UNI-T

Programming Manual

UPO3000E&UPO2000E Series

Programmable Digital Oscilloscope

Warranty and Statement

Copyright

Brand Information

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

Software Version

00.00.01

Software upgrade may have some change and add more function, please subscribe **UNI-T** website to get the new version or contact **UNI-T.**

Statement

- UNI-T products are protected by patents (including obtained and pending) in China and other countries and regions.
- UNI-T reserves the right to change specifications and prices.
- The information provided in this manual supersedes all previous publications.
- Information provided in this manual is subject to change without prior notice.
- UNI-T shall not be liable for any errors that may be contained in this manual. For any incidental or consequential damages, arising out of the use or the information and deductive functions provided in this manual.
- Without the written permission of UNI-T, this manual cannot photocopied, reproduced or adapted.

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, you can contact us,

Website：https://www.uni-trend.com

SCPI

SCPI was defined as an additional layer on top of the IEEE 488.2-1987 specification "Standard Codes, Formats, Protocols, and Common Commands".The standard specifies a common syntax, command structure, and data formats, to be used with all instruments. It introduced generic commands (such as CONFigure and MEASure) that could be used with any instrument. These commands are grouped into subsystems. SCPI also defines several classes of instruments. For example, any controllable power supply would implement the same DCPSUPPLY base functionality class. Instrument classes specify which subsystems they implement, as well as any instrument-specific features.

The physical hardware communications link is not defined by SCPI. While it was originally created for the IEEE-488.1 (GPIB) bus, SCPI can also be used with RS-232, RS-422, Ethernet, USB, VXIbus, HiSLIP, etc.

SCPI commands are ASCII textual strings, which are send to the instrument over the physical layer (e.g., IEEE-488.1). Commands are a series of one or more keywords, many of which take parameters. In the specification, keywords are written CONFigure: The entire keyword can be used, or it can be abbreviated to just the uppercase portion. Responses to Query commands are typically ASCII strings. However, for bulk data, binary formats can be used.

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.

  Such as DISPlay:GRID:MODE { FULL | GRID | CROSS | NONE} command

- **Square Brackets [ ]** the contents in square brackets (command keywords) can omissible. If the parameter is ignoret, the instrument will set the parameter as the default value.

  Such as MEASure:NDUTy? [<source>] command, it presents 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, CHANnell: 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 identify 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 presents by the scientific notation method. 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 #3123xxxxxxxxxxxxxxxxxx\n presents 123 strings batch data return format, '3' presents "123" occupies three character bits.

SCPI Command

IEEE488.2 Common Command

*IDN?

- Command format:
  *IDN?
- Functional description:
  For query 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, UPO2000CS, UPO1000, 00.00.01

*RST

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

SYSTem Command

This command is for oscilloscope basic operation, including operating control, lock full qwerty, error queue and system data.

:RUN

- Command format:
  :RUN
- Functional description:
  Start to sampling, execute :STOP command to stop it.

:STOP

- Command format:
  :STOP
- Functional description:
  Stop sampling, execute :RUN command to restart it.

:AUTO

- Command format:

  :AUTO

- Functional description:

  Set the instrument control value automatically to display waveform to the best effect.

:SYSTem:LOCK

- Command format:

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

  :SYSTem:LOCK?

- Functional description:

  For lock/unlock full qwerty.

- Return format:

  Query return lock full qwerty status，0 presents unlock, 1 presents lock.

- For example:

:SYSTem:LOCK ON/:SYST:LOCK 1	lock full qwerty
:SYSTem:LOCK OFF/:SYST:LOCK 0	unlock full qwerty
:SYSTem:LOCK?	query return 1，it presents lock

:SYSTem:ERRor

- Command format:

  :SYSTem:ERRor

  :SYSTem:ERRor?

- Functional description:

  Empty error message queue.

- Return format:

  Query return the last error message, like Undefined header.

- For example:

:SYSTem:ERR	empty error message queue
:SYSTem:ERR?	query return Undefined header

:SYSTem:SETup

- Command format:

  :SYSTem:SETup <setup_data>

  :SYSTem:SETup?

- Functional description:

  For system configuration data, <setup_data> is conform to binary data IEEE 488.2 # format.

- Return format:

  Query return setup data, return data conform to binary data IEEE 488.2 # format.

:SYSTem:LANGuage

- Command format:

  :SYSTem:LANGuage { ENGLish | SIMPlifiedchinese | TRADitionalchinese }

  :SYSTem:LANGuage?
- Functional description:

  Set system language.
- Return format:

  Query return{ ENGLish | SIMPlifiedchinese | TRADitionalchinese }。
- For example:

  :SYSTem:LANGuage ENGL set system language as English

  :SYSTem:LANGuage? query return ENGLish

:SYSTem:RTC

- Command format:

  :SYSTem:RTC <year>,<month>,<day>,<hour>,<minute>,<second>

  :SYSTem:RTC?
- Functional description:

  Set system time.
- Return format:

  Query return year, month, date, hour, minute and second.
- For example:

  :SYSTem:RTC 2017,7,7,20,8,8 set system time at 20:08:08 July 7,2017

  :SYSTem:RTC? query return 2017,7,7,20,8,8

:SYSTem:CAL

- Command format:

  :SYSTem:CAL
- Functional description:

  Set system self-calibration, the instrument can not communicate when in self-calibration.

:SYSTem:CLEAr

- Command format:

  :SYSTem:CLEAr
- Functional description:

  Empty system storage waveform and setup data.

:SYSTem:CYMOmeter

- Command format:

  :SYSTem:CYMOmeter {1 | ON} | {0 | OFF}

  :SYSTem:CYMOmeter?

- Functional description:

  The switch of frequency meter.

- Return format:

  Query return frequency meter status， 1 presents on, 0 presents off.

- For example:

  :SYSTem:CYMOmeter ON　　turn on frequency meter

  :SYSTem:CYMOmeter?　　query return1

:SYSTem:SQUare:SELect

- Command format:

  :SYSTem:SQUare:SELect　{ 10Hz | 100Hz | 1KHz | 10KHz }

  :SYSTem:SQUare:SELect?

- Functional description:

  Select square wave output.

- Return format:

  Query return{ 10Hz | 100Hz | 1KHz | 10KHz }.

- For example:

  :SYSTem:SQUare:SELect 10Hz　　select 10Hz square wave output

  :SYSTem:SQUare:SELect?　　query return 10Hz

:SYSTem:OUTPut:SELect

- Command format:

  :SYSTem:OUTPut:SELect　{ TRIGger | PASS_FAIL }

  :SYSTem:OUTPut:SELect?

- Functional description:

  Set output selection, TRIGger, PASS_FAIL（pass &fail）.

- Return format:

  Query return{ TRIGger | PASS_FAIL }.

- For example:

  :SYSTem:OUTPut:SELect TRIG　　select TRIG

  :SYSTem:OUTPut:SELect?　　query return TRIG

:SYSTem:MNUDisplay

- Command format:

  :SYSTem:MNUDisplay { 1S | 2S | 5S | 10S | 20S | INFinite}

  :SYSTem:MNUDisplay?

- Functional description:

  Set menu display time，INFinite present menu stay on.

- Return format:

  Query return{ 1S | 2S | 5S | 10S | 20S | INFinite }.

- For example:

  :SYSTem:MNUDisplay 1S set menu display time as 1s, fold menu fold automatically

  :SYSTem:MNUDisplay? query return 1S

:SYSTem:BRIGhtness

- Command format:

  :SYSTem:BRIGhtness <count>

  :SYSTem:BRIGhtness?

- Functional description:

  Set screen luminance, <count> take value from 1~100，the number greater the screen brighter .

- Return format:

  Query return the current screen luminance.

- For example:

  :SYSTem:BRIGhtness 50 set screen luminance as 50

  :SYSTem:BRIGhtness? query return 50

:SYSTem:VERSion?

- Command format:

  :SYSTem:VERSion?

- Return format:

  Query return version information, character string information of 128 bytes.

  HW is hardware version number, SW is software version number，PD is creation date，ICV is protocol version number.

- For example:

  :SYST:VERS? query returnHW: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:

  Set the current internet parameter to take effect immediately.

:SYSTem:COMMunicate:LAN:GATEway

- Command format:

  :SYSTem:COMMunicate:LAN:GATEway <gateway>

  :SYSTem:COMMunicate:LAN:GATEway?

- Functional description:

  Set the default gateway, <gateway> is belong to ASCII character string parameter，the format is xxx.xxx.xxx.xxx.

- Return format:

  Query return the default gateway.

- For example:

  :SYST:COMM:LAN:GATE "192.168.1.1" set the default gateway as 192.168.1.1

  :SYST:COMM:LAN:GATE? query return192.168.1.1

:SYSTem:COMMunicate:LAN:SMASK

- Command format:

  :SYSTem:COMMunicate:LAN:SMASK <submask>

  :SYSTem:COMMunicate:LAN:SMASK?

- Functional description:

  Set subnet mask, <submask> is belong to ASCII character string, the format is xxx.xxx.xxx.xxx.

- Return format:

  Query return subnet mask.

- For example:

  :SYST:COMM:LAN:SMASK "255.255.255.0" set subnet mask 255.255.255.0

  :SYST:COMM:LAN:SMASK? query return 255.255.255.0

:SYSTem:COMMunicate:LAN:IPADdress

- Command format:

  :SYSTem:COMMunicate:LAN:IPADdress <ip>

  :SYSTem:COMMunicate:LAN:IPADdress?
- Functional description:

  Set IP address, <ip> is belong to ASCII character string, the format is xxx.xxx.xxx.xxx.
- Return format:

  Query return IP IP address.
- For example:

  :SYST:COMM:LAN:IPAD "192.168.1.10" set IP address 192.168.1.10

  :SYST:COMM:LAN:IPAD? query return 192.168.1.10

:SYSTem:COMMunicate:LAN:DHCP

- Command format:

  :SYSTem:COMMunicate:LAN:DHCP {{1 | ON} | {0 | OFF}}

  :SYSTem:COMMunicate:LAN:DHCP?
- Functional description:

  To switch configuration mode（auto IP）and（manual IP）.
- Return format:

  Query return dynamic configuration mode, 0 presents（manual IP）, 1 presents（auto IP）.
- For example:

  :SYST:COMM:LAN:DHCP ON turn on IP dynamic configuration mode

  :SYST:COMM:LAN:DHCP? query return 1

:SYSTem:COMMunicate:LAN:MAC?

- Command format:

  :SYSTem:COMMunicate:LAN:MAC?
- Return format:

  Query return MAC physical address
- For example:

  :SYST:COMM:LAN:MAC? query return 00-2A-A0-AA-E0-56

KEY Command

This command is to control key and knob on oscilloscope panel.

:KEY:<key>

- Command format:

  :KEY:<key>

  :KEY:<key>:LOCK { {1 | ON} | {0 | OFF} }

  :KEY:<key>:LOCK?

  :KEY:<key>:LED?

- Functional description:

  Set key function and lock/unlock this key. <key> definition and description see **Appendix 1: <Key> List**.

- Return format:

  Query return lock status or LED status.

  Lock status: 0 presents unlock, 1 presents lock;

  LED status: 0 presents light off, 1 presents light on (green), 2 presents light on (red).

- For example:

:KEY:AUTO	set oscilloscope control value
:KEY:AUTO:LOCK ON/OFF	lock/unlock key
:KEY:AUTO:LOCK?	query return lock status of this key , 1 presents lock.
:KEY:AUTO:LED?	query return LED status, 0 presents light off

CHANnel Command

This command is set channel independently, <n> take value as 1/2/3/4/5/6/7/8/9, it presents {CH1/CH2/ CH3/ CH4/ MATH/ REF-A/ REF-B/ REF-C/ REF-D}.

:CHANnel<n>:BWLimit

- Command format:

  :CHANnel<n>:BWLimit {{1|ON}|{0|OFF}}

  :CHANnel<n>:BWLimit?
- Functional description:

  Set bandwidth limit function is ON（turn on bandwidth limit to 20MHz，to reduce display noise）or OFF（turn off bandwidth limit to achieve full display bandwidth）.
- Return format:

  Query return 1 or 0， it presents ON or OFF.
- For example:

  :CHAN1:BWL ON　　turn on channel 1 bandwidth limit.

  :CHAN1:BWL?　　query return 1， it presents channel 1 bandwidth limit is turn on

:CHANnel<n>:COUPling

- Command format:

  :CHANnel<n>:COUPling {DC|AC|GND}

  :CHANnel<n>:COUPling?
- Functional description:

  Set channel coupling mode, DC presents the AC and DC component of input signal; AC presents blocked the DC component of input signal; GND presents cut off input signal.
- Return format:

  Query AC, DC or GND.
- For example:

  :CHAN1:COUP DC　　set channel 1 coupling mode as DC

  :CHAN1:COUP?　　query return DC

:CHANnel<n>:DISPlay

- Command format:

  :CHANnel<n>:DISPlay { {1|ON} | {0|OFF} }

  :CHANnel<n>:DISPlay?

- Functional description:

  The switch setting of the specified channel.

- Return format:

  Query return 1 or 0，it presents ON or OFF.

- For example:

  :CHAN1:DISP ON　　turn on channel 1

  :CHAN1:DISP?　　query return 1，it presents channel 1 is turn on

:CHANnel<n>:INVert

- Command format:

  :CHANnel<n>:INVert { {1|ON} | {0|OFF} }

  :CHANnel<n>:INVert?

- Functional description:

  Turn on/off inversion display function, ON（turn on inversion display function）or OFF（waveform normal display）.

- Return format:

  Query return 1 or 0，it presents ON or OFF.

- For example:

  :CHAN1:INV OFF　　turn off channel 1 inversion display function

  :CHAN1:INV?

  Query return 0，it presents channel 1 inversion display function is turn off

:CHANnel<n>:OFFSet

- Command format:

  :CHANnel<n>:OFFSet <offset>

  :CHANnel<n>:OFFSet?

- Functional description:

  Set waveform offset on vertical position, <n> take value as 1/2/3/4/5, it presents {CH1/CH2/CH3/ CH4/ MATH }.

- Return format:

  Query return offset value by scientific notation method，unit is V.

- For example:

  :CHAN1:OFFS 20V　　set channel 1 vertical offset as 20V

  :CHAN1:OFFS?　　query return 2.000e001

:CHANnel<n>:PROBe

- Command format:

  :CHANnel<n>:PROBe { 0.001X | 0.01X | 0.1X | 1X | 10X | 100X | 1000X }

  :CHANnel<n>:PROBe?

- Functional description:

  Set the probe attenuation factor corresponding to the probe.

- Return format:

  Query return { 0.001X | 0.01X | 0.1X | 1X | 10X | 100X | 1000X }.

- For example:

  :CHAN1:PROB 10X set channel 1 probe attenuation factor as 10

  :CHAN1:PROB? query return 10X

:CHANnel<n>:SCALe

- Command format:

  :CHANnel<n>:SCALe {<scale> | UP | DOWN}

  :CHANnel<n>:SCALe?

