATLAB software, click File>>New>>Script on Matlab interface to create an empty M file.
2. Source Code

a) USBTMC Example

```
function usbtmc_test()
% This code demonstrates sending synchronous read & write commands
% to an USB Test & Measurement Class (USBTMC) instrument using
% NI-VISA

%Create a VISA-USB object connected to a USB instrument
vu = visa('ni','USB0::0x5345::0x1234::SN20220718::INSTR');

%Open the VISA object created
fopen(vu);

%Send the string "*IDN?",asking for the device's identification.
fprintf(vu,'*IDN?');

%Request the data

outputbuffer = fscanf(vu);
disp(outputbuffer);

%Close the VISA object
fclose(vu);
delete(vu);
clear vu;

end
```

b) TCP/IP Example

```
function tcp_ip_test()
% This code demonstrates sending synchronous read & write commands
% to an TCP/IP instrument using NI-VISA
%Create a VISA-TCPIP object connected to an instrument

%configured with IP address.
vt = visa('ni',['TCPIP0::','192.168.20.11','::inst0::INSTR']);

%Open the VISA object created

fopen(vt);

%Send the string "*IDN?",asking for the device's identification.
fprintf(vt,'*IDN?');

%Request the data
outputbuffer = fscanf(vt);
disp(outputbuffer);

%Close the VISA object
fclose(vt);
```

```
delete(vt);
clear vt;

end
```

## Python Example

- Environment: Window system, Python3.8, PyVISA 1.11.0
- Description: Access the instrument via USBTMC and TCP/IP, send "*IDN?" command on NI-VISA to query the device information.
- Steps:

1.Install python firs, and then turn on Python script compiling software, create an empty test.py file.

2.Use pip install PyVISA instruction to install PyVISA, if it cannot install, please refer to this link (https://pyvisa.readthedocs.io/en/latest/)

3.Source Code

a) USBTMC Example

```
import pyvisa
rm = pyvisa.ResourceManager()
rm.list_resources()
my_instrument = rm.open_resource('USB0::0x5345::0x1234::SN20220718::INSTR')
print(my_instrument.query('*IDN?'))
```

b) TCP/TP Example

```
import pyvisa
rm = pyvisa.ResourceManager()
rm.list_resources()
my_instrument = rm.open_resource('TCPIP0::192.168.20.11::inst0::INSTR')
print(my_instrument.query('*IDN?'))
```

# 5. Programming Application Example

## Set bandwidth limit

When observing the low-frequency signal, it is necessary to reduce the high-frequency noise in the signal, then attenuate the high-frequency signal above 20 MHz in the signal. It can set bandwidth limit by the following command, such as set CH1,

```
CHANnel1:BWLimit ON
# Turn on bandwidth limit of CH1.
CHANnel1:BWLimit?
# Query returns 1, it represents bandwidth limit of CH1 is turned on.
```

## Set offset voltage

Set offset voltage of channel, it can set by the following command, such as set offset voltage of CH1,

```
CHANnel1:OFFSet 1 V    # CH1 move up 1V, so the offset voltage is 1 V.
CHANnel1:OFFSet?       # Query offset voltage of channel.
```

## Set volts/div scale

Set volts/div scale of channel, it can set by the following command, such as set volts/div scale of CH1,

```
CHANnel1:SCALe    500 mV    # Set volts/div scale of CH1 to 500mV.
CHANnel1:SCALe?             # Query volts/div scale value of CH1.
```

## Set timebase scale

Set timebase scale of the oscilloscope, it can set by the following command.

```
TIMebase:SCALe 0.005       # Set timebase scale of the oscilloscope to 5 ms.
TIMebase:SCALe?            # Query timebase scale of the oscilloscope.
```

## Query amplitude value

User can run the following command to query the amplitude measurement results without opening the measurement window, such as query amplitude value of CH1 waveform,

```
MEASure:VPP? CHANnel1       # Query amplitude value of CH1 waveform.
```

## Query time value of rising delay

User can run the following command to query measuring time of rising delay without opening the measurement window, such as query rising delay value of CH1 and channel 2,

```
MEASure:PDELay? CHANnel1,CHANnel2        # Query rising delay value of CH1 and channel 2.
```