- Functional description:

  Set voltage base in vertical position.

  <scale>: voltage base value;

  UP: plus one position based on the current position;

  DOWN : minus one position based on the current position.

- Return format:

  Query return the current voltage position value by scientific notation method, unit is V.

- For example:

  :CHAN1:SCAL 20V set channel 1 voltage base as 20V

  :CHAN1:SCAL? query return 2.000e001

  :CHAN1:SCAL UP plus one position based on 20V voltage base

~~**:CHANnel<n>:FILTer**~~

- ~~Command format:~~

  ~~:CHANnel<n>:FILTer { {1|ON} | {0|OFF} }~~

  ~~:CHANnel<n>:FILTer?~~

- ~~Functional description:~~

  ~~The switch setting of digital filter function.~~

- ~~Return format:~~

  ~~Query return 1 or 0, it presents ON or OFF.~~

- ~~For example:~~

  ~~:CHAN1:FILT OFF turn off channel 1 digital filter function~~

  ~~:CHAN1:FILT? Query return 0, it presents channel 1 digital filter function is turn off~~

:CHANnel<n>:UNITs

- Command format:

  :CHANnel<n>:UNITs {VOLTs|AMPeres|WATTs|UNKNown}

  :CHANnel<n>:UNITs?

- Functional description:

  Set channel unit as VOLTs, AMPeres, WATTs and UNKNown.

- Return format:

  Query return VOLTs, AMPeres, WATTs or UNKNown.

- For example:

  :CHAN1:UNIT VOLT　　set channel 1 unit as voltage

  :CHAN1:UNIT?　　query return VOLTs

:CHANnel<n>:VERNier

- Command format:

  :CHANnel<n>:VERNier { {1|ON} | {0|OFF} }

  :CHANnel<n>:VERNier?

- Functional description:

  Set position accommodation mode, ON presents Fine tuning，to further subdivides between coarse tuning settings to improve vertical resolution; OFF presents Coarse tuning， based on 1-2-5 system to set vertical sensitivity.

- Return format:

  Query return 1 or 0， it presents ON or OFF.

- For example:

  :CHAN1:VERN ON　　turn on channel 1 fine tuning function

  :CHAN1:VERN?　　query return 1

:CHANnel<n>:BIAS

- Command format:

  :CHANnel<n>:BIAS { {1|ON} | {0|OFF} }

  :CHANnel<n>:BIAS?

- Functional description:

  The switch setting of channel bias voltage.

- Return format:

  Query return 1 or 0， it presents ON or OFF.

- For example:

  :CHAN1:BIAS ON　　turn on channel 1 bias voltage function

  :CHAN1: BIAS?　　query return 1

:CHANnel<n>:BIASV

- Command format:

  :CHANnel<n>:BIASV <value>

  :CHANnel<n>:BIASV?

- Functional description:

  Set channel bias voltage value.

- Return format:

  Query return bias voltage value by scientific notation method，unit is V.

- For example:

  :CHAN1:BIASV 2　　set channel 1 bias voltage value as 2V

  :CHAN1:BIASV?　　query return 2.000e000

:CHANnel<n>:BIASV:ZREO

- Command format:

  :CHANnel<n>:BIASV:ZERO

- Functional description:

  Set channel bias voltage go to zero.

:CHANnel<n>:SELect

- Command format:

  :CHANnel<n>:SELect

  :CHANnel<n>:SELect?

- Functional description:

  For channel selection.

- Return format:

  Query return 1 or 0，it presents ON or OFF.

- For example:

  :CHAN1:SELect　　select channel 1

  :CHAN1:SELect?　　query return 1，it presents channel 1 has been selected

TIMebase Command

This command is to change the current channel horizontal scale (time base) and horizontal position of trigger in memory (trigger offset). Horizontal scale change will make waveform expand or shrink relative to screen center. Horizontal position change will make wave shift relative to screen center.

:TIMebase:MODE

- Command format:

  :TIMebase:MODE {MAIN | WINDow}

  :TIMebase:MODE?
- Functional description:

  Set timebase mode, MAIN or WINDow (zoom timebase <Zoomed>)
- Return format:

  Query return MAIN or WINDow.
- For example:

  :TIM:MODE MAIN　　set timebase mode as MAIN

  :TIM:MODE?　　query return MAIN

:TIMebase:OFFSet

- Command format:

  :TIMebase:OFFSet <offset>

  :TIMebase:OFFSet?
- Functional description:

  For adjust MAIN offset value, wave shift relative to screen center.
- Return format:

  Query return<offset> value by scientific notation， unit is s.
- For example:

  :TIM:OFFS 1s　　set MAIN offset value as 1s

  :TIM:OFFS?　　query return 1.000e000。

:TIMebase:WINDow:OFFSet

- Command format:

  :TIMebase:WINDow:OFFSet <offset>

  :TIMebase:WINDow:OFFSet?

- Functional description:

  For adjust WINDow offset value（zoom timebase <Zoomed>）,wave shift relative to screen center.

- Return format:

  Query return<offset> value by scientific notation method, unit is s.

- For example:

  :TIM:WIND:OFFS 1　　set WINDow offset value as 1s

  :TIM:WIND:OFFS?　　query return 1.000e000

:TIMebase:SCALe

- Command format:

  :TIMebase:SCALe {<scale> | UP | DOWN}

  :TIMebase:SCALe?

- Functional description:

  Set timebase position of MAIN, s/div.

  <scale>：timebase value；

  UP：plus one position based on the current position；

  DOWN ：minus one position based on the current position.

- Return format:

  Query return< scale> value by scientific notation method, unit is s/div.

- For example:

  :TIM:SCAL 2　　set MAIN offset value as 2s/div

  :TIM:SCAL?　　query return 2.000e000

:TIMebase:WINDow:SCALe

- Command format:

  :TIMebase:WINDow:SCALe < scale >

  :TIMebase:WINDow:SCALe?

- Functional description:

  For adjust WINDow timebase（zoom timebase <Zoomed>）, that is s/div.

- Return format:

  Query return < scale> value by scientific notation method, in s/div.

- For example:

  :TIM:WIND:SCAL 2　　set WINDow timebase offset value as 2s/div

  :TIM:WIND:SCAL?　　query return 2.000e000

:TIMebase:INDPendent

- Command format:
  :TIMebase:INDPendent { {1|ON} | {0|OFF} }
  :TIMebase:INDPendent?
- Functional description:
  The switch setting of independent mode of time base.
- Return format:
  Query return 1 or 0，it presents ON or OFF.
- For example:

:TIM:INDP ON	turn on independent mode of time base
:TIM:INDP?	query return 1

FUNCtion Command

Display the operational result plus, minus, multiply, divide, AND, OR, negation, exclusive OR and FFT of CH1,CH2,CH3,CH4 waveform, set filter to use expression operation.

:FUNCtion:MATH:MODE

- Command format:
  FUNCtion:MATH:MODE {MATH|FFT|LOGic|FILTer|ADVance}
  FUNCtion:MATH:MODE?
- Functional description:
  Select MATH function mode.
- Return format:
  Query return{MATH|FFT|LOGic|FILTer|ADVance}.
- For example:

FUNC:MATH:MODE FFT	select MATH mode as FFT mode
FUNC:MATH:MODE?	query return FFT

:FUNCtion:OPERation

- Command format:
  :FUNCtion:OPERation {ADD | SUBTract | MULTiply | DIVide | AND|OR|NOT|XOR}
  :FUNCtion:OPERation?
- Functional description:
  Set functional operator, including basic operation and logic operation, they are plus, minus, multiply, divide, AND, OR, negation, exclusive OR.
- Return format:
  Query return{ADD | SUBTract | MULTiply | DIVide | AND|OR|NOT|XOR}.

- For example:

  :FUNCtion:OPERation ADD — use plus operator: src1+src2

  :FUNCtion:OPERation? — query return ADD

:FUNCtion:DIGital<n>:THReshold

- Command format:

  :FUNCtion:DIGital<n>:THReshold <value>

  :FUNCtion:DIGital<n>:THReshold?

- Functional description:

  Set the logic threshold of specified channel, if great than the logic threshold take value as 1, less than the logic threshold take value as 0, n take value as 1, 2, 3, 4.

- Return format:

  Query return 1.000e000, unit is V.

- For example:

  :FUNC:DIG1:THR 1V — set the logic threshold value of channel 1 as 1V

  :FUNC:DIG1:THR? — query return 1.000e000

:FUNCtion:SOURce<m>

- Command format:

  :FUNCtion:SOURce<m> <value>

  :FUNCtion:SOURce<m>?

- Functional description:

  SOURce <m> presents source 1 or source 2, <m> take vaule as 1, 2.

  SOURce1 is to select the first source of operator and math function or it can be single source of Filter and FFT.

  SOURce2 is to select the second source of operator and math function. Single source of Filter, and FFT is not applicable.

  <value> presents CHANnel<n>, <n> take vaule as 1/2/3/4{CH1/ CH2/ CH3/ CH4}.

- Return format:

  Query return<n> take value as 1/2/3/4/5/6/7/8/9.

- For example:

  :FUNCtion:SOUR1 CHAN1 — choose channel 1 as source 1

  :FUNCtion:SOUR1? — query return 1

  :FUNCtion:SOUR2 CHAN2 — choose channel 2 as source 2

  :FUNCtion:SOUR2? — query return 2

  :FUNCtion:OPERation ADD — add source 1 and source 2

:FUNCtion:FFT:WINDow

- Command format:

  :FUNCtion:FFT:WINDow {RECTangular|HANNing|HAMMing|BMAN}

  :FUNCtion:FFT:WINDow?

- Functional description:

  FFT windowing to intercept signal. RECT, HANN, HAMM, BMAN is rectangular window, Hanning window, Hamming window and Blacman window respectively.

- Return format:

  Query return {RECTangular|HANNing|HAMMing|BMAN}.

- For example:

  :FUNCtion:SOUR1 CHAN1 set channel 1 as source 1

  :FUNC:FFT:WIND HAMM windowing Hamming window

  :FUNC:FFT:WIND? query return HAMMing

:FUNCtion:FFT:VTYPe

- Command format:

  :FUNCtion:FFT:VTYPe {VRMS|DBRMS}

  :FUNCtion:FFT:VTYPe?

- Functional description:

  Set unit of FFT vertical diection as dbrms or VRMS. dbrms presents root mean square of power; VRMS presents root mean square of voltage.

- Return format:

  Query return VRMS, DBRMS.

- For example:

  :FUNCtion:SOUR1 CHAN1 set channel 1 as source 1

  :FUNC:FFT:VTYP VRMS

  set unit of FFT vertical direction as root mean square of voltage

  :FUNC:FFT:VTYP? query return VRMS

:FUNCtion:FFT:FREQuency

- Command format:

  :FUNCtion:FFT:FREQuency?

- Functional description:

  The center frequency of spectrum and waveform after obtaining FFT.

- Return format:

  Query return 1.000e003，unit is Hz.

- For example:

:FUNCtion:SOUR1 CHAN1	set channel 1 as source 1
:FUNC:FFT:FREQ?	query return 1.000e003

:FUNCtion:FILTer:TYPE

- Command format:

  :FUNCtion:FILTer:TYPE {LP|HP|BP|BS}

  :FUNCtion:FILTer:TYPE?
- Functional description:

  Set filter mode, LP、HP、BP、BS presents low-pass filter, high-pass filter, band-pass filter and band-stop filer respectively.
- Return format:

  Query return LP, HP, BP, BS.
- For example:

:FUNCtion:SOUR1 CHAN1	set channel 1 as source 1
:FUNC:FILT:TYPE BP	set as band-pass filter
:FUNC:FILT:TYPE?	query return BP

:FUNCtion:FILTer:FREQuency:HIGH

- Command format:

  :FUNCtion:FILTer:FREQuency:HIGH < freq>

  :FUNCtion:FILTer:FREQuency:HIGH?
- Functional description:

  The setting of the upper frequency limit of filter. It is applicable for high-pass filter, band-pass filer and band-stop filter.
- Return format:

  Query return 1.000e003，unit is Hz.
- For example:

:FUNCtion:SOUR1 CHAN1	set channel 1 as source 1
:FUNC:FILT:FREQ:HIGH 1KHz	set the upper frequency limit of filter as 1kHz
:FUNC:FILT:FREQ:HIGH?	query return 1.000e003

:FUNCtion:FILTer:FREQuency:LOW

- Command format:

  :FUNCtion:FILTer:FREQuency:LOW <freq>

  :FUNCtion:FILTer:FREQuency:LOW?
- Functional description:

  The setting of the low frequency limit of filter. It is applicable for high-pass filter, band-pass filer and band-stop filter.

- Return format:

  Query return 6.000e001，unit is Hz.

- For example:

  :FUNC:SOUR1 CHAN1　　set channel 1 as source 1

  :FUNC:FILT:FREQ:LOW 60Hz　　set the low frequency limit of filter as 60Hz

  :FUNC:FILT:FREQ:LOW?　　query return 6.000e001

:FUNCtion:LOGic:INVert

- Command format:

  :FUNCtion:LOGic:INVert　{{1|ON}|{0|OFF}}

  :FUNCtion:LOGic:INVert?

- Functional description:

  The switch setting of logic inversion function.

- Return format:

  Query return 1 or 0，it presents ON or OFF.

- For example:

  :FUNCtion:LOGic:INVert OFF　　turn off inversion function

  :FUNCtion:LOGic:INVert?　　query return 0，turn off inversion function

:FUNCtion:EXPRession

- Command format:

  :FUNCtion:EXPRession　<expression>

- Functional description:

  Use free combination expression to mathematical calculation.

  Expression format see Advance item in MATH manu，<expression> is belong to ASCII character string parameter.

- For example:

  :FUNCtion:EXRP "CH1*CH2"　　channel 1 multiply channel 2

MEASure Command

This command is for oscilloscope basic measurement, return test result by scientific notation method.

:MEASure:ALL

- Command format:

  :MEASure:ALL {{1 | ON} | {0 | OFF}}

  :MEASure:ALL?

- Functional description:

  The switch setting of all measurement function.

- Return format:

  Query return whether all measurement function is turn on.

- For example:

:MEASure:ALL ON	turn on all measurement function
:MEASure:ALL?	query return1

:MEASure:CLEar

- Command format:

  :MEASure:CLEar

- Functional description:

  Empty the current measured parameter value.

- For example:

:MEAS:CLE	delete the current measured parameter value

:MEASure:SOURce

- Command format:

  :MEASure:SOURce <source>

  :MEASure:SOURce?

- Functional description:

  Select measurement source, <source> is CHANnel<n>，n take value as 1, 2, 3, 4.

- Return format:

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

- For example:

:MEAS:SOUR CHAN1	choose channel 1 as measurement source
:MEAS:SOUR?	return CHANnel1

:MEASure:PDUTy?

- Command format:

  :MEASure:PDUTy? [<source>]

- Functional description:

  Measuring positive duty ratio of the specified channel waveform, <source> take value as CHANnel1, CHANnel2, CHANnel3 or CHANnel4. Ignore means the current channel.

- Return format:

  Query return 5.000e001，unit is %.

:MEASure:NDUTy?

- Command format:

  :MEASure:NDUTy? [<source>]

- Functional description:

  Measuring negative duty ratio of the specified channel waveform, <source> take value as CHANnel1, CHANnel2, CHANnel3, and CHANnel4. Ignore means the current channel.

- Return format:

  Query return 5.000e001，unit is %。

:MEASure:PDELay?

- Command format:

  :MEASure:PDELay? [<source1>,<source2>]

- Functional description:

  Measuring <source1> and <source2> relative to time delay of rise edge, <source> take value as CHANnel1, CHANnel2, CHANnel3 and CHANnel4. Ignore means the current channel.

- Return format:

  Query return-1.000e-004，unit is s.

- For example:

  Measuring the relative to time delay of rise edge

  {

  :MEAS:DEL:SOUR CHAN1,CHAN2

  :MEASure:PDEL?

  }

  equivalent to

  {

  :MEASure:PDEL? CHAN1,CHAN2

  }

:MEASure:NDELay?

- Command format:

  :MEASure:NDELay? [<source1>,<source2>]

- Functional description:

  Measuring <source1> and <source2> relative to time delay of fall edge, <source> take value as CHANnel1, CHANnel2, CHANnel3 and CHANnel4.

- Return format:

  Query return-1.000e-004，unit is s.

- For example:

  Measuring the relative to time delay of fall edge

  {

  :MEAS:DEL:SOUR CHAN1,CHAN2

  :MEASure:NDEL?

  }

  equivalent to

  {

  :MEASure:NDEL? CHAN1,CHAN2

  }

:MEASure:PHASe?

- Command format:

  :MEASure:PHASe? [<source1>,<source2>]

- Functional description:

  Timing measuring <source1> relative to <source2> the advance or hysteretic of time quantum, it presents by degree, 360°as a period, source take value as CHANnel1, CHANnel2, CHANnel3, and CHANnel4.

- Return format:

  Query return 1.000e001，unit is degree.

- For example:

  Measuring <source1> relative to <source2> the advance or hysteretic of time quantum

  {

  :MEAS:PHAS:SOUR CHAN1,CHAN2

  :MEASure:PHAS?

  }

  equivalent to

  {

  :MEASure:PHAS? CHAN1,CHAN2

  }

:MEASure:VPP?

- Command format:

  :MEASure:VPP? [<source>]

- Functional description:

  Measuring peak-to-peak value of the specified channel waveform, <source> take value as CHANnel1, CHANnel2, CHANnel3 and CHANnel4. Ignore means the current channel.

- Return format:

  Query return 3.120e000，unit is V.

:MEASure:VMAX?

- Command format:

  :MEASure:VMAX? [<source>]

- Functional description:

  Measuring the maximum value of the specified channel waveform, <source> take value as CHANnel1, CHANnel2, CHANnel3 and CHANnel4. Ignore means the current channel.

- Return format:

  Query return2.120e000，unit is V.

:MEASure:VMIN?

- Command format:

  :MEASure:VMIN? [<source>]

- Functional description:

  Measuring the minimum value of the specified channel waveform, <source> take value as CHANnel1, CHANnel2, CHANnel3 and CHANnel4. Ignore means the current channel.

- Return format:

  Query return -2.120e000，unit is V.

:MEASure:VAMPlitude?

- Command format:

  :MEASure:VAMPlitude? [<source>]

- Functional description:

  Measuring the amplitude value of the specified channel waveform, <source> take value as CHANnel1, CHANnel2, CHANnel3 and CHANnel4. Ignore means the current channel.

- Return format:

  Query return3.120e000，unit is V.

:MEASure:VTOP?

- Command format:

  :MEASure:VTOP? [<source>]

- Functional description:

  Measuring the top value of the specified channel waveform, <source> take value as CHANnel1, CHANnel2, CHANnel3 and CHANnel4. Ignore means the current channel.

- Return format:

  Query return 3.120e000，unit is V.

:MEASure:VBASe?

- Command format:

  :MEASure:VBASe? [<source>]

- Functional description:

  Measuring the bottom value of the specified channel waveform, <source> take value as CHANnel1, CHANnel2, CHANnel3 and CHANnel4. Ignore means the current channel.

- Return format:

  Query return-3.120e000，unit is V.

:MEASure:VMIDdle?

- Command format:

  :MEASure:VMIDdle? [<source>]

- Functional description:

  Measuring the middle value of the specified channel waveform,<source> take value as CHANnel1, CHANnel2, CHANnel3 and CHANnel4. Ignore means the current channel.

- Return format:

  Query return0.120e000，unit is V.

:MEASure:VAVerage?

- Command format:

  :MEASure:VAVerage? [<interval>][,<source>]

- Functional description:

  Measuring the average value of the specified channel waveform, <source> take value as CHANnel1, CHANnel2, CHANnel3, CHANnel4 and MATH. If there has no designated <source>, the current channel as <source> by default; <interval> appoint the measurement separation, take value as CYCLe and DISPlay, CYCLe presents integer cycle period, DISPlay presents full screen. If there has no designated <interval>, take DISPlay by default.

- Return format:

Query return1.120e000，unit is V.

:MEASure:VRMS?

- Command format：

  :MEASure:VRMS? [<interval>][,<source>]

- Functional description：

  Measuring the root mean square value of the specified channel waveform, <source> take value as CHANnel1, CHANnel2, CHANnel3, CHANnel4 and MATH. If there has no designated <source>, the current channel as <source> by default; <interval> appoint the measurement separation, take value as CYCLe and DISPlay, CYCLe presents integer cycle period, DISPlay presents full screen. If there has no designated <interval>, take DISPlay by default.

- Return format：

  Query return1.230e000，unit is V.

:MEASure:AREa?

- Command format：

  :MEASure:AREa? [<interval>][,<source>]

- Functional description：

  Measuring the area of the specified channel waveform, <source> take value as CHANnel1, CHANnel2, CHANnel3, CHANnel4 and MATH. If there has no designated <source>, the current channel as <source> by default; <interval> appoint the measurement separation, take value as CYCLe and DISPlay, CYCLe presents integer cycle period, DISPlay presents full screen. If there has no designated <interval>, take DISPlay by default.

- Return format：

  Query return 3.456e002，unit is Vs.

:MEASure:OVERshoot?

- Command format：

  :MEASure:OVERshoot? [<source>]

- Functional description：

  Measuring overshoot of the specified channel waveform, <source> take value as CHANnel1, CHANnel2, CHANnel3 and CHANnel4. Ignore means the current channel.

- Return format：

  Query return1.230e002，unit is V.

:MEASure:PREShoot?

- Command format：

:MEASure:PREShoot? [<source>]

- Functional description:

  Measuring preshoot of the specified channel waveform, <source> take value as CHANnel1, CHANnel2, CHANnel3 and CHANnel4. Ignore means the current channel.

- Return format:

  Query return1.230e-002，unit is V.

:MEASure:FREQuency?

- Command format:

  :MEASure:FREQuency? [<source>]

- Functional description:

  Measuring frequency of the specified channel waveform, <source> take value as CHANnel1, CHANnel2, CHANnel3 and CHANnel4. Ignore means the current channel.

- Return format:

  Query return 2.000e003，unit is Hz.

:MEASure:RISetime?

- Command format:

  :MEASure:RISetime? [<source>]

- Functional description:

  Measuring rise time of the specified channel waveform, <source> take value as CHANnel1, CHANnel2, CHANnel3 CHANnel4. Ignore means the current channel.

- Return format:

  Query return 5.000e-005，unit is s.

:MEASure:FALLtime?

- Command format:

  :MEASure:FALLtime? [<source>]

- Functional description:

  Measuring fall time of the specified channel waveform, <source> take value as CHANnel1, CHANnel2, CHANnel3 and CHANnel4. Ignore means the current channel.

- Return format:

  Query return 5.000e-005，unit is s.

:MEASure:PERiod?

- Command format:

  :MEASure:PERiod? [<source>]

- Functional description:

Measuring period of the specified channel waveform, <source> take value as CHANnel1, CHANnel2, CHANnel3 and CHANnel4. Ignore means the current channel.

- Return format:

  Query return 5.000e-003，unit is s.

:MEASure:PWIDth?

- Command format:

  :MEASure:PWIDth? [<source>]

- Functional description:

  Measuring positive pulse width of the specified channel waveform, <source> take value as CHANnel1, CHANnel2, CHANnel3 and CHANnel4. Ignore means the current channel.

- Return format:

  Query return 5.000e-003，unit is s.

:MEASure:NWIDth?

- Command format:

  :MEASure:NWIDth? [<source>]

- Functional description:

  Measuring negative pulse width of the specified channel waveform, <source> take value as CHANnel1, CHANnel2, CHANnel3 and CHANnel4. Ignore means the current channel.

- Return format:

  Query return 5.000e-003，unit is s.

:MEASure:FRR?

- Command format:

  :MEASure:FRR? <source1>,<source2>

- Functional description:

  Measuring time of the first rise edge between <source1> and <source2>, <source> take value as CHANnel1, CHANnel2, CHANnel3 and CHANnel4.

- Return format:

  Query return 5.000e-003，unit is s.

:MEASure:FRF?

- Command format:

  :MEASure:FRF? <source1>,<source2>

- Functional description:

  Measuring time of the first fall edge between <source1> and <source2>, <source> take value as CHANnel1, CHANnel2, CHANnel3 and CHANnel4. Ignore means the current channel.

- Return format:

  Query return 5.000e-003，unit is s.

:MEASure:FFR?

- Command format:

  :MEASure:FFR? <source1>,<source2>

- Functional description:

  Measuring time between the first fall edge of <source1> and the first rise edge of <source2>, <source> take value as CHANnel1, CHANnel2, CHANnel3 and CHANnel4.

- Return format:

  Query return 5.000e-003，unit is s.

:MEASure:FFF?

- Command format:

  :MEASure:FFF? <source1>,<source2>

- Functional description:

  Measuring time of the first fall edge between <source1> and <source2>, <source> take value as CHANnel1, CHANnel2, CHANnel3 and CHANnel4.

- Return format:

  Query return 5.000e-003，unit is s.

:MEASure:LRR?

- Command format:

  :MEASure:LRR? <source1>,<source2>

- Functional description:

  Measuring time between the first rise edge of <source1> and the rise edge of <source2>, <source> take value as CHANnel1, CHANnel2, CHANnel3 and CHANnel4.

- Return format:

  Query return 5.000e-003，unit is s.

:MEASure:LRF?

- Command format:

  :MEASure:LRF? <source1>,<source2>

- Functional description:

  Measuring time between the first rise edge of <source1> and the first fall edge of <source2>, <source> take value as CHANnel1, CHANnel2, CHANnel3 and CHANnel4.

- Return format:

  Query return 5.000e-003，unit is s.

:MEASure:LFR?

- Command format:

  :MEASure:LFR? <source1>,<source2>

- Functional description:

  Measuring time between the first fall edge of <source1> and the last rise edge of <source2>, <source> take value as CHANnel1, CHANnel2, CHANnel3 and CHANnel4.

- Return format:

  Query return 5.000e-003，unit is s.

:MEASure:LFF?

- Command format:

  :MEASure:LFF? <source1>,<source2>

- Functional description:

  Measuring time between the first fall edge of <source1> and the last fall edge of <source2>, <source> take value as CHANnel1, CHANnel2, CHANnel3 and CHANnel4.

- Return format:

  Query return 5.000e-003，unit is s.

TRIGger Command

This command is to control trigger sweep mode and trigger standard, trigger decide when oscilloscope start to collect data and display waveform.

Trigger Command

:TRIGger:MODE

- Command format:

  :TRIGger:MODE <mode>

  :TRIGger:MODE?

- Functional description:

  Set trigger mode.

  <mode> divided into EDGE（edge trigger）, PULSe（pulse width trigger）, VIDeo（video trigger）, SLOPe（slope trigger）, RUNT（low level trigger）、WINDow（window trigger）、DELay（delay trigger ）、TIMeout（timeout trigger）、DURation （duration trigger）、SHOLd（setup hold trigger）、NE（N edge trigger）、PATTern（code pattern trigger）.

- Return format:

  Query return trigger mode.

- For example:

  :TRIGger:MODE NE N edge trigger

  :TRIGger:MODE? query return NE

:TRIGger:FORCe

- Command format:

  :TRIGger:FORCe

- Functional description:

  Execute the command if the oscilloscope has no apporiate trigger term. Force the oscilloscope to produce a trigger signal to trigger and display the input waveform.

- For example:

  :TRIG:FORC force trigger

:TRIGger:SWEep

- Command format:

  :TRIGger:SWEep {AUTO|NORMal|SINGle}

  :TRIGger:SWEep?

- Functional description:

  Select trigger sweep mode.

  AUTO：If there has no trigger term, internal will produce trigger signal to force trigger

  NORMal：Trigger only when in trigger term

  SINGle：In trigger term, trigger one time and stop

- Return format:

  Query return trigger sweep mode {AUTO|NORMal|SINGle}.

- For example:

  :TRIGger:SWEep AUTO set channel 1 as auto trigger mode

  :TRIGger:SWEep? query return AUTO

:TRIGger:LEVel:ASETup

- Command format:

  :TRIGger:LEVel:ASETup

- Functional description:

  Set trigger level on the vertical center point of signal amplitude.

- For example:

  :TRIG:LEVel:ASETup set trigger level on the center point

:TRIGger:LEVel

- Command format:

  :TRIGger:LEVel <level>

  :TRIGger:LEVel?

- Functional description:

  Set the trigger level value of NORMal mode. <level> value must set based on the conversion of voltage base and screen information.

- Return format:

  Query return <level> value, unit is V.

- For example:

  :TRIG:LEV 2 set trigger level as 2V

  :TRIG:LEV? query return 2.000e000

:TRIGger:LEVel:LOW

- Command format:

  :TRIGger:LEVel:LOW <level>

  :TRIGger:LEVel:LOW?

- Functional description:

  Set low level value of slope trigger. <level>value must set based on the conversion of voltage base and screen information.

- Return format:

  Query return <level> value, unit is V.

- For example:

  :TRIG:LEV:LOW 2 set trigger level as 2V

  :TRIG:LEV:LOW? query return 2.000e000

:TRIGger:LEVel:HIGH

- Command format:

  :TRIGger:LEVel:HIGH <level>

  :TRIGger:LEVel:HIGH?

- Functional description:

  Set high level value of slope trigger. <level> value must set based on the conversion of voltage base and screen information.

- Return format:

  Query return <level> value, unit is V.

- For example:

  :TRIG:LEV:HIGH 2 set trigger level as 2V

:TRIG:LEV:HIGH? query return 2.000e000

:TRIGger:SOURce

- Command format:

  :TRIGger:SOURce <source>

  :TRIGger:SOURce?