# 6. Appendix 1： Key List

| Key | Functional Description | LED |
|---|---|---|
| CH1 | CH1 switch | √ |
| CH2 | CH2 switch | √ |
| CH3 | CH3 switch | √ |
| CH4 | CH4 switch | √ |
| MATH | Mathematical operation and menu | √ |
| AUTO | The control values of the oscilloscope are automatically set to condign display the waveform for observation. | |
| RS | Control the oscilloscope's running status, continuous send this command, the oscilloscope can switch to stop or run. | √ |
| SINGle | Single trigger | √ |
| TMENu | Trigger menu | |
| HMENu | Horizontal system menu | |
| MENu | Menu display switch | |
| F1 | Select the first menu item of the current menu | |
| F2 | Select the second menu item of the current menu | |
| F3 | Select the third menu item of the current menu | |
| F4 | Select the fourth menu item of the current menu | |
| F5 | Select the fifth menu item of the current menu | |
| F6 | Select the sixth menu item of the current menu | |
| PGDN | Next page | |
| MEASure | Meaurement function | |
| CURSor | Cursor measurement function and menu | √ |
| ACQuire | Sampling menu | |
| DISPlay | Storage menu | |
| STORage | System auxiliary menu | |
| UTILity | Meaurement function | |
| HOMe | Main function menu | |
| BUS | Bus menu | √ |
| GEN | Waveform generation menu | √ |
| NAVigate | Navigation menu | |
| DEFault | Restore to defualt setting | |
| PSCReen | One-key print or one-key save screenshot | |
| REF | Reference waveform menu | √ |
| DIGital | Digital channel menu | √ |
| LEFT | Left function key | √ |
| STOP | Stop menu | √ |
| RIGHt | Right function key | √ |

| | | |
|---|---|---|
| TFORe | Force trigger | |
| MODE | Mode switch | |
| FKNob | Multipurpose key | |
| FKNLeft | Multipurpose left rotary knob | |
| FKNRight | Multipurpose right rotary knob | |
| VPKNob | Vertical rotary knob | |
| VPKNLeft | Vertical left rotary knob | |
| VPKNRight | Vertical right rotary knob | |
| HPKNob | Horizontal rotary knob | |
| HPKNLeft | Horizontal left rotary knob | |
| HPKNRight | Horizontal right rotary knob | |
| TPKNob | Trigger rotary knob | |
| TPKNLeft | Trigger left rotary knob | |
| TPKNRight | Trigger right right knob | |
| VBKNob | Voltage reference rotary knob | |
| VBKNLeft | Voltage reference left rotary knob | |
| VBKNRight | Voltage reference right right knob | |
| TBKNob | Timebase rotary knob | |
| TBKNLeft | Timebase left rotary knob | |
| TBKNRight | Timebase right right knob | |
| ASTAtus | Automatic status indicator, no key, only has LED | √ |
| NSTAtus | Normal status indicator, no key, only has LED | √ |
| SSTAtus | Single status indicator, no key, only has LED | √ |
| FSTAtus | Multipurpose rotary knob status indicator, no key, only has LED | √ |

## 7. Appendix 2：IEEE 488.2 Binary Data System

DATA is data flow, other is ASCII character, as shown in the following figure<#812345678 + DATA + \n>

| Start（1Byte） | Length Bit Width（1Byte） | Total Data Length（Bit Width Byte） | DATA（n Byte） | End（1Byte） |
|---|---|---|---|---|
| # | x | x x x x x x x x | ………………… | \n |

## 8. Appendix 3：Cursor Measurement XY Mode List

| Bit Order | Name |
|---|---|
| 1 | Cursor Ax |
| 2 | Cursor Bx |
| 3 | Bx-Ax difference |
| 4 | Ay of CH1 |
| 5 | By of CH1 |
| 6 | By-Ay difference of CH1 |
| 7 | Ay of CH2 |
| 8 | By of CH2 |
| 9 | By-Ay difference of CH2 |
| 10 | Ay’s radius of two channels |
| 11 | By’s adius of two channels |
| 12 | By-Ay’s adius of two channels |
| 13 | Ay’s angle of two channels |
| 14 | By angle of two channels |
| 15 | By-Ay’s angle of two channels |
| 16 | The product of Ay of CH2 and Ay of CH1 |
| 17 | The product of By of CH2 and By of CH1 |
| 18 | The product of By-Ay of CH2 and By-Ay of CH1 |
| 19 | The ratio of the Ay of CH2 to the Ay of CH1 |
| 20 | The ratio of the By of CH2 and the By of CH1 |
| 21 | The ratio of the By-Ay of CH2 and the By-Ay of CH1 |