- Functional description:

  Set single information source trigger, input channel(CHANnel1, CHANnel 2, CHANnel 3, CHANnel 4）, external trigger(EXT, EXT5), AC Line. Only EDGE/ PULSe / VIDeo support AC Line, EXT and EXT5.

  <source> presents information source trigger,

  CHANnel<n>|EXT|EXT5|ACLine，<n> take value as 1, 2, 3, 4.

- Return format:

  Query return information source trigger { CHANnel1| CHANnel2| CHANnel3| CHANnel4|EXT|EXT5|ACLINE}.

- For example:

  :TRIGger:SOUR CHAN1 set channel 1 as edge trigger

  :TRIGger:SOUR? query return CHANnel1

:TRIGger:COUPling

- Command format:

  :TRIGger:COUPling {DC|AC|LF|HF|NOISE}

  :TRIGger:COUPling?

- Functional description:

  Set the coupling mode, DC, AC, LF, HF and NOISE, except VIDeo.

- Return format:

  Query return coupling mode {DC|AC|LF|HF|NOISE}.

- For example:

  :TRIGger:COUPling AC set edge trigger as AC

  :TRIGger:COUPling? query return AC

Edge Trigger

:TRIGger:EDGE:SLOPe

- Command format:
  :TRIGger:EDGE:SLOPe {POSitive|NEGative|ALTernation}
  :TRIGger:EDGE:SLOPe?
- Functional description:
  Set edge trigger mode，POSitive, NEGative and ALTernation.
- Return format:
  Query return edge trigger mode { POSitive | NEGative | ALTernation }.
- For example:
  :TRIGger:EDGE:SLOP POS　　set edge trigger mode as POSitive
  :TRIGger:EDGE:SLOP?　　query return POSitive

Pulse Width Trigger

:TRIGger:PULSe:QUALifier

- Command format:
  :TRIGger:PULSe:QUALifier　{GREaterthan | LESSthan | EQUal}
  :TRIGger:PULSe:QUALifier?
- Functional description:
  Set the term of pulse time, GREaterthan, LESSthan and EQUal.
- Return format:
  Query return { GREaterthan | LESSthan | EQUal }.
- For example:
  :TRIGger:PULSe:QUALifier GRE　　set the pulse term as GREaterthan
  :TRIGger:PULSe:QUALifier?　　query return GREaterthan

:TRIGger:PULSe:POLarity

- Command format:
  :TRIGger:PULSe:POLarity　{POSitive | NEGative}
  :TRIGger:PULSe:POLarity?
- Functional description:
  Set the pulse polarity, POSitive and NEGative.
- Return format:

Query return{ POSitive | NEGative }.

- For example:

:TRIGger:PULSe:POL POS set pulse polarity as POSitive

:TRIGger:PULSe:POL? query return POSitive

:TRIGger:PULSe:TIME

- Command format:

:TRIGger:PULSe:TIME <time>

:TRIGger:PULSe:TIME?

- Functional description:

Set time interval of pulse width trigger.

- Return format:

Query return the current time interval, unit is s.

- For example:

:TRIGger:PULSe:TIME 1 set pulse width as 1s

:TRIGger:PULSe:TIME? query return1.000e000

Video Trigger

:TRIGger:VIDeo:MODE

- Command format:

:TRIGger:VIDeo:MODE { ODD | EVEN | LINE| ALINes}

:TRIGger:VIDeo:MODE?

- Functional description:

Set the sync mode of video trigger, ODD, EVEN, LINE and ALINes.

- Return format:

Query return{ ODD | EVEN | LINE| ALIN}.

- For example:

:TRIGger:VIDeo:MODE ODD set the sync mode as ODD

:TRIGger:VIDeo:MODE? query return ODD

:TRIGger:VIDeo:STANdard

- Command format:

:TRIGger:VIDeo:STANdard { NTSC | PAL | SECam }

:TRIGger:VIDeo:STANdard?

- Functional description:

  Set the video standard.

- Return format:

  Query return{ NTSC | PAL | SECam }.

- For example:

  :TRIGger:VIDeo:STANdard NTSC set the video standard as NTSC

  :TRIGger:VIDeo:STANdard? query return NTSC

:TRIGger:VIDEO:LINE

- Command format:

  :TRIGger:VIDEO:LINE <value>

  :TRIGger:VIDEO:LINE?

- Functional description:

  Set the specified line of video sync, <value> presents the specified line and the range, which is related to the video standard.

- Return format:

  Query return the specified line.

- For example:

  :TRIG:VIDEO:LINE 50 set the specified line of video sync as 50

  :TRIG:VIDEO:LINE? query return 50

:TRIGger:VIDEO:SRATe?

- Command format:

  :TRIGger:VIDEO:SRATe?

- Functional description:

  To obtain the slew rate of video.

- Return format:

  Query return the current slew rate, return format is conform to binary data IEEE 488.2 # format.

Slope Trigger

:TRIGger:SLOPe:QUALifier

- Command format:

  :TRIGger:SLOPe:QUALifier {GREaterthan | LESSthan | EQUal}

  :TRIGger:SLOPe:QUALifier?
- Functional description:

  Set the term of slope time, GREaterthan, LESSthan and EQUal.
- Return format:

  Query return{GREaterthan | LESSthan | EQUal}.
- For example:

  :TRIGger:SLOPe:QUALifier GRE set the slope term as GREaterthan

  :TRIGger:SLOPe:QUALifier? query return GREaterthan

:TRIGger:SLOPe:SLOPe

- Command format:

  :TRIGger:SLOPe:SLOPe {POSitive|NEGative}

  :TRIGger:SLOPe:SLOPe?
- Functional description:

  Set the type of slope trigger, POSitive and NEGative.
- Return format:

  Query return{POSitive|NEGative}。
- For example:

  :TRIGger:SLOPe:SLOPe POS set the slope trigger as POSitive

  :TRIGger:SLOPe:SLOPe? query return POSitive

:TRIGger:SLOPe:TIME

- Command format:

  :TRIGger:SLOPe:TIME <time>

  :TRIGger:SLOPe:TIME?
- Functional description:

  Set the time interval of slope trigger.
- Return format:

  Query return the current time interval， unit is s.
- For example:

:TRIGger:SLOPe:TIME 1 set the time interval of slope trigger as 1s

:TRIGger:SLOPe:TIME? query return1.000e000

:TRIGger:SLOPe:THReshold

- Command format:

  :TRIGger:SLOPe:THReshold {LOW|HIGH|LH}

  :TRIGger:SLOPe:THReshold?
- Functional description:

  Set the threshold mode of slope trigger.
- Return format:

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

  :TRIGger:SLOPe:THR HIGH set the threshold mode of slope trigger as HIGH

  :TRIGger:SLOPe:THR? query return HIGH

Runt Trigger

:TRIGger:RUNT:QUALifier

- Command format:

  :TRIGger:RUNT:QUALifier {GREaterthan | LESSthan | EQUal | NONE}

  :TRIGger:RUNT:QUALifier?
- Functional description:

  Set time term of low level, GREaterthan, LESSthan, EQUal and NONE.
- Return format:

  Query return{GREaterthan | LESSthan | EQUal | NONE}.
- For example:

  :TRIGger:RUNT:QUALifier GRE set the slope term as GREaterthan

  :TRIGger:RUNT:QUALifier? query return GREaterthan

:TRIGger:RUNT:POLarity

- Command format:

  :TRIGger:RUNT:POLarity {POSitive | NEGative}

  :TRIGger:RUNT:POLarity?
- Functional description:

  Set the polarity of low level, POSitive and NEGative.
- Return format:

Query return{POSitive | NEGative}.

- For example:

:TRIGger:RUNT:POL POS　　set the pulse polarity as POSitive

:TRIGger:RUNT:POL?　　query return POSitive

:TRIGger:RUNT:LEVel

- Command format:

:TRIGger:RUNT:LEVel　{LOW | HIGH}

:TRIGger:RUNT:LEVel?

- Functional description:

Set the trigger mode of low level.

- Return format:

Query return {LOW | HIGH}.

- For example:

:TRIGger:RUNT:LEV HIGH　　set the trigger mode of low level as HIGH

:TRIGger:RUNT:LEV?　　query return HIGH

:TRIGger:RUNT:TIME

- Command format:

:TRIGger:RUNT:TIME　<time>

:TRIGger:RUNT:TIME?

- Functional description:

Set the time interval of low level.

- Return format:

Query return the current time interval，unit is s.

- For example:

:TRIGger:RUNT:TIME 1　　set the time interval of low level as 1s

:TRIGger:RUNT:TIME?　　query return 1.000e000

Exceed-amplitude Trigger

:TRIGger:WINDow:SLOPe

- Command format:

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

  :TRIGger:WINDow:SLOPe?
- Functional description:

  Set edge trigger mode，POSitive, NEGative and ALTernation.
- Return format:

  Query return edge trigger mode {POSitive|NEGative|ALTernation}.
- For example:

  :TRIGger:WINDow:SLOP POS　　set the window trigger as POSitive

  :TRIGger:WINDow:SLOP?　　query return POSitive

:TRIGger:WINDow:LEVel

- Command format:

  :TRIGger:WINDow:LEVel　{LOW | HIGH}

  :TRIGger:WINDow:LEVel?
- Functional description:

  Set the level mode of window trigger.
- Return format:

  Query return {LOW | HIGH}.
- For example:

  :TRIGger:WINDow:LEV HIGH　　set the level mode of window trigger as HIGH

  :TRIGger:WINDow:LEV?　　query return HIGH

:TRIGger:WINDow:TIME

- Command format:

  :TRIGger:WINDow:TIME　<time>

  :TRIGger:WINDow:TIME?
- Functional description:

  Set the time interval of window trigger.
- Return format:

  Query return the current time interval，unit is s.
- For example:

:TRIGger:WINDow:TIME 1	set the time interval of window trigger as 1s
:TRIGger:WINDow:TIME?	query return1.000e000

:TRIGger:WINDow:POSition

- Command format:

  :TRIGger:WINDow:POSition {ENTER|EXIT|TIME}

  :TRIGger:WINDow:POSition?
- Functional description:

  Set the position of window trigger.
- Return format:

  Query return {ENTER|EXIT|TIME}.
- For example:

:TRIGger:WINDow:POS TIME	set the position of window trigger at TIME
:TRIGger:WINDow:POS?	query return TIME

Delay Trigger

:TRIGger:DELay:ARM:SOURce

- Command format:

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

  :TRIGger:DELay:ARM:SOURce?
- Functional description:

  Set the focus source of delay trigger.
- Return format:

  Query return{CHANnel1 | CHANnel2| CHANnel3| CHANnel4}.
- For example:

:TRIGger:DELay:ARM:SOUR CHAN1	set channel 1 as the focus source
:TRIGger:DELay:ARM:SOUR?	query returnCHANnel1

:TRIGger:DELay:ARM:SLOPe

- Command format:

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

  :TRIGger:DELay:ARM:SLOPe?
- Functional description:

  Set edge trigger mode, POSitive and NEGative.
- Return format:

Query return {NEGative | POSitive}.

- For example:

:TRIGger:DELay:ARM:SOUR NEG set edge trigger source as NEGative

:TRIGger:DELay:ARM:SOUR? query return NEGative

:TRIGger:DELay:TRIGger:SOURce

- Command format:

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

:TRIGger:DELay:TRIGger:SOURce?

- Functional description:

Set the delay trigger source.

- Return format:

Query return{CHANnel1 | CHANnel2| CHANnel3| CHANnel4}。

- For example:

:TRIGger:DELay:TRIGger:SOUR CHAN1 set channel 1 as the trigger source

:TRIGger:DELay:TRIGger:SOUR? query return CHANnel1

:TRIGger:DELay:TRIGger:SLOPe

- Command format:

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

:TRIGger:DELay:TRIGger:SLOPe?

- Functional description:

Set edge trigger mode, POSitive and NEGative.

- Return format:

Query return {NEGative | POSitive}.

- For example:

:TRIGger:DELay:TRIGger:SOUR NEG set edge trigger source as NEGative

:TRIGger:DELay:TRIGger:SOUR? query return NEGative

:TRIGger:DELay:QUALifier

- Command format:

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

  :TRIGger:DELay:QUALifier?
- Functional description:

  Set the time interval term of delay trigger, GREaterthan, LESSthan, INRange and OUTRange.
- Return format:

  Query return{ GREaterthan | LESSthan | INRange | OUTRange }.
- For example:

  :TRIGger:DELay:QUALifier GRE set the slope term as GREaterthan

  :TRIGger:DELay:QUALifier? query return GREaterthan

:TRIGger:DELay:MODE

- Command format:

  :TRIGger:DELay:MODE { NORMAL | UPPER | LOW }

  :TRIGger:DELay:MODE?
- Functional description:

  Set the time mode of delay trigger.
- Return format:

  Query return { NORMAL | UPPER | LOW }.
- For example:

  :TRIGger:DELay:MODE UPPER set the delay trigger mode as UPPER

  :TRIGger:DELay:MODE? query return UPPER

:TRIGger:DELay:TIME

- Command format:

  :TRIGger:DELay:TIME <time>

  :TRIGger:DELay:TIME?
- Functional description:

  Set the time interval of delay trigger.
- Return format:

  Query return the current time interval， unit is s.
- For example:

  :TRIGger:DELay:TIME 1 set the time interval of delay trigger as 1s

  :TRIGger:DELay:TIME? query return1.000e000

:TRIGger:DELay:SELect

- Command format:
  :TRIGger:DELay:SELect <SOURce<n>>
  :TRIGger:DELay:SELect
- Functional description:
  For switch selected source, SOURce<n>presents source，n take value as1, 2.
  SOURce1 presents the focus source； SOURce2 presents trigger source.
- Return format:
  Query return{ SOURce1 | SOURce2 }.
- For example:
  :TRIGger:DELay:SELect SOURce1 select the focus source
  :TRIGger:DELay:SELect? query return SOURce1

Timeout Trigger

:TRIGger:TIMEOUT:TIME

- Command format:
  :TRIGger:TIMEOUT:TIME <time>
  :TRIGger:TIMEOUT:TIME?
- Functional description:
  Set the time interval of timeout trigger.
- Return format:
  Query return the current time interval， unit is s.
- For example:
  :TRIGger:TIMEOUT:TIME 1 set the time interval of timeout trigger as 1s
  :TRIGger:TIMEOUT:TIME? query return 1.000e000

:TRIGger:TIMEOUT:SLOPe

- Command format:
  :TRIGger:TIMEOUT:SLOPe {POSitive|NEGative|ALTernation}
  :TRIGger:TIMEOUT:SLOPe?
- Functional description:
  Set edge trigger mode, POSitive, NEGative and ALTernation.
- Return format:
  Query return edge trigger mode {POSitive|NEGative|ALTernation}.

- For example:

  :TRIGger:TIMEOUT:SLOP POS set the edge trigger as POSitive

  :TRIGger:TIMEOUT:SLOP? query return POSitive

Duration Trigger

:TRIGger:DURation:MODE

- Command format:

  :TRIGger:DURation:MODE { NORMAL | UPPER | LOW }

  :TRIGger:DURation:MODE?
- Functional description:

  Set the time mode of duration trigger.
- Return format:

  Query return{ NORMAL | UPPER | LOW }.
- For example:

  :TRIGger:DURation:MODE UPPER set duration trigger as UPPER mode

  :TRIGger:DURation:MODE? query return UPPER

:TRIGger:DURation:PATTern

- Command format:

  :TRIGger:DURation:PATTern { HIGH | LOW | X }

  :TRIGger:DURation:PATTern?
- Functional description:

  Set the code pattern of duration trigger, HIGH(code pattern value as 1), LOW（code pattern value as 0）, X (channel is invalid.)
- Return format:

  Query return{ HIGH | LOW | X }.
- For example:

  :TRIGger:DURation:PATTern HIGH set the code pattern of duration trigger as 1

  :TRIGger:DURation:PATTern? query return HIGH

:TRIGger:DURation:QUALifier

- Command format:

  :TRIGger:DURation:QUALifier { GREaterthan | LESSthan | INRange }

  :TRIGger:DURation:QUALifier?

- Functional description:

  Set the time interval term of delay trigger, GREaterthan, LESSthan and INRange.

- Return format:

  Query return{ GREaterthan | LESSthan | INRange }.

- For example:

  :TRIGger:DURation:QUALifier GRE set the slope terms as GREaterthan

  :TRIGger:DURation:QUALifier? query return GREaterthan

Setup Hold Trigger

:TRIGger:SHOLd:DATA:SOURce

- Command format:

  :TRIGger:SHOLd:DATA:SOURce {CHANnel1 | CHANnel2| CHANnel3| CHANnel4}

  :TRIGger:SHOLd:DATA:SOURce?

- Functional description:

  Set data source of setup hold trigger.

- Return format:

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

- For example:

  :TRIGger:SHOLd:DATA:SOUR CHAN1 set channel 1 as data source

  :TRIGger:SHOLd:DATA:SOUR? query return CHANnel1

:TRIGger:SHOLd:CLOCK:SOURce

- Command format:

  :TRIGger:SHOLd:CLOCK:SOURce {CHANnel1 | CHANnel2| CHANnel3| CHANnel4}

  :TRIGger:SHOLd:CLOCK:SOURce?

- Functional description:

  Set clock source of setup hold trigger.

- Return format:

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

- For example:

  :TRIGger:SHOLd:CLOCK:SOUR CHAN1 set channel 1 as clock source

  :TRIGger:SHOLd:CLOCK:SOUR? query return CHANnel1

:TRIGger:SHOLd:SLOPe

- Command format:

  :TRIGger:SHOLd:SLOPe {POSitive|NEGative}

  :TRIGger:SHOLd:SLOPe?

- Functional description:

  Set edge trigger mode of setup hold, POSitive and NEGative.

- Return format:

  Query return {POSitive|NEGative}.

- For example:

  :TRIGger:SHOLd:SLOPe POS set setup hold trigger as POSitive

  :TRIGger:SHOLd:SLOPe? query return POSitive

:TRIGger:SHOLd:PATTern

- Command format:

  :TRIGger:SHOLd:PATTern { HIGH | LOW }

  :TRIGger:SHOLd:PATTern?

- Functional description:

  Set the code pattern of setup hold trigger, HIGH(code pattern value as 1), LOW（code pattern value as 0）.

- Return format:

  Query return{ HIGH | LOW }.

- For example:

  :TRIGger:SHOLd:PATTern HIGH set the code pattern of setup hold trigger 1

  :TRIGger:SHOLd:PATTern? query return HIGH

:TRIGger:SHOLd:MODE

- Command format:

  :TRIGger:SHOLd:MODE { SETup | HOLD | SH }

  :TRIGger:SHOLd:MODE?

- Functional description:

  Set the time of mode of trigger, SETup, HOLD and SH.

- Return format:

  Query return{ SETup | HOLD | SH }.

- For example:

  :TRIGger:SHOLd:MODE HOLD set the time mode as HOLD

  :TRIGger:SHOLd:MODE? query return HOLD

:TRIGger:SHOLd:TIME:STEP

- Command format:

  :TRIGger:SHOLd:TIME:STEP <step>

  :TRIGger:SHOLd:TIME:STEP?

- Functional description:

  Set time interval of setup hold trigger to adjust stepped value.

- Return format:

  Query return the current stepped value, unit is s.

- For example:

  :TRIGger:SHOLd:TIME:STEP 0.001 set time interval of setup hold trigger as 1ms

  :TRIGger:SHOLd:TIME:STEP? query return 1.000e-003

:TRIGger:SHOLd:TIME

- Command format:

  :TRIGger:SHOLd:TIME <time>

  :TRIGger:SHOLd:TIME?

- Functional description:

  Set the time interval of setup hold trigger.

- Return format:

  Query return the current time interval, unit is s.

- For example:

  :TRIGger:SHOLd:TIME 1 set time interval of setup hold trigger as 1ms

  :TRIGger:SHOLd:TIME? query return 1.000e000

:TRIGger:SHOLd:SELect

- Command format:

  :TRIGger:SHOLd:SELect <SOURce<n>>

  :TRIGger:SHOLd:SELect

- Functional description:

  For switch selected source, SOURce<n> presents source, n take value as1, 2.

  SOURce1 presents data source； SOURce2 presents clock source.

- Return format:

  Query return{ SOURce1 | SOURce2 }.

- For example:

  :TRIGger:SHOLd:SELect SOURce1 set the selected focus source

  :TRIGger:SHOLd:SELect? query return SOURce1

N Edge Trigger

:TRIGger:NEDGE:SLOPe

- Command format:
  :TRIGger:NEDGE:SLOPe {POSitive|NEGative }
  :TRIGger:NEDGE:SLOPe?
- Functional description:
  Set edge trigger mode, POSitive and NEGative.
- Return format:
  Query return edge trigger mode {POSitive|NEGative }.
- For example:
  :TRIGger:NEDGE:SLOP POS set edge trigger as POSitive
  :TRIGger:NEDGE:SLOP? query return POSitive

:TRIGger:NEDGE:TIME

- Command format:
  :TRIGger:NEDGE:TIME <time>
  :TRIGger:NEDGE:TIME?
- Functional description:
  Set time interval of N edge trigger.
- Return format:
  Query return the current time interval, unit is s.
- For example:
  :TRIGger:NEDGE:TIME 1 set time interval of N edge trigger as 1s
  :TRIGger:NEDGE:TIME? query return 1.000e000

Code Pattern Trigger

:TRIGger:PATTern:PATTern

- Command format:
  :TRIGger:PATTern:PATTern { HIGH | LOW | X | POSitive | NEGative }
  :TRIGger:PATTern:PATTern?
- Functional description:
  Set the code pattern, HIGH（code pattern value as 1）, LOW（code pattern value as 0）, X（channel is invalid）, POSitive and NEGative.

- Return format:

  Query return{ HIGH | LOW | X | POSitive | NEGative }。

- For example:

  :TRIGger:PATTern:PATTern HIGH　　set the code pattern as 1

  :TRIGger:PATTern:PATTern?　　query return HIGH

CURSor Command

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

:CURSor:MODE

- Command format:

  :CURSor:MODE　{ TRACK | INDependent }

  :CURSor:MODE?

- Functional description:

  Set cursor mode, TRACK and INDependent.

- Return format:

  Query return{ TRACK | INDependent }.

- For example:

  :CURSor:MODE TRACK　　set cursor mode as TRACK

  :CURSor:MODE?　　query return TRACK

:CURSor:TYPE

- Command format:

  :CURSor:TYPE　{AMPlitude | TIME | CLOSe }

  :CURSor:TYPE?

- Functional description:

  Set cursor mode, AMPlitude, TIME and CLOSe.

- Return format:

  Query return {AMPlitude | TIME | CLOSe }.

- For example:

  :CURSor:TYPE AMP　　set cursor mode as AMPlitude

  :CURSor:TYPE?　　query return AMPlitude

:CURSor:SOURce

- Command format:

:CURSor:SOURce <source>

:CURSor:SOURce?

- Functional description:

Set cursor source of manual cursor mode, <source> take value as CHANnel<n>，n take value as 1, 2, 3, 4.

- Return format:

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

- For example:

:CURSor:SOURce CHAN1	set channel 1 as cursor source
:CURSor:SOURce?	query return CHANnel1

:CURSor:CURA

- Command format:

:CURSor:CURA <value>

:CURSor:CURA?

- Functional description:

Set the horizontal or vertical position of A cursor line. It is related with directive CURSor:TYPE，amplitude presents the vertical position, time presents the horizontal position. Vertical line range [0,699]，horizontal line range [28,227].

- Return format:

Query return the position of A cursor line.

- For example:

:CURSor:CURA 50	set manual A cursor line position as 50
:CURSor:CURA?	query return 50

:CURSor:CURB

- Command format:

:CURSor:CURB <value>

:CURSor:CURB?

- Functional description:

Set the horizontal or vertical position of B cursor line. It is related with directive CURSor:TYPE.

- Return format:

Query return the position of B cursor line.

- For example:

:CURSor:CURB 50	set manual B cursor line position as 50
:CURSor:CURB?	query return 50

STORage Command

This command is to set and read waveform data and system setup.

:STORage:TYPE

- Command format:

  :STORage:TYPE { SETUP | WAVE}

  :STORage:TYPE?

- Functional description:

  Set storage mode, SETUP and WAVE.

- Return format:

  Query return{ SETUP | WAVE}.

- For example:

  :STORage:TYPE WAVE set storage mode as WAVE

  :STORage:TYPE? query return WAVE

:STORage:WAVE:SOURce

- Command format:

  :STORage:WAVE:SOURce <source>

  :STORage:WAVE:SOURce?

- Functional description:

  Select storage channel of waveform data.

  <source> take value as CHANnel<n>，n take value as 1, 2, 3, 4.

- Return format:

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

- For example:

  :STORage:WAVE:SOURce CHAN1 save channel 1 waveform

  :STORage:WAVE:SOURce? query return CHANnel1

:STORage:CSV:STARt

- Command format:

  :STORage:CSV:STARt

- Functional description:

  Start storage CSV file function.

:STORage:SAVE

- Command format:

  :STORage:SAVE

- Functional description:

  Based on the current storage mode to save waveform or set it.

:STORage:LOAD

- Command format:

  :STORage:LOAD

- Functional description:

  Based on the current storage mode to call out setup data.

:STORage:LOCation

- Command format:

  :STORage:LOCation <location>

  :STORage:LOCation?

- Functional description:

  Set the current internal storage position. <location> 0~255，stepped as 1.

- Return format:

  Query return the current internal storage position.

- For example:

  :STORage:LOCation 5 set the fifth file as the internal storage position

  :STORage:LOCation? query return 5

:STORage:DISK:SELect

- Command format:

  :STORage:DISK:SELect {FLASH | UDISK}

  :STORage:DISK:SELect?

- Functional description:

  For select storage medium.

- Return format:

  Query return{FLASH | UDISK}.

- For example:

  :STORage:DISK:SELect FLASH select FLASH as internal storage medium

  :STORage:DISK:SELect? query return FLASH

REF Command

This command is to set and read reference waveform.

:REF:LOCation

- Command format:
  :REF:LOCation <location>
  :REF:LOCation?
- Functional description:
  Set load internal storage position of reference waveform, <location> 0~255，stepped as 1.
- Return format:
  Query return the current internal storage position.
- For example:
  :REF:LOCation 5 set the fifth file as the internal storage position
  :REF:LOCation? query return 5

:REF:LOAD

- Command format:
  :REF:LOAD
- Functional description:
  Load local reference waveform data.

:REF:CLEar

- Command format:
  :REF:CLEar
- Functional description:
  Empty displayed reference waveform.

:REF:DISK:SELect

- Command format:
  :REF:DISK:SELect {FLASH | UDISK}
  :REF:DISK:SELect?
- Functional description:
  For select storage medium of reference waveform.
- Return format:

Query return{FLASH | UDISK}.

- For example:

:REF:DISK:SELect FLASH reference waveform extract from internal storage

:REF:DISK:SELect? query return FLASH

ACQuire Command

This command is to set sampling mode.

:ACQuire:TYPE

- Command format:

:ACQuire:TYPE {NORMal | AVERage | PEAKdetect | ENVelope | HRESolution }

:ACQuire:TYPE?

- Functional description:

Set sampling mode, NORMal, AVERage, PEAKdetect, ENVelope and HRESolution.

- Return format:

Query return {NORMal | AVERage | PEAKdetect | ENVelope | HRESolution }.

- For example:

:ACQ:TYPE AVER set aquire mode as AVERage

:ACQ:TYPE? query return AVERage

~~:ACQuire:MODE~~

- ~~Command format:~~

~~:ACQuire:MODE {RTIMe | ETIMe }~~

~~:ACQuire:MODE?~~

- ~~Functional description:~~

~~Set sampling mode, RTIMe and ETIMe.~~

- ~~Return format:~~

~~Query return{RTIMe | ETIMe }.~~

- ~~For example:~~

~~:ACQ:MODE RTIM set sampling mode as real-time sampling~~

~~:ACQ:MODE? query return RTIMe~~

:ACQuire:AVERages:COUNt

- Command format:

:ACQuire:AVERages:COUNt <count>

:ACQuire:AVERages:COUNt?

- Functional description:

Set average count of average sampling mode, <count> stepped in Nth power of 2, take value from 2~8192 , 1≤N≤30.

- Return format:

  Query return the current average count.

- For example:

  :ACQ:AVER:COUN 32 set average count as 32

  :ACQ:AVER:COUN? query return 32

:ACQuire:MEMory:DEPTh

- Command format:

  :ACQuire:MEMory:DEPTh { AUTO | 700 | 7K | 70K | 700K | 7M | 70M }

  :ACQuire:MEMory:DEPTh?

- Functional description:

  Set depth mode of sampling storage.

- Return format:

  Query return{ AUTO | 700 | 7K | 70K | 700K | 7M | 70M }.

- For example:

  :ACQ:MEM:DEPT ATUO set storage depth as AUTO mode

  :ACQ:MEM:DEPT? query return AUTO

DISPlay Command

This command is to set or query display function or data.

:DISPlay:DATA?

- Command format:

  :DISPlay:DATA?

- Functional description:

  For query the current image data on the screen.

- Return format:

  Query return BMP image data, return data is conform to binary data IEEE 488.2 # format.

- For example:

  :DISPlay:DATA? query return image data image data

  data format：#800012345+bitmap data

:DISPlay:FORMat

- Command format:

  :DISPlay:FORMat { VECTors | DOTS }

  :DISPlay:FORMat?

- Functional description:

  Set display format of sampling point, VECTors and DOTS.

- Return format:

  Query return{ VECTors | DOTS }.

- For example:

  :DISPlay:FORMat VECT set display format as VECTors

  :DISPlay:FORMat query return VECTors

:DISPlay:GRID:BRIGhtness

- Command format:

  :DISPlay:GRID:BRIGhtness <count>

  :DISPlay:GRID:BRIGhtness?

- Functional description:

  Set grid luminance, <count> take value as 1~100, the number is greater the grid brighter.

- Return format:

  Query return the current grid luminance.

- For example:

  :DISPlay:GRID:BRIGhtness 50 set grid luminance as 50

  :DISPlay:GRID:BRIGhtness? query return 50

:DISPlay:WAVE:BRIGhtness

- Command format:

  :DISPlay:WAVE:BRIGhtness <count>

  :DISPlay:WAVE:BRIGhtness?

- Functional description:

  Set waveform luminance, <count> take value as 1~100，the number is greater the waveform brighter.

- Return format:

  Query return the current waveform luminance.

- For example:

  :DISPlay:WAVE:BRIGhtness 50 set waveform luminance as 50

  :DISPlay:WAVE:BRIGhtness? query return 50

~~**:DISPlay:GRID:MODE**~~

- ~~Command format:~~

  ~~:DISPlay:GRID:MODE { FULL | GRID | CROSS | NONE}~~

  ~~:DISPlay:GRID:MODE?~~

- ~~Functional description:~~

~~Display format of grid, FULL（display grid and coordinate）、GRID（display grid）、CROSS（display coordinate）、NONE（not display grid and coordinate）.~~

- ~~Return format:~~

  ~~Query return{ FULL | GRID | CROSS | NONE}.~~

- ~~For example:~~

  ~~:DISPlay:GRID:MODE FULL set display grid and coordinate~~

  ~~:DISPlay:GRID:MODE? query return FULL~~

:DISPlay:CLEar

- Command format:

  :DISPlay:CLEar

- Functional description:

  Empty waveform display on the screen.

:DISPlay:TYPE

- Command format:

  :DISPlay:TYPE {XY12| XY34|YT}

  :DISPlay:TYPE?

- Functional description:

  Set the display type of time base as XY12 (X-Y method: display amplitude of channel 1 on the horizontal axis, display amplitude of channel 2 on the vertical axis). XY34 (X-Y method: display amplitude of channel 3 on the horizontal axis, display amplitude of channel 4 on the vertical axis). YT (Y-T method: display the relative relationship of vertical voltage and horiziontal time).

- Return format:

  Query return XY12, XY34 and YT.

- For example:

  :DISP:TYPE YT set the time base as YT method

  :DISP:TYPE? query return YT

WAVeform Command

This command is to read waveform data and relevant parameters.

:WAVeform:FORMat

- Command format:

  :WAVeform:FORMat { WORD | BYTE | ASCII | CSV | DAT }

  :WAVeform:FORMat?

- Functional description:

Set return format of waveform AD data. This command is only valid for AD data.

BYTE：return 8 bits value, single byte binary data

WORD：return 16 bits valule, short type binary data

ASCII：return waveform AD data character string, single AD value turn to ASCII character string, each data point separated by comma mark. In addition, it is conform to binary data IEEE488.2 format, for example #41234120,125,128.....\n.

CSV：return waveform AD data set of CSV file format <filed separated by comma mark, line separated by\n >, single AD value turn to ASCII character string.

DAT： set return waveform data by built-in data format

For example：single point of AD value is 120

BYTE：return 0x78

WORD：return two bytes 0x78 and 0x00,（little end format）

ASCII：return '120' of ASCII character string

CSV：return '120' of ASCII character string by CSV standard format

- Return format：

Query return { WORD | BYTE | ASCII }.

- For example：

:WAVeform:FORMat BYTE　　return format of waveform data is single byte mode

:WAVeform:FORMat?　　query return BYTE

:WAVeform:VOLtage:FORMat

- Command format：

:WAVeform:VOLtage:FORMat { BYTE | ASCII | CSV }

:WAVeform:VOLtage:FORMat?

- Functional description：

Set return format of waveform point voltage data, this command is only valid for voltage data, unit is the current channel unit.

BYTE：return float type binary data

ASCII：return waveform voltage data character string, single voltage value turn to ASCII character string, each data separated by comma mark and it's conform to binary data IEEE488.2 format. For example#412342.0, 2.1, 2.3.....\n.

CSV：return waveform AD data set of CSV file format <file separated by comma mark, line separated by\n >, single voltage value turn to ASCII character string

For example：voltage value of single point is 2.0V

BYTE：return 0x40、0x00、0x00、0x00

ASCII：return '2.0' of ASCII character string

CSV：return '2.0' of ASCII character string by CSV standard format

- Return format：

Query return{ BYTE | ASCII | CSV }

- For example：

:WAVeform:VOLtage:FORMat BYTE

return format of waveform data is single byte mode

:WAVeform:VOLtage:FORMat? query return BYTE

:WAVeform:SOURce

- Command format:

  :WAVeform:SOURce {CHANnel<n> }

  :WAVeform:SOURce?

- Functional description:

  Set signal source of the cuurent query waveform data

- Return format:

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

- For example:

  :WAVeform:SOURce CHAN1

  set signal source of the current query waveform data as channel 1

  :WAVeform:SOURce? query return CHANnel1

:WAVeform:POINts

- Command format:

  :WAVeform:POINts <points>

  :WAVeform:POINts?

- Functional description:

  Set return waveform point, the default value is 0.

- Return format:

  Query return waveform point.

- For example:

  :WAVeform:POINts 120 set return waveform point as 120

  :WAVeform:POINts? query return 120

:WAVeform:DATA?

- Command format:

  :WAVeform:DATA?

- Functional description:

  To read waveform data of the specified data source.

- Return format:

  Query return:WAVeform:POINts the specified quantity of waveform data, waveform data source is related with :WAVeform:SOURc, data format is related with WAVeform:FORMat, returns data is conform to format of binary data IEEE 488.2 # format.

- For example:

  The execute order to obtain the specified data source the waveform data as following,

:WAVeform:FORMat BYTE	The return format of waveform data is AD single byte mode
:WAVeform:SOURce CHAN1	Set channel 1 as singal source of query waveform data
:WAVeform:POINts 120	set return 120 waveform point
:WAVeform:DATA?	acquire waveform data

:WAVeform:PREamble?

- Command format:

  :WAVeform:PREamble?

- Functional description:

  Query return the current setup parameter of system waveform.

- Return format:

  Query return separated by comma mark “,”

  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), ENVelope (3), HRESolution (4)

  Points：return point of waveform data

  Count：average count in average sampling, other mode is 1

  Xinc：the time difference between in two point of X direction of waveform data source

  Xor：the relative time of trigger point

  Xref：X reference

  Yinc：unit voltage of Y direction

  Yor：the relative position of Y direction and YREF zero position

  Yref：the reference value of Y direction, the center point of screen

- For example:

  :WAVeform:PREamble?

  return 1,0,0,1,8.000e-009,-6.000e-006,0,4.000e-002,0.000e000,100

:WAVeform:XINCrement?

- Command format:

  :WAVeform:XINCrement?

- Functional description:

  Query the time difference between two adjacent points in X direction of the specified waveform data source.

- Return format:

  Query return time base count，unit is s.

- For example:

:WAV:XINC? query return 3.000e-003

:WAVeform:XORigin?

- Command format:

  :WAVeform:XORigin?

- Functional description:

  Query the time from trigger point to XREF of the specified waveform data.

- Return format:

  Query return time value，unit is s.

- For example:

  :WAV:XOR? query return 3.000e-002

:WAVeform:XREFerence?

- Command format:

  :WAVeform:XREFerence?

- Functional description:

  Query reference time baseline of X direction data point.

- Return format:

  Query reference time.

- For example:

  :WAV:XREF? query return 0

:WAVeform:YINCrement?

- Command format:

  :WAVeform:YINCrement?

- Functional description:

  Query unit voltage value in Y direction of the specified waveform data source.

- Return format:

  Query return unit voltage value in Y direction, unit is V.

- For example:

  :WAV:YINC? query return 2.000e000

:WAVeform:YORigin?

- Command format:

  :WAVeform:YORigin?

- Functional description:

  Query vertical offset of the specified waveform data source.

- Return format:

  Query return offset, unit is V.

- For example:

  :WAV:YOR? query return 1.000e001

:WAVeform:YREFerence?

- Command format:

  :WAVeform:YREFerence?

- Functional description:

  Query reference position in Y direction, the system is fixed at half the height of the screen.

- Return format:

  Query return reference position.

- For example:

  :WAV:YREF? query return 100

SBUS Command

This command is to set the relevant parameter of RS232, SPI, I2C bus.

Basic attribute

:SBUS:DISPlay

- Command format:

  :SBUS:DISPlay { {1|ON} | {0|OFF} }

  :SBUS:DISPlay?

- Functional description:

  The switch setting of bus encoding status.

- Return format:

  Query return 1 or 0， it presents ON or OFF.

- For example:

  :SBUS:DISPlay ON turn on bus encoding status to display encoding waveform

  :SBUS:DISPlay? query return 1

:SBUS:MODE

- Command format:

  :SBUS:MODE { RS232 | I2C | SPI | CAN | USB | OFF}

:SBUS:MODE?

- Functional description:

Select bus encoding mode, OFF presents encoding function is turn off.

- Return format:

Query return{ RS232 | I2C | SPI | CAN | USB | OFF }.

- For example:

:SBUS:MODE I2C	select I2C bus encoding mode
:SBUS:MODE?	query return I2C

:SBUS:BASE

- Command format:

:SBUS:BASE {ASCii | BINary | HEX | DEC}

:SBUS:BASE?

- Functional description:

Set display format of bus.

- Return format:

Query return {ASCii | BINary | HEX | DEC}.

- For example:

:SBUS:BASE BIN	set display format of bus encoding as binary
:SBUS:BASE?	query return BINary

:SBUS:EVENt

- Command format:

:SBUS:EVENt { {1|ON} | {0|OFF} }

:SBUS:EVENt?

- Functional description:

The switch setting of bus event.

- Return format:

Query return 1 or 0，it presents ON or OFF.

- For example:

:SBUS:EVENt ON	turn on bus event
:SBUS:EVENt?	query return 1

:SBUS:SQUare

- Command format:

:SBUS:SQUare { {1|ON} | {0|OFF} }

:SBUS:SQUare?

- Functional description:

  The switch setting of pseudo square wave.

- Return format:

  Query return 1 or 0， it presents ON or OFF.

- For example:

  :SBUS:SQUare ON　　turn on pseudo square wave

  :SBUS:SQUare?　　query return 1

:SBUS:VERTical:POSition

- Command format:

  :SBUS:VERTical:POSition　<value>

  :SBUS:VERTical:POSition?

- Functional description:

  Set vertical position value of bus, parameter type is integer, stepped as 6, range [-160,160], screen center as zero point, up to positive, down to negative.

- Return format:

  Query return vertical position value.

- For example:

  :SBUS:VERTical:POSition 10　　vertical position value of bus is 10

  :SBUS:VERTical:POSition?　　query return 10

:SBUS:TRIGger:SWEep

- Command format:

  :SBUS:TRIGger:SWEep {AUTO|NORMal}

  :SBUS:TRIGger:SWEep?

- Functional description:

  Select the sweep mode of bus trigger.

  AUTO：if there no trigger term, internal will produce trigger signal to force trigger.

  NORMal：only trigger when in trigger term.

- Return format:

  Query return trigger sweep mode {AUTO|NORMal}.

- For example:

  :SBUS:TRIGger:SWEep AUTO　　set sweep mode of bus trigger as AUTO

  :SBUS:TRIGger:SWEep?　　query return AUTO

:SBUS:TRIGger:COUPling

- Command format:
  :SBUS:TRIGger:COUPling {DC|AC|LF|HF|NOISE}
  :SBUS:TRIGger:COUPling?
- Functional description:
  Set coupling mode of bus trigger, DC, AC, LF, HF and NOISE.
- Return format:
  Query return coupling mode {DC|AC|LF|HF|NOISE}.
- For example:
  :SBUS:TRIGger:COUPling AC set coupling mode as AC
  :SBUS:TRIGger:COUPling? query return AC

RS232

:SBUS:RS232:BAUDrate

- Command format:
  :SBUS:RS232:BAUDrate <baudrate>
  :SBUS:RS232:BAUDrate?
- Functional description:
  Set baud rate of RS232 bus encoding. Parameter is integer, range 120~5000000.
- Return format:
  Query return baud rate.
- For example:
  :SBUS:RS232:BAUDrate 500 set RS232 baud rate as 500b/s
  :SBUS:RS232:BAUDrate? query return 500

:SBUS:RS232:BITorder

- Command format:
  :SBUS:RS232:BITorder {LSBFirst | MSBFirst}
  :SBUS:RS232:BITorder?
- Functional description:
  Set byte order of RS232 bus encoding. LSBFirst (little-end mode), MSBFirst(big-end mode).
- Return format:
  Query return{LSBFirst | MSBFirst}.

- For example:

  :SBUS:RS232:BITorder LSBF set RS232 byte order as LSB

  :SBUS:RS232:BITorder? query return LSBFirst

:SBUS:RS232:SOURce

- Command format:

  :SBUS:RS232:SOURce { CHANnel1 | CHANnel2 | CHANnel3 | CHANnel4 }

  :SBUS:RS232:SOURce?
- Functional description:

  Set encoding source of RS232.
- Return format:

  Query return{ CHANnel1 | CHANnel2 | CHANnel3 | CHANnel4 }.
- For example:

  :SBUS:RS232:SOURce CHANnel1 set channel 1 as RS232 bus encoding source

  :SBUS:RS232:SOURce? query return CHANnel1

:SBUS:RS232:POLarity

- Command format:

  :SBUS:RS232:POLarity { POSitive | NEGative }

  :SBUS:RS232:POLarity?
- Functional description:

  Set the polarity of RS232 bus encoding, POSitive and NEGative.
- Return format:

  Query return{ POSitive | NEGative }.
- For example:

  :SBUS:RS232:POLarity POSitive set the polarity of RS232 bus encoding as POSitive

  :SBUS:RS232:POLarity? query return POSitive

:SBUS:RS232:PARity

- Command format:

  :SBUS:RS232:PARity {EVEN | ODD | NONE}

  :SBUS:RS232:PARity?
- Functional description:

  Set parity of RS232 bus.
- Return format:

  Query return{EVEN | ODD | NONE}.

- For example:

  :SBUS:RS232:PARity ODD et parity of RS232 bus as ODD

  :SBUS:RS232:PARity? query return 6

:SBUS:RS232:DATA:BIT

- Command format:

  :SBUS:RS232:DATA:BIT {5 | 6 | 7 | 8 }

  :SBUS:RS232:DATA:BIT?

- Functional description:

  Set data bit of RS232 bus.

- Return format:

  Query return{5 | 6 | 7 | 8 }.

- For example:

  :SBUS:RS232:DATA:BIT 6 set RS232 data bit as 6

  :SBUS:RS232:DATA:BIT? query return 6

:SBUS:RS232:STOP:BIT

- Command format:

  :SBUS:RS232:STOP:BIT {1 | 2 }

  :SBUS:RS232:STOP:BIT?

- Functional description:

  Set stop bit of RS232 bus.

- Return format:

  Query return{1 | 2 }.

- For example:

  :SBUS:RS232:STOP:BIT 6 set stop bit as 6

  :SBUS:RS232:STOP:BIT? query return 6

:SBUS:RS232:DATA

- Command format:

  :SBUS:RS232:DATA <value>

  :SBUS:RS232:DATA?

- Functional description:

  Set RS232 bus data,, parameter is character string of 0x format, range is related with setup parameter of command SBUS:RS232:DATA:BIT, which is [0~2^databit – 1].

- Return format:

Query return character string of 0x format of bus data.

- For example:

  :SBUS:RS232:DATA "0x7F" set RS232 bus data as 0x7F

  :SBUS:RS232:DATA? query return 0x7F

:SBUS:RS232:QUALifier

- Command format:

  :SBUS:RS232:QUALifier {BEGFrame | ERRFrame | ECCError | DATA}

  :SBUS:RS232:QUALifier?

- Functional description:

  Set RS232 bus term.

- Return format:

  Query return{BEGFrame|ERRFrame|ECCError|DATA}.

- For example:

  :SBUS:RS232:QUALifier ERRF set RS232 bus term as error frame

  :SBUS:RS232:QUALifier? query return ERRFrame

I2C

:SBUS:I2C:CLOCk:SOURce

- Command format:

  :SBUS:I2C:CLOCk:SOURce { CHANnel1 | CHANnel2 | CHANnel3 | CHANnel4 }

  :SBUS:I2C:CLOCk:SOURce?

- Functional description:

  Set clock source of I2C bus.

- Return format:

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

- For example:

  :SBUS:I2C:CLOCk:SOURce CHANnel1 set channel 1 as I2C bus clock source

  :SBUS:I2C:CLOCk:SOURce? query return CHANnel1

:SBUS:I2C:DATA:SOURce

- Command format:

  :SBUS:I2C:DATA:SOURce { CHANnel1 | CHANnel2 | CHANnel3 | CHANnel4 }

  :SBUS:I2C:DATA:SOURce?

- Functional description:

Set data source of I2C bus.

- Return format:

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

- For example:

  :SBUS:I2C:DATA:SOURce CHANnel1 set channel 1 as I2C bus data source

  :SBUS:I2C:DATA:SOURce? query return CHANnel1

:SBUS:I2C:ASIZe

- Command format:

  :SBUS:I2C:ASIZe {7 | 10}

  :SBUS:I2C:ASIZe?

- Functional description:

  Set bit wide of I2C bus address.

- Return format:

  Query return{7 | 10}.

- For example:

  :SBUS:I2C:ASIZe 7 set bit wide as 7

  :SBUS:I2C:ASIZe? query return 7

:SBUS:I2C:ADIRection

- Command format:

  :SBUS:I2C:ADIRection { READ | WRITE }

  :SBUS:I2C:ADIRection?

- Functional description:

  Set direction of I2C bus address.

- Return format:

  Query return{ READ | WRITE }.

- For example:

  :SBUS:I2C:ADIRection READ et I2C bus address direction as read only

  :SBUS:I2C:ADIRection? query return READ

:SBUS:I2C:ADDRess

- Command format:

  :SBUS:I2C:ADDRess <value>

  :SBUS:I2C:ADDRess?

- Functional description:

Set I2C bus address, parameter is character string of 0x format, range is related with setup parameter of command SBUS:I2C:ASIZe, which is [0~2^addressbit – 1].

- Return format:

  Query return character string address value of 0x format.

- For example:

  :SBUS:I2C:ADDRess "0x7F" I2C bus address as 0x7F

  :SBUS:I2C:ADDRess? query return 0x7F

:SBUS:I2C:DATA:QUALifier

- Command format:

  :SBUS:I2C:DATA:QUALifier {EQUal | LESSthan | GREaterthan}

  :SBUS:I2C:DATA:QUALifier?

- Functional description:

  Set data term of I2C bus.

- Return format:

  Query return{EQUal | LESSthan | GREaterthan}.

- For example:

  :SBUS:I2C:DATA:QUALifier EQUal set I2C bus term as EQUal

  :SBUS:I2C:DATA:QUALifier? query return EQUal

:SBUS:I2C:DATA

- Command format:

  :SBUS:I2C:DATA <value>

  :SBUS:I2C:DATA?

- Functional description:

  Set I2C bus data, parameter is character string of 0x format, data range is 0x0~0xFFFFFFFFFFFFFFFF.

- Return format:

  Query return data of character string of 0x format.

- For example:

  :SBUS:I2C:DATA "0xFFFFF" set I2C bus data as 0xFFFFF

  :SBUS:I2C:DATA? query return 0xFFFFF

:SBUS:I2C:QUALifier

- Command format:

  :SBUS:I2C:QUALifier {STARt | RESTart | STOP | LOSS | ADDRess | DATA | ADATA}

:SBUS:I2C:QUALifier?

- Functional description:

  Set I2C bus term.

- Return format:

  Query return{STARt | RESTart | STOP | LOSS | ADDRess | DATA | ADATA}.

- For example:

  :SBUS:I2C:QUALifier STOP　　set I2C bus term as STOP

  :SBUS:I2C:QUALifier?　　query return STOP

SPI

:SBUS:SPI:CS:SOURce

- Command format:

  :SBUS:SPI:CS:SOURce　{ CHANnel1 | CHANnel2 | CHANnel3 | CHANnel4 }

  :SBUS:SPI:CS:SOURce?

- Functional description:

  Set chip selection source of SPI bus.

- Return format:

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

- For example:

  :SBUS:SPI:CS:SOURce CHANnel1　　set channel 1 as chip selection source

  :SBUS:SPI:CS:SOURce?　　query return CHANnel1

:SBUS:SPI:CLOCk:SOURce

- Command format:

  :SBUS:SPI:CLOCk:SOURce　{ CHANnel1 | CHANnel2 | CHANnel3 | CHANnel4 }

  :SBUS:SPI:CLOCk:SOURce?

- Functional description:

  Set clock source of SPI bus.

- Return format:

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

- For example:

  :SBUS:SPI:CLOCk:SOURce CHANnel1　　set channel 1 as clock source of SPI bus

  :SBUS:SPI:CLOCk:SOURce?　　query return CHANnel1

:SBUS:SPI:MISO:SOURce

- Command format:

  :SBUS:SPI:MISO:SOURce { CHANnel1 | CHANnel2 | CHANnel3 | CHANnel4 | OFF}

  :SBUS:SPI:MISO:SOURce?

- Functional description:

  Set host computer input and slave computer output source of SPI bus.

- Return format:

  Query return{ CHANnel1 | CHANnel2 | CHANnel3 | CHANnel4 | OFF}.

- For example:

  :SBUS:SPI:MISO:SOURce CHANnel1

  set channel 1 as host computer input and slave computer output source of SPI bus

  :SBUS:SPI:MISO:SOURce? query return CHANnel1

:SBUS:SPI:MOSI:SOURce

- Command format:

  :SBUS:SPI:MOSI:SOURce { CHANnel1 | CHANnel2 | CHANnel3 | CHANnel4 | OFF}

  :SBUS:SPI:MOSI:SOURce?

- Functional description:

  Set host computer output and slave computer input source of SPI bus

- Return format:

  Query return{ CHANnel1 | CHANnel2 | CHANnel3 | CHANnel4 | OFF}。

- For example:

  :SBUS:SPI:MOSI:SOURce CHANnel1

  set channel 1 as host computer output and slave computer input source of SPI bus

  :SBUS:SPI:MOSI:SOURce? query return CHANnel1

:SBUS:SPI:BITorder

- Command format:

  :SBUS:SPI:BITorder {LSBFirst | MSBFirst}

  :SBUS:SPI:BITorder?

- Functional description:

  Set encoding byte order of SPI bus, LSBFirst(little-end mode), MSBFirst(big-end mode).

- Return format:

  Query return {LSBFirst | MSBFirst}.

- For example:

  :SBUS:SPI:BITorder LSBF set byte order as LSB

  :SBUS:SPI:BITorder? query return LSBFirst

:SBUS:SPI:CS:POLarity

- Command format:

  :SBUS:SPI:CS:POLarity {NEGative | POSitive}

  :SBUS:SPI:CS:POLarity?
- Functional description:

  Set the polarity of SPI bus chip selection, POSitive and NEGative.
- Return format:

  Query return{NEGative | POSitive}.
- For example:

  :SBUS:SPI:CS:POLarity POSitive set the polarity of SPI bus chip selection as POSitive

  :SBUS:SPI:CS:POLarity? query return POSitive

:SBUS:SPI:CLOCk:POLarity

- Command format:

  :SBUS:SPI:CLOCk:POLarity {NEGative | POSitive}

  :SBUS:SPI:CLOCk:POLarity?
- Functional description:

  Set the clock polarity of SPI bus, POSitive and NEGative.
- Return format:

  Query return{NEGative | POSitive}.
- For example:

  :SBUS:SPI:CLOCk:POLarity POSitive set the clock polarity of SPI bus as POSitive

  :SBUS:SPI:CLOCk:POLarity? query return POSitive

:SBUS:SPI:MISO:POLarity

- Command format:

  :SBUS:SPI:MISO:POLarity {NEGative | POSitive}

  :SBUS:SPI:MISO:POLarity?
- Functional description:

  Set the polarity of host computer input slave computer output source of SPI bus, POSitive and NEGative.
- Return format:

  Query return{NEGative | POSitive}.
- For example:

  :SBUS:SPI:MISO:POLarity POSitive

the polarity of host computer input slave computer output source as POSitive

:SBUS:SPI:MISO:POLarity? query return POSitive

:SBUS:SPI:MOSI:POLarity

- Command format:

  :SBUS:SPI:MOSI:POLarity {NEGative | POSitive}

  :SBUS:SPI:MOSI:POLarity?

- Functional description:

  Set host computer output slave computer input polarity of SPI bus, POSitive and NEGative.

- Return format:

  Query return{NEGative | POSitive}.

- For example:

  :SBUS:SPI:MOSI:POLarity POSitive set the polarity as POSitive

  :SBUS:SPI:MOSI:POLarity? query return POSitive

:SBUS:SPI:WIDTh

- Command format:

  :SBUS:SPI:WIDTh {4 | 8 | 12 | 16}

  :SBUS:SPI:WIDTh?

- Functional description:

  Set bit wide of SPI bus data.

- Return format:

  Query return{4 | 8 | 12 | 16}.

- For example:

  :SBUS:SPI:WIDTh 4 set bit wide as 4

  :SBUS:SPI:WIDTh? query return 4

~~:SBUS:SPI:TRIGger:TYPE~~

- ~~Command format:~~

  ~~:SBUS:SPI:TRIGger:TYPE {MOSI | MISO | ANY}~~

  ~~:SBUS:SPI:TRIGger:TYPE?~~

- ~~Functional description:~~

  ~~Set trigger mode of SPI bus, MOSI（host computer output slave computer input）, MISO（host computer input slave computer input）and ANY.~~

- ~~Return format:~~

~~Query return {MOSI | MISO | ANY}。~~

- ~~For example:~~

~~:SBUS:SPI:TRIGger:TYPE MOSI set trigger mode of SPI bus as MOSI~~

~~:SBUS:SPI:TRIGger:TYPE? query return MOSI~~

:SBUS:SPI:TRIGger:TIMeout

- Command format:

:SBUS:SPI:TRIGger:TIMeout <vlaue>

:SBUS:SPI:TRIGger:TIMeout?

- Functional description:

Set timeout of SPI bus trigger, parameter type is integer.

<vlaue> equal to n * 4ns and value not exceed range [100ns,1s) .n range [25,25*10^8].

- Return format:

Query return timeout value of trigger by scientific notation method，unit is s.

- For example:

:SBUS:SPI:TRIGger:TIMeout 100ns set timeout of SPI bus trigger as 100ns

:SBUS:SPI:TRIGger:TIMeout? query return 1.000e-007

:SBUS:SPI:DATA

- Command format:

:SBUS:SPI:DATA <value>

:SBUS:SPI:DATA?

- Functional description:

Set SPI bus data, parameter is character string of 0x format，data range 0x0~0xFFFFFFFFFFFFFFFF.

- Return format:

Query return data of character string of 0x format.

- For example:

:SBUS:SPI:DATA "0xFFFFF" set SPI bus data as 0xFFFFF

:SBUS:SPI:DATA? query return 0xFFFFF

:SBUS:SPI:QUALifier

- Command format:

:SBUS:SPI:QUALifier {CS | CS&MOSI | CS&MISO | CS&ANY | IDLE | IDLE&MOSI | IDLE&MISO | IDLE&ANY}

:SBUS:SPI:QUALifier?

- Functional description:

  Set SPI bus term.

- Return format:

  Query return{CS | CS&MOSI | CS&MISO | CS&ANY | IDLE | IDLE&MOSI | IDLE&MISO | IDLE&ANY}.

- For example:

  :SBUS:SPI:QUALifier CS set I2C bus term as CS

  :SBUS:SPI:QUALifier? query return CS

:SBUS:SPI:FRAMelen

- Command format:

  :SBUS:SPI:FRAMelen <len>

  :SBUS:SPI:FRAMelen?

- Functional description:

  Set frame length of SPI bus data.

- Return format:

  Query return<len>.

- For example:

  :SBUS:SPI:FRAMelen 1 set frame length of SPI bus data as 1

  :SBUS:SPI:FRAMelen? query return 1

CAN

:SBUS:CAN:POSitive:SOURce

- Command format:

  :SBUS:CAN:POSitive:SOURce { CHANnel1 | CHANnel2 | CHANnel3 | CHANnel4 }

  :SBUS:CAN:POSitive:SOURce?

- Functional description:

  Set input positive source of CAN bus.

- Return format:

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

- For example:

  :SBUS:CAN:POSitive:SOURce CHANnel1 set channel 1 as CAN bus positive source

  :SBUS:CAN:POSitive:SOURce? query return CHANnel1

:SBUS:CAN:NEGative:SOURce

- Command format:

  :SBUS:CAN:NEGative:SOURce { CHANnel1 | CHANnel2 | CHANnel3 | CHANnel4 }

  :SBUS:CAN:NEGative:SOURce?

- Functional description:

  Set input negative source of CAN bus

- Return format:

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

- For example:

  :SBUS:CAN:NEGative:SOURce CHANnel1 set channel 1 as CAN bus negative source

  :SBUS:CAN:NEGative:SOURce? query return CHANnel1

:SBUS:CAN:SIGNal:DEFinition

- Command format:

  :SBUS:CAN:SIGNal:DEFinition { CANH | CANL | RXTX | DIFFerential }

  :SBUS:CAN:SIGNal:DEFinition?

- Functional description:

  Set signal mode of CAN bus.

- Return format:

  Query return{ CANH | CANL | RXTX | DIFFerential }.

- For example:

  :SBUS:CAN:SIGNal:DEFinition CANH set CANH as the signal of CAN bus

  :SBUS:CAN:SIGNal:DEFinition? query return CANH

:SBUS:CAN:SAMPlepoint

- Command format:

  :SBUS:CAN:SAMPlepoint <percent>

  :SBUS:CAN:SAMPlepoint?

- Functional description:

  Set the sampling percentage of CAN bus signal, <percent> range 1~99.

- Return format:

  Query return sampling percentage.

- For example:

  :SBUS:CAN:SAMPlepoint 30 sampling percentage as 30%

  :SBUS:CAN:SAMPlepoint? query return 30

:SBUS:CAN:SIGNal:BAUDrate

- Command format:

  :SBUS:CAN:SIGNal:BAUDrate <baudrate>

  :SBUS:CAN:SIGNal:BAUDrate?

- Functional description:

  Set the baud rate of CAN bus signal, <baudrate> range 10000~1000000, unit is bps.

- Return format:

  Query return the baud rate of signal.

- For example:

  :SBUS:CAN:SIGNal:BAUDrate 100000 set the baud rate as 100kbps

  :SBUS:CAN:SIGNal:BAUDrate? query return 100000

:SBUS:CAN:QUALifier

- Command format:

  :SBUS:CAN:QUALifier {START |TYPE |ID | DATA | ACKerror | BITFILL | IDDATA | END}

  :SBUS:CAN:QUALifier?

- Functional description:

  Set the the trigger term of CAN bus.

- Return format:

  Query return{START | TYPE | ID | DATA | ACKerror | BITFILL | IDDATA | END}.

- For example:

  :SBUS:SPI:QUALifier ACK set the trigger term as ACKerror

  :SBUS:SPI:QUALifier? query return ACKerror

:SBUS:CAN:FRAMe:TYPE

- Command format:

  :SBUS:CAN:FRAMe:TYPE { DATA | REMote | OVERload | ERRor }

  :SBUS:CAN:FRAMe:TYPE?

- Functional description:

  Set the trigger frame mode of CAN bus.

- Return format:

  Query return{ DATA | REMote | OVERload | ERRor }.

- For example:

  :SBUS:CAN:FRAMe:TYPE ERRor set CAN bus trigger mode as ERRor

  :SBUS:CAN:FRAMe:TYPE? query return ERRor

:SBUS:CAN:ID:MODE

- Command format:

  :SBUS:CAN:ID:MODE {STANdard | EXTended}

  :SBUS:CAN:ID:MODE?

- Functional description:

  Set ID frame format of CAN bus.

- Return format:

  Query return{STANdard | EXTended}。

- For example:

  :SBUS:CAN:ID:MODE STANdard set ID frame format as standars frame

  :SBUS:CAN:ID:MODE? query return STANdard

:SBUS:CAN:ID:STANdard

- Command format:

  :SBUS:CAN:ID:STANdard <string>

  :SBUS:CAN:ID:STANdard?

- Functional description:

  Set ID standard frame data of CAN bus, parameter is character string of 0x format. Standard frame range 0x0~0xFFF.

- Return format:

  Query return character string of 0x format.

- For example:

  :SBUS:CAN:ID:STANdard "0xFFF" set ID standard frame data as 0xFFF

  :SBUS:CAN:ID:STANdard? query return 0xFFF

:SBUS:CAN:ID:EXTend

- Command format:

  :SBUS:CAN:ID:EXTend <string>

  :SBUS:CAN:ID:EXTend?

- Functional description:

  Set ID expand frame data of CAN bus, parameter is character string of 0x format. Expand frame range 0x0~0xFFFFF.

- Return format:

  Query return character string of 0x format.

- For example:

:SBUS:CAN:ID:EXTend "0xFFFFF" set ID expand frame data as 0xFFFFF
:SBUS:CAN:ID:EXTend? query return 0xFFFFF

:SBUS:CAN:ID:DIRection

- Command format:
  :SBUS:CAN:ID:DIRection { READ | WRITE | ANY}
  :SBUS:CAN:ID:DIRection?
- Functional description:
  Set ID direction of CAN bus.
- Return format:
  Query return{ READ | WRITE | ANY}.
- For example:
  :SBUS:CAN:ID:DIRection READ set ID direction as read only
  :SBUS:CAN:ID:DIRection? query return READ

:SBUS:CAN:DATA:QUALifier

- Command format:
  :SBUS:CAN:DATA:QUALifier {EQUal|LESSthan|GREaterthan|NEQUal|LEQUal|GEQUal}
  :SBUS:CAN:DATA:QUALifier?
- Functional description:
  Set compare term of CAN bus DATA.
- Return format:
  Query return{EQUal | LESSthan | GREaterthan | NEQUal | LEQUal | GEQUal}.
- For example:
  :SBUS:CAN:DATA:QUALifier EQUal set compare term as EQUal
  :SBUS:CAN:DATA:QUALifier? query return EQUal

:SBUS:CAN:DATA

- Command format:
  :SBUS:CAN:DATA <string>
  :SBUS:CAN:DATA?
- Functional description:
  Set DATA of CAN bus, parameter is character string of 0x format, data range 0x0~0xFFFFFFFFFFFFFFFF.
- Return format:
  Query return character string of 0x format.

- For example:

  :SBUS:CAN:ID "0xFFFFF" set DATA as 0xFFFFF

  :SBUS:CAN:ID? query return 0xFFFFF

USB

:SBUS:USB:POSitive:SOURce

- Command format:

  :SBUS:USB:POSitive:SOURce { CHANnel1 | CHANnel2 | CHANnel3 | CHANnel4 }

  :SBUS:USB:POSitive:SOURce?
- Functional description:

  Set input positive source of USB bus.
- Return format:

  Query return{ CHANnel1 | CHANnel2 | CHANnel3 | CHANnel4 }.
- For example:

  :SBUS:USB:POSitive:SOURce CHANnel1 set channel 1 as positive source of USB bus

  :SBUS:USB:POSitive:SOURce? query return CHANnel1

:SBUS:USB:NEGative:SOURce

- Command format:

  :SBUS:USB:NEGative:SOURce { CHANnel1 | CHANnel2 | CHANnel3 | CHANnel4 }

  :SBUS:USB:NEGative:SOURce?
- Functional description:

  Set input negative source of USB bus.
- Return format:

  Query return{ CHANnel1 | CHANnel2 | CHANnel3 | CHANnel4 }.
- For example:

  :SBUS:USB:NEGative:SOURce CHANnel1 set channel 1 as negative source of USB bus

  :SBUS:USB:NEGative:SOURce? query return CHANnel1

:SBUS:USB:SPEED

- Command format:

  :SBUS:USB:SPEED { LOW | FULL}

  :SBUS:USB:SPEED?
- Functional description:

Set speed of USB bus.

- Return format:

  Query return{ LOW | FULL}.

- For example:

  :SBUS:USB:SPEED LOW set speed as LOW

  :SBUS:USB:SPEED? query return LOW

:SBUS:USB:QUALifier

- Command format:

  :SBUS:USB:QUALifier {SYNC|RESET|PAUSE|RESUME|END|TOKEN|HANDSHAKE|
  DATA| ERRor }

  :SBUS:USB:QUALifier?

- Functional description:

  Set trigger term of USB bus.

- Return format:

  Query return{SYNC|RESET|PAUSE|RESUME|END|TOKEN|HANDSHAKE|DATA| ERRor }.

- For example:

  :SBUS:USB:QUALifier ERRor set trigger term as errot frame

  :SBUS:USB:QUALifier? query return ERRor

:SBUS:USB:TOKEN:TYPE

- Command format:

  :SBUS:USB:TOKEN:TYPE { ANY | OUT | IN | SOF | SETUP}

  :SBUS:USB:TOKEN:TYPE?

- Functional description:

  Set token mode of USB.

- Return format:

  Query return{ ANY | OUT | IN | SOF | SETUP}.

- For example:

  :SBUS:USB:TOKEN:TYPE ANY set USB toke mode as ANY

  :SBUS:USB:TOKEN:TYPE? query return ANY

:SBUS:USB:TOKEN:ENDPoint

- Command format:

  :SBUS:USB:TOKEN:ENDPoint <point>

  :SBUS:USB:TOKEN:ENDPoint?

- Functional description:

  Set endpoint number of USB token pack, range 0~15.

- Return format:

  Query return token endpoint number.

- For example:

  :SBUS:USB:TOKEN:ENDPoint 10 set endpoint number as 10

  :SBUS:USB:TOKEN:ENDPoint? query return 10

:SBUS:USB:TOKEN:QUALifier

- Command format:

  :SBUS:USB:TOKEN:QUALifier {EQUal|LESSthan|GREaterthan|NEQUal|LEQUal|GEQUal|INRange|OUTRange}

  :SBUS:USB:TOKEN:QUALifier?

- Functional description:

  Set trigger term of USB token pack.

- Return format:

  Query return{EQUal|LESSthan|GREaterthan|NEQUal|LEQUal|GEQUal|INRange|OUTRange}.

- For example:

  :SBUS:USB:TOKEN:QUALifier EQUal set trigger term as EQUal

  :SBUS:USB:TOKEN:QUALifier? query return EQUal

:SBUS:USB:TOKEN:ADDRess

- Command format:

  :SBUS:USB:TOKEN:ADDRess <string>

  :SBUS:USB:TOKEN:ADDRess?

- Functional description:

  Set address of USB token pack, parameter is character string of 0x format. Data range 0x00~0x7F.

- Return format:

  Query return character string of 0x format.

- For example:

  :SBUS:USB:TOKEN:ADDRess "0x5F" set address of USB toke packe as 0x5F

  :SBUS:USB:TOKEN:ADDRess? query return 0x5F

:SBUS:USB:TOKEN:FRAMenum

- Command format:

:SBUS:USB:TOKEN:FRAMenum <number>

:SBUS:USB:TOKEN:FRAMenum?

- Functional description:

Set frame of SOF token pack of USB.

- Return format:

Query return frame of SOF token pack.

- For example:

:SBUS:USB:TOKEN:FRAMenum 20 set frame of SOF token pack as 20

:SBUS:USB:TOKEN:FRAMenum? query return 20

:SBUS:USB:HAND:TYPE

- Command format:

:SBUS:USB:HAND:TYPE {ANY | STALL | ACK | NAK}

:SBUS:USB:HAND:TYPE?

- Functional description:

Set handshake pack mode of USB.

- Return format:

Query return{ANY | STALL | ACK | NAK}.

- For example:

:SBUS:USB:HAND:TYPE ANY set handshake pack mode as ANY

:SBUS:USB:HAND:TYPE? query return ANY

:SBUS:USB:DATA:TYPE

- Command format:

:SBUS:USB:DATA:TYPE {ANY | DATA0 | DATA1}

:SBUS:USB:DATA:TYPE?

- Functional description:

Set data pack mode of USB.

- Return format:

Query return{ANY | DATA0 | DATA1}.

- For example:

:SBUS:USB:DATA:TYPE ANY set data pack mode as ANY

:SBUS:USB:DATA:TYPE? query return ANY

:SBUS:USB:DATA:QUALifier

- Command format:

:SBUS:USB:DATA:QUALifier {EQUal|LESSthan|GREaterthan|NEQUal|LEQUal|GEQUal|INRange|OUTRange}

:SBUS:USB:DATA:QUALifier?

- Functional description:

Set trigger term of USB data pack.

- Return format:

Query return{EQUal|LESSthan|GREaterthan|NEQUal|LEQUal|GEQUal|INRange|OUTRange}.

- For example:

:SBUS:USB:DATA:QUALifier EQUal set trigger term as EQUal

:SBUS:USB:DATA:QUALifier? query return EQUal

:SBUS:USB:DATA:OFFSet

- Command format:

:SBUS:USB:DATA:OFFSet <offset>

:SBUS:USB:DATA:OFFSet?

- Functional description:

Set offset byte of USB data pack, <offset> is integer data, range 0~15.

- Return format:

Query return offset of USB data pack.

- For example:

:SBUS:USB:DATA:OFFSet 2 set offset byte as 2

:SBUS:USB:DATA:OFFSet? query return 2

:SBUS:USB:DATA:LOW

- Command format:

:SBUS:USB:DATA:LOW <string>

:SBUS:USB:DATA:LOW?

- Functional description:

Set low trigger data of USB data pack, parameter is character string of 0x format, data range 0x0~0xFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF.

- Return format:

Query return character string of 0x format.

- For example:

:SBUS:USB:DATA:LOW "0x5F" set negative data of USB signal line as 0x5F

:SBUS:USB:DATA:LOW? query return 0x5F

:SBUS:USB:DATA:HIGH

- Command format:

  :SBUS:USB:DATA:HIGH <string>

  :SBUS:USB:DATA:HIGH?

- Functional description:

  Set high trigger data of USB data pack, parameter is character string of 0x format, data range 0x0~0xFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF.

- Return format:

  Query return character string of 0x format.

- For example:

  :SBUS:USB:DATA:HIGH "0x5F" set positive data of USB signal line as 0x5F

  :SBUS:USB:DATA:HIGH? query return 0x5F

:SBUS:USB:ERRor:TYPE

- Command format:

  :SBUS:USB:ERRor:TYPE {PID | TOKEN | DATA | BITFILL}

  :SBUS:USB:ERRor:TYPE?

- Functional description:

  Set error mode of USB.

- Return format:

  Query return{PID | TOKEN | DATA | BITFILL}.

- For example:

  :SBUS:USB:ERRor:TYPE PID set error mode as PID

  :SBUS:USB:ERRor:TYPE? query return PID

Appendix 1：<Key> Table

Key	Functional Description	LED
AUTO	Auto setting to display waveform	
RS	Control the operating status of oscilloscope, continuous to send the command, oscilloscope will switch between stop and run status	√
TMENu	Trigger menu	
SINGle	Single trigger	√
TFORe	Force trigger	
HELP	Help system	
HMENu	Horizontal system menu	
DISPlay	Display menu	
MATH	mathematical operation function and meue	√
REF	Reference waveform function and menu	√
CH1	The switch of channel 1	√
CH2	The switch of channel 2	√
CH3	The switch of channel 3	√
CH4	The switch of channel 4	√
F1	Select the first menu item	
F2	Select the second menu item	
F3	Select the third menu item	
F4	Select the fourth menu item	
F5	Select the fifth menu item	
MENu	Menu display function	
PSCReen	One-button print or One-button save screen image	
MEASure	Measurement function	
CURSor	Cursor measurement function and menu	
ACQuire	Sampling menu	
STORage	Storage menu	
UTILity	Secondary system help menu	
CLEar	Clear display waveform	
DECode	Decode menu	
DEFault	Recovery default setting	
FKNob	Multifunction knob	
FKNLeft	Multifunction left knob	
FKNRight	Multifunction right knob	
VPKNob	Vertical postion knob	
VPKNLeft	Vertical postion left knob	
VPKNRight	Vertical postion right knob